AUTOMATIC HIGHER ORDER MESH GENERATION AND MOVEMENT UTILIZING SPRING-FIELD AND VECTOR-ADDING

By

Tuo Liu

C. Bruce Hilbert
Algorithms/Optimization
Branch Technology
(Chair)

Kidambi Sreenivas
Associate Professor of
Mechanical Engineering
(Committee Member)

James C. Newman III
Professor of Computational Engineering
(Committee Member)

Steve L. Karman Jr.
Staff Specialist
Pointwise
(Committee Member)

AUTOMATIC HIGHER ORDER MESH GENERATION AND MOVEMENT UTILIZING SPRING-FIELD AND VECTOR-ADDING

By

Tuo Liu

A Dissertation Submitted to the Faculty of the University of
Tennessee at Chattanooga in Partial Fulfillment
of the Requirements of the Degree of
Doctor of Philosophy in Computational Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

December 2017

ABSTRACT

In this research, an automatic unstructured curved 2D mesh generation method is developed. It can handle complex geometries. Based on this 2D method, an extension to 3D geometries is developed. The methodology of this mesh generation is to provide near-body and off-body meshes by an advancing-layer method to avoid mesh quality problems in transition areas, while using spring models to complement the advancing-layers by smoothing the mesh, especially in medial axis regions. After generating the linear mesh, a higher-order finite-element ready, curved mesh is obtained by deforming the linear mesh through "pipes" using a simple Vector-Adding approach. Different from most curved viscous mesh generation approaches, this method eschews a linear or nonlinear elasticity analogy while still providing positive-Jacobian mesh elements. In addition, the transformation from linear to curved meshes can be achieved by only deforming the necessary edges of the elements near domain boundaries while retaining the remaining non-curved edges to the benefit of numerical solvers. The CFD results of 30P30N airfoil on the curved mesh are given and are in good agreement with the experimental data.

Further, a new method for moving boundary problems with viscous mesh layers is also presented. This method is an extension of the mesh generation approach above. It can be used not only for high-order mesh moving but also for high-order mesh generation. Also, based on the curved mesh deformation strategy, the method can handle high-order mesh movement or regeneration with minimal extra computational time compared with linear mesh movement. Several 2D boundary motion cases are tested including boundary translations, boundary rotations,

and boundary morphing. The CFD results on the curved mesh after rotation are given and are in good agreement with the experimental data. This technique is also tested through the optimization of an airfoil to achieve a maximized lift and drag ratio. The results demonstrate that this approach can handle large boundary movements while preserving a good mesh quality.

DEDICATION

This dissertation is dedicated to my parents. Mom and Dad, I love you.

ACKNOWLEDGMENTS

I would like to express my special appreciation and thanks to my advisor Dr. C. Bruce Hilbert for devoting a great deal of time to answer my questions and offer suggestions, and his company Branch Technology for supporting him to do this. I also would like to express my gratitude to my previous advisors Dr. Li Wang and Dr. Steve Karman for their guidance and precious advice. Without their guidance and encouragement throughout my time as a graduate student, this work would not have been possible. I also thank my other committee members, Dr. Kidambi Sreenivas and Dr. James Newman for their guidance regarding this work. In addition, I would like to express my appreciation to Dr. Timothy Swafford for his support and for believing in me.

My gratitude also goes to my wife Siman Wei. I cannot imagine sailing my vessel without her support and encouragement.

Finally, I would like to thank all the teachers who impart knowledge to me, all my classmates with whom I study and play soccer and basketball, all the staff who offer me help and care, and all my friends, especially my friends in Chattanooga, who guided me towards more scenic lookout points when I hit the rocks.

You light up my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CFD, Computational Fluid Dynamics

AFSF, Advancing-Front-Spring-Field

DG, Discontinuous Galerkin

SUPG, Streamline Upwind Petrov Galerkin

LIST OF SYMBOLS

g_{given}, Growth ratio in the boundary mesh generation

G_{ratio}, Growth ratio in the domain mesh generation

l_p, Node spacing

κ_{apro}, Approximated curvature

l_{extru}, The first extruding length

l_{α}, Local spacing length limitation

F_{extru_eg}, Force of the extruding edge spring

F_{nei_eg}, Force of the neighboring edge spring

F_{ang}, Force of the angle spring

$F_{tension}$, Force of the tension spring

K_{extru_eg}, Extruding edge spring coefficient

K_{nei_eg}, Neighboring edge spring coefficient

K_{ang}, Angle spring coefficient

$K_{tension}$, Tension spring coefficient

l_{ideal}, Ideal length of an edge

l_{real}, Real length of an edge

C_{aniso}, Anisotropic coefficient

l_{charac}, Characteristic length

C_p, Static Pressure coefficient

C_f, Skin-friction coefficient

Greek symbols

δ, Maximum length limitation

κ, Curvature

α, Angle

ξ, 2D Computational coordinate

η, 2D Computational coordinate

ξ_1, 3D Computational coordinate

ξ_2, 3D Computational coordinate

ξ_3, 3D Computational coordinate

CHAPTER 1

INTRODUCTION

In recent decades, computational design and analysis have become fundamental parts of industry. Generally, to obtain the desired accuracy of a target problem, a mesh discretization of the domain that represents the real-world object or phenomenon is needed in some reasonable fidelity. Unfortunately, man-hours spent on mesh generation in obtaining a numerical-solution-based analysis are too enormous. And the necessity for manual intervention rises as the complexity of the target object's phenomenon or the geometry increases. This process calls for automatic mesh generation and automated mesh refinement and leads to a significant amount of research in both areas.

There are various standards to measure mesh quality. A good mesh should first be capable of representing the target object or phenomenon to an acceptable fidelity. Furthermore, it is usually desirable for high quality meshes to have as few elements as possible while maintaining a satisfactory level of fidelity representing the target geometry. As such, for many decades aerospace researchers have put a lot of time and effort into research regarding anisotropic meshes, hybrid meshes and high-order (curved) meshes. Let us first take a brief look at these three types of meshes.

In aerospace research, high Reynolds number flow simulations require anisotropic meshes to represent the domain in the boundary layer. A considerable volume of work has been published on the development of mesh algorithms for this purpose, such as the advancing front/layers approach[1-6], which generates the boundary layer mesh prior to the interior domain mesh as

shown in Fig. 1, or the bubble mesh approach[7], which generates the entire mesh isotropically then adjusts the mesh in the boundary layer to be anisotropic. However, compared with isotropic meshes generation, the generation of quality anisotropic meshes is far from mature. Many challenges remain. One such challenge relates to an overall and simultaneous mesh quality control in both the anisotropic boundary layer region and the off-body isotopic region. The quality in one region is often compromised while attempting to enforce quality elements in the other region. To overcome these obstacles, some researchers choose a mesh type that contains multiple elementary mesh elements: a hybrid mesh.

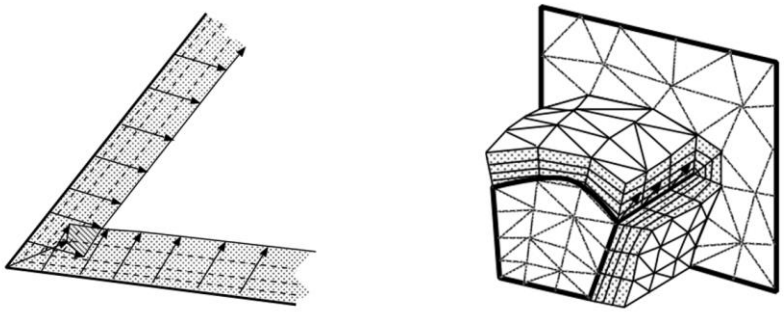


Figure 1 Advancing-layer method[5]

As a mixed-element mesh, a hybrid mesh has multiple advantages deriving from the contained elementary mesh type. First, it benefits the solver by using fewer mesh cells compared with pure triangular or tetrahedral mesh. Second, for some hybrid meshes, certain numerical schemes could be used to accelerate the simulation[8]. Third, hybrid meshes generally improve the accuracy of numerical simulations, especially with anisotropic viscous boundary layer meshes. A typical hybrid mesh approach is as shown in Fig. 2, which is introduced in Kallinderis' work[9]. In this article, prismatic layers are created around objects and the remaining gaps are filled with

tetrahedra, which have the capability to fill a given volume of arbitrary shape. This mesh structure contains quadrilateral faces normal to the surface to provide good orthogonality and mesh clustering capabilities, whereas the triangulation in the lateral direction allows flexibility in surface modeling[10]. This approach is robust and flexible enough to handle quite a range of different cases. Nevertheless, many challenges remain in mesh quality control of prism layers for complex boundary geometries, as well as in tetrahedral areas which are constrained by the shape of gaps. Researchers have also developed other types of hybrid meshes, containing hexahedral and prismatic elements mixed with pyramids and tetrahedra[11, 12], or Cartesian meshes containing triangular/tetrahedral elements[13, 14]. However, since these hybrid mesh approaches generate the boundary layer mesh and off-body mesh separately, they all face the same challenge: mesh quality control at the interface of two different types of meshes.

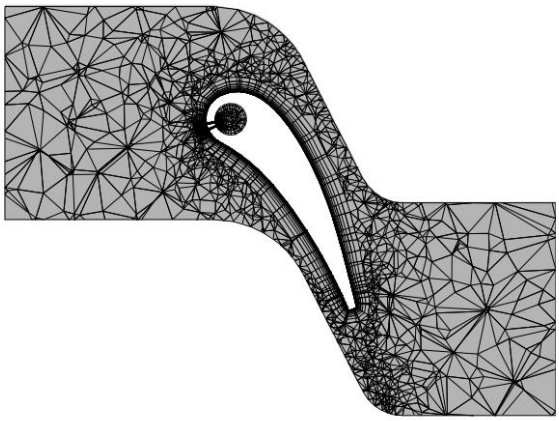


Figure 2 A typical hybrid mesh[9]

Also, due to the increasing popularity of high-order finite element solvers, much attention has been directed towards to the generation of curved meshes. Previous research on this includes works from Shephard, Sherwin and Persson[15-17]. A typical curved mesh is as shown in Fig. 3.

A common idea of obtaining a curved mesh is to deform an existing linear mesh into a curved one with all positive-Jacobian mesh cells. For the transformation from linear to curved meshes, the main obstacle of automation is that the conversion from straight-side edges to curved edges not only concentrates on curved geometry boundaries, but also on the edges of the adjacent elements and elements in their adjacent regions. It is imperative to ensure that all elements are untangled and the Jacobian in each mesh element is positive after the conversion. Previous research on this includes works from Shephard and Sherwin[16, 18]. While regular linear smoothing techniques sometimes fail to deliver positive-Jacobian elements, some nonlinear methods can address this uncertainty, such as nonlinear elasticity analogy[17]. Nevertheless, computing a non-linear mechanics problem, involving large deformations on a high-order and highly stretched mesh is at least as complex as, solving Navier-Stokes equations on the same mesh[19].

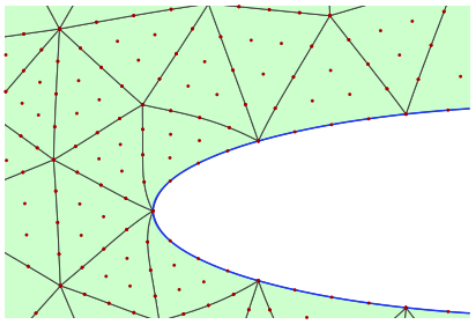


Figure 3 A typical curved mesh[17]

All in all, an automatic high-order mesh generation procedure with good mesh quality is still challenging. The difficulty of an automation even increases when dealing with viscous flow simulations, which generally requires anisotropic meshes near solid-body boundaries. The difficulty of automations also increases when the meshes are provided for high-order finite element solvers, which generally requires curved meshes. But the viscous mesh and the high-order mesh

generation are important that they cannot be ignored for their difficulty level. Hence, the automatic mesh generation for a high-order finite element solver[20] in viscous flow simulations is a challenging and meaningful research topic.

The whole process of curved viscous mesh generation can be generally separated into three steps. The first step is straight-sided viscous mesh generation. The second step involves constructing a high-order boundary representation to accurately account for the curvatures of the geometry. The third step is the deformation of the linear mesh into a curved mesh to accommodate the high-order curved boundary mesh generated in the second step. The difficulty of automatic curved mesh generation normally lies in two parts: one is to automatically generate a linear viscous mesh; the other is to automatically transform this linear mesh to a curved mesh. Much work has been done in these two parts, while obstacles remain to be addressed.

For the linear viscous mesh generation, the main obstacle of automation is to create a fine quality mesh while ensuring automation. If the boundary layer mesh and the domain interior mesh generation are handled differently, a boundary layer mesh of fine quality can be achieved by using an effective method, such as the advancing front methods[3, 5, 21], PostBL[22], or linear-elastic smoothing[23]. However, this type of approach separates the mesh generation into two parts, and it is difficult to control the mesh quality in the transition area. On the other hand, if the boundary layer mesh and domain interior mesh are handled by the same approach that is naturally suitable to domain interior mesh, like the bubble method[24, 25] or interacting particle system[26], the whole domain mesh generation could be more automatic, but anisotropy will not fully meet the stringent numerical requirements of boundary layer mesh, in which high-gradient and non-linear physics such as turbulence can be involved. In turn, if boundary layer mesh and domain interior mesh are handled by the same approach that is naturally suitable to the boundary layer mesh, such

as inflating one layer at a time[3, 5, 21], the anisotropy can be satisfactory. However, it will be challenging to control mesh quality in areas of front edge merging, and highly uneven elements can be generated as a result.

In computational design and analysis, another major challenge is to handle moving boundaries and interfaces. Many applications, such as fluid structure interaction (FSI) problems, biological fluid mechanics and geometric designs via optimization procedures, must face this challenge. In all these cases, the spatial domain moves or deforms with time. There are several methods to handle this, for instance, to use radial basis interpolation functions[27]. The common method, however, is to discretize spatial domains through meshing. The following is a review of the methods in this category.

The meshing methods used to handle moving boundaries can generally be divided into two classes: boundary-fitted (boundary-conforming) methods and non-boundary-fitted (non-boundary-conforming) methods. The boundary-fitted method is to change the mesh shape to fit the changed geometries. This type of method could be divided into two sub-classes: mesh-topology-changing and mesh-topology-unchanging. The mesh-topology-changing method is to change the mesh topology to fit the changed geometries. For this type of method, the mesh can either be regenerated[28], which is generally inefficient, or restructured[29]. The mesh-topology-unchanging method can generally be divided into two sub-classes, by the strategies of deforming the mesh: physical analogy and interpolation[30]. The physical analogy approach describes the mesh deformation by a physical process and then models this process to deform the mesh. One representative method in this sub-class is the spring method. The spring approach views the mesh as a network of springs. After each step of boundary motion, the mesh is deformed into a new shape through the balance of spring forces. The pioneer of this method is Batina[31]. A torsional

spring method was introduced by Farhat et. al[32, 33] to prevent cell collapse. Similarly, several elasticity-based approaches have been developed[34-36]. However, this kind of method can only handle small motion. Sometimes local remeshing is still necessary. Otherwise, the interpolation schemes distribute the domain nodal displacements based on the distance from boundaries. Representative methods in this sub-class are The Radial Basis Function (RBF) method proposed by Broomhead and Lowe[37], and the Inverse Distance Weighting (IDW) function method proposed by Witteven and Bijl[38], where the IDW function was defined by Shepard[39]. However, under this type of approach, mesh quality needs to be improved.

In the non-boundary-fitted group, there are immersed boundary (IB) methods[40-42] and Cartesian Mesh methods[43-47]. Peskin[40] first proposed the terms of IB to simulate cardiac mechanics and associated blood flow. The IB methods that simulate viscous flows with immersed (or embedded) boundaries on meshes do not conform to the shape of these boundaries [48]. These methods are similar to the Cartesian mesh method in that they do not conform to the boundaries. Cartesian mesh methods were originally developed for simulating inviscid flows with complex embedded solid boundaries[43-45]. A typical Cartesian mesh is as shown in Fig. 4. Udaykumar[46] and Ye et al.[47] then extended the method to simulate unsteady viscous flows. The primary advantage of non-boundary-fitted methods lies in the great simplification of the mesh generation. On the other hand, the difficulty in these methods is the choice of accurate interpolation schemes.

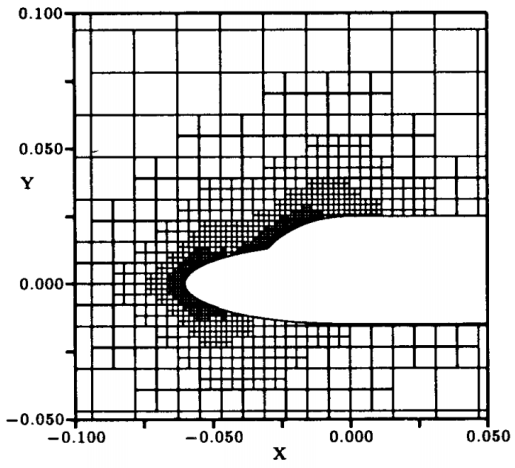


Figure 4 A typical Cartesian mesh[45]

Besides all the methods above, several mixed methods have been developed from various methods to inherit their complementary advantages, such as overset mesh methods[49, 50], a dynamic hybrid mesh method[51] and a multi-zone moving mesh algorithm[52]. The disadvantage is, with mixed-type methods, multiple spatial domains are generated separately, which cause challenging problems in the junction area.

To overcome these barriers mentioned above, this research develops an automated mesh generation method for high-order numerical solvers used in viscous flow simulation. The method generates a linear mesh layer by layer with the advancing-front concept. Then based on this linear mesh, which has a special “pipe-like” structure, it is easy to generate the high-order curved mesh by deforming the linear mesh with a simple Vector-Adding method. To complement the advancing-front approach, a force model named Spring-Field which contains several types of springs is used to smooth the mesh. Hence the linear domain mesh generation is based on the advancing-front method and this Spring-Field smoothing method. For convenience in this research, this linear domain mesh generation method was named Advancing-Front-Spring-Field

method (AFSF). Spring-Field and Vector-Adding will be introduced in Chapter 2. This automatic mesh generation method has several advantages compared with former methods: it provides a better mesh quality in view of limitation on obtuse triangles. It is capable of handling anisotropic, complex-geometry cases with a smoother transition area from the viscous layer mesh to the interior mesh. It can generate all positive-Jacobian high-order meshes and the computational cost of the deformation from a linear mesh to a curved mesh is relatively negligible. It only curves required mesh edges and keeps remaining straight edges to benefit numerical solvers. It can generate hybrid mixed-element meshes with complex geometries and can be extended to 3D cases.

To generate 2D meshes, three main procedure functions in sequence: the boundary mesh generation procedure, which sets the number of nodes and node spacing on the domain boundaries to produce edge meshes; the linear domain mesh generation procedure; and the curved mesh deformation procedure, which deforms the linear domain mesh to a curved mesh. These three procedures will be introduced in Section 3.1, 3.2 and 3.2. In Section 3.4, results will be presented to demonstrate the performance and capabilities of the high-order viscous mesh generation approach. Section 3.5 describes its potential of the approach to extend to 3D cases.

This spring approach views the mesh as a network of springs and can, thus, be used for mesh movement. As a spring model, Spring-Field is readily extensible to mesh movement or morphing. One advantage of this mesh movement method is that, it can handle cases containing viscous anisotropic boundary mesh layers with large moving or morphing distances. In Chapter 4, the extension for 2D mesh movement and morphing will be introduced. Section 4.1 describes the procedure of the mesh movement method. Section 4.2 describes the spring force model used for the mesh movement purpose. Note that this spring force model can also be used for mesh generation. Section 4.3 describes several cases used to test this mesh movement method: mesh

generation cases with complex geometries and mesh moving cases with large deformations. In Section 4.4, several more tests will be also implemented to check the potential of this mesh movement method in the optimization application.

The extension of the mesh generation method to 3D cases will be introduced in Chapter 5. Since for 3D cases, the linear domain mesh generation method has special requirements for input surface meshes, the surface mesh as an input must be qualified for the linear domain mesh generation procedure. Section 5.1 describes how to obtain these special surface meshes for linear volume mesh generation. Section 5.2 describes the linear 3D domain (volume) mesh generation procedure. Section 5.3 describes the curved 3D volume mesh generation procedure. In Section 5.4, mesh results are presented to demonstrate the performance and capabilities of this 3D high-order viscous mesh generation approach. In section 5.5, mesh results are presented to demonstrate the performance of the method in front-closing cases. The CFD results based on the meshes created by this method are presented in Section 5.6.

Two papers are published based on the works in Chapter 3[53, 54]. One research note and one paper are published based on the works in Chapter 4[55, 56]. One paper is published based on the works in Chapter 5[57].

CHAPTER 2

SPRING-FIELD AND VECTOR-ADDING

2.1 Spring-Field

As introduced in Chapter 1, there are multiple methods to generate a linear mesh. If the linear mesh generation involves a viscous mesh layer, one common choice is the advancing-front method. This method generates domain mesh layer by layer from the boundary mesh. Its disadvantage is that it is hard to control the mesh quality around the medial-axis area where two mesh fronts are merging, which probably means a mesh smoothing procedure is needed once the advancing-front method has been employed to generate the mesh.

One of the mesh smoothing methods is the spring approach, which views the mesh as a network of springs. Generally, the mesh starts with the original shape, which needs to be smooth; at this stage, the force of springs is off balance as designed. Then the whole mesh is deformed into a new shape through the balancing of spring forces. Among all the spring approaches, the linear-spring method is probably the most popular one. It is robust and easy to implement. Its disadvantage is that when a mesh element has a negative area (in 3D it is a negative volume), the traditional linear-spring method cannot guarantee to smooth this element into a positive-area mesh element. Because of this inconvenient feature, each moving step must be a relatively small one to avoid negative mesh elements, which leads to a larger time cost. To overcome this obstacle, a new spring force model is provided, which still treats the mesh as a network of springs. Only in the latter model does the network have a direction. The spring network is constructed layer by layer based on the boundary mesh, simultaneously with the new mesh layer constructed by the

advancing-front method. When the mesh is smoothed, only the top several mesh layers are involved. That can decrease the total time cost by providing a stand-still "root" mesh layer, which also benefits the mesh quality.

The concept of this force model is inspired by an interacting particle system[26, 58], spring force[32, 59, 60], and tissue-growth model[61]. This force model is denoted as Spring-Field, since it consists of numerous individual spring model units and covers the whole field. An individual spring model unit is demonstrated in Fig. 5b).

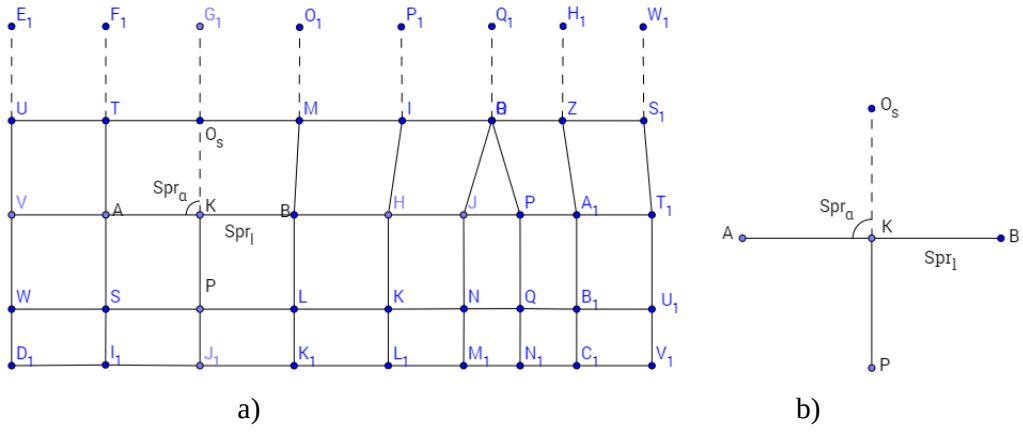


Figure 5 Spring model. a) Spring-Field; b) One single spring unit

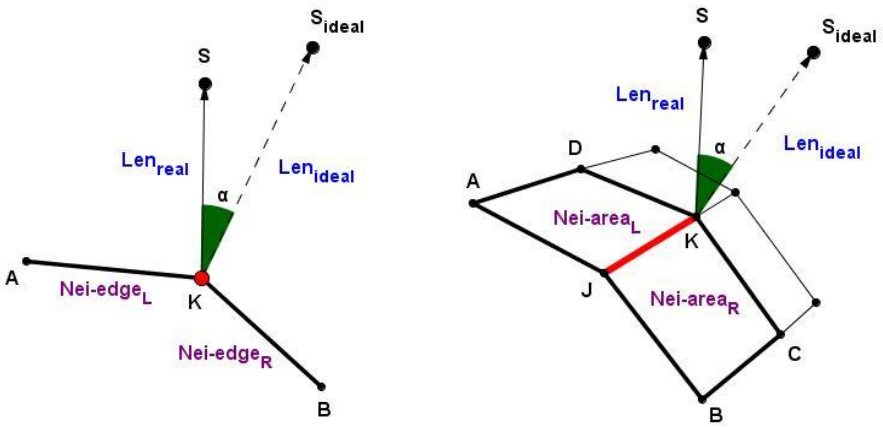
Fig. 5a) shows the Spring-Field diagram. If the mesh node *D1* is on mesh layer *n*, its child node *W* is on mesh layer *n+1*, and node *W*'s child node *V* is on mesh layer *n+2*. Fig. 5b) shows one spring unit that is a part of the Spring-Field in Fig. 5a). Spr_α and Spr_l are conceptual angle and edge springs, respectively. They are used to show that one spring unit could contain several different types of spring; they are not the actual spring force models used in the research. Node *K* is a central child node, which is generated by parent node *P*. Parent node *P* is located one-layer closer to the boundary mesh than node *K*. Nodes *A* and *B* are the left and right adjacent child nodes to child node *K*.

What functions should one spring unit have? First, to benefit the advancing-front mesh generation method, the spring model should push the existing mesh layer back to create enough space for the new mesh layers in case two fronts collapse. Second, the force of the spring model should make the node distribution more even as the mesh layer is advancing. Last but not the least, the spring model should have the ability to deform the negative-area mesh/negative-volume mesh element into a positive one. Based on these principles, the following three types of springs could be provided.

The first type is the edge spring model in the extruding direction of the domain mesh. The purpose of this spring type is to push the child nodes away from their parent nodes and the mesh boundaries towards their designed target positions. This type of edge spring links the parent and child nodes. In Fig. 6a) and b), there is one edge spring of this type, *KS*. It has a real length Len_{real} and a given ideal length Len_{ideal} obtained based on the ideal position S_{ideal}. The extrusion edge spring force is a function of the difference between Len_{ideal} and Len_{real}. If the child node S is too far from its ideal position, the edge spring could generate a push or drag force to drive the node to move closer to its ideal position.

The second type is the edge spring model in the neighboring direction. The purpose of this spring type is to dissipate the difference between the mesh elements along the neighboring direction on the same mesh layer. This difference originates from the difference between boundary mesh elements and it gradually dissipates as the mesh layer advances. The spring force is a function of the difference between the neighboring mesh elements and is to help dissipate this difference. For a 2D case shown in Fig. 6a), the neighboring edge spring attaches to the edges, which connect the adjacent child nodes. In Fig. 6a), there is one edge spring model of this type, shown as *A-K-B*. The spring force could be a function of the difference between the length of Nei_edge_L and

Nei_edge_R. If Nei_edge_R is longer, the neighboring edge spring force pushes the node *K* towards node *B* to shorten Nei_edge_R and stretch Nei_edge_L to decrease the difference between them. For a 3D case shown in Fig. 6b), the spring force could be a function of the areas of the two neighboring facets *AJKD* and *JBCK*. Again, this force should narrow the difference between the area of *AJKD* and *JBCK*.



a) b)

Figure 6 Diagram of spring models. a) 2D spring models; b) 3D spring models

The third type is the angle spring model. The purpose of this spring type is to keep the extruding direction of node *S* close to the direction of an ideal extruding direction $\overrightarrow{KS_{ideal}}$ to benefit mesh quality. Furthermore, this spring model will help to avoid negative mesh elements. The spring force is based on the angle between the real extruding direction and the ideal extruding direction. Fig. 6a) and b) both include one angle spring. The angle spring is a function of angle α between $\overrightarrow{KS_{ideal}}$ and $\overrightarrow{KS}$. $\overrightarrow{KS_{ideal}}$ is defined based on the mesh elements around the node *K*. For example, in Fig. 6a), $\overrightarrow{KS_{ideal}}$ can be defined as the average norm direction of edge *AK* and *KB*, which is also the direction of the bisector of angle $\angle AKB$. This angle spring force acts on both node *K* and *S* with opposite force directions, which creates a torsion force. The reason to distribute

this force to both nodes instead of only to child node *S* is that, by this design, the information of node *S* could transfer to node *K*. This means if the front mesh layer from another extruding direction meets node *S* and changes its position, the position of node *K* could be changed to improve the mesh quality in that area.

If the required number of nodes in the interior domain is much less than the number of nodes on boundaries, a strategy to decrease the number of mesh nodes as the mesh layer advances is needed. In case the number of mesh nodes on each layer needs to shrink as the mesh layer advances, the rules of springs are provided as follows.

1) Springs can be either active or inactive.
2) Edge springs that link adjacent child nodes can break.
3) If an edge spring breaks, the forces of the involved edge and angle springs will vanish.
4) If an edge spring breaks, the two nodes of this edge must overlap each other.
5) Overlapping nodes merge together as one node.

If two front mesh layers meet, several front nodes could merge. The merged node will have force from both front layers. Therefore, the nodes from different front layers will have interaction effects caused by Spring-Field force models. By these interactions, the improvement of mesh quality in the medial-axis area has been achieved, which is usually challenging to accomplish with advancing front methods. Another advantage of this spring force model is that, as shown in Fig. 6, the spring models in 2D and in 3D are similar. Hence, after finding proper expressions of spring models for the 2D cases, the extension to the 3D expressions of spring models is not very difficult to achieve.

One drawback of using force models to smooth meshes is that it does not always guarantee a perfect force balance, which indicates that nodes may keep on oscillating and a full convergence

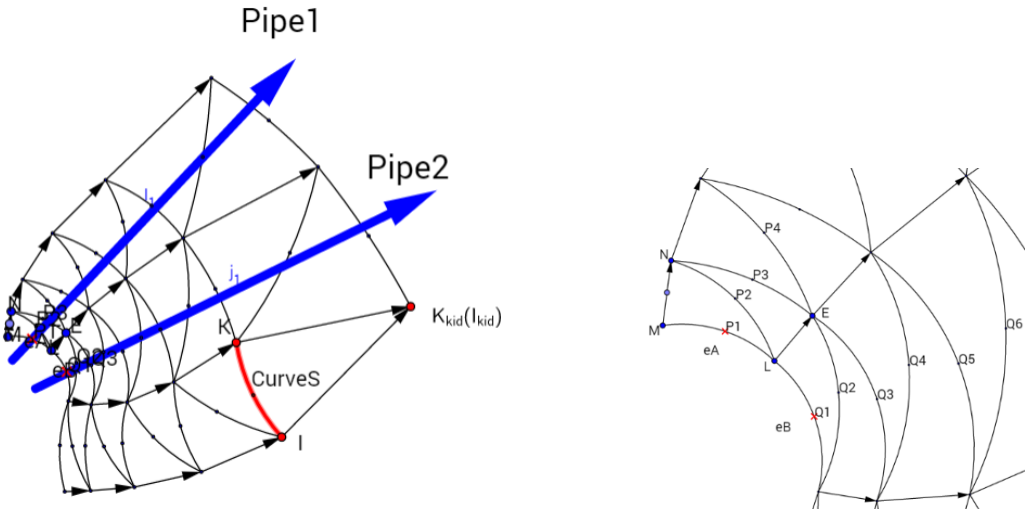
will never be achieved. The Spring-Field model faces the same challenge. However, the mass of each node, as a parameter in the force model, is introduced to dampen oscillation and increases the probability of a force balance. In this manner, if the balance of one node is not achieved after the total travel distance of this node increasing to a certain value, the mass of the node (with the initial value set to be unity) increases by a given factor, and the total travel distance of this node is reset to zero. By doing this even stubborn oscillations are numerically annihilated by the enormous mass of the oscillated node. In such circumstances, the relative moving distance can be an appropriate convergence condition instead of the sum of spring force.

2.2 Vector-Adding

2.2.1 What is Vector-Adding?

A Vector-Adding method serves to deform the linear mesh to the curved mesh. In general, the previous methods used for the deformation purpose, such as the one in Reference 17[17], provide new positions of deformed high-order nodes first. Then parametric curves and mesh elements are built based on the positions of the nodes. A Jacobian-checking process is necessary to detect negative-Jacobian elements. If a negative-Jacobian element is detected, this element needs to be deformed again and possibly the elements around it need to be deformed too, which can be time consuming. The proposed Vector-Adding deformation method, however, goes in an opposite direction. First, it tries to provide parametric positive-Jacobian mesh elements. Then, based on the parametric expression of these elements, it gives the new positions to the high-order nodes.

In a 2D case, the process of the deformation with Vector-Adding method is that, if a linear mesh has the "pipe-like" structure as shown in Fig. 7, the Vector-Adding method deforms the mesh elements one after another through the "pipe-like" mesh structure with stop conditions. In 2D, this deformation strategy can be employed for quadrilaterals and triangles. For example, if the target high-order mesh is quadratic, as shown in Fig. 7b), parametric triangle ΔLMN is constructed based on curved edge *eA* on the domain boundary and characteristic vector $\overrightarrow{MN}$ (the extruding non-curved edge *MN*). The new position of node P_2 is obtained based on the algebraic expression of the triangle. Next, since curved edge $\widehat{NP_2L}$ is known, parametric triangle ΔENL is built along with characteristic vector $\overrightarrow{LE}$ to provide node P_3. A recursive process is performed through the "pipe" until one of the stop conditions is satisfied. Note that the edges like *MN* and *LE* still maintain their non-curved shape. Benefiting from this building approach, the deformation procedure can be even easier and faster by simply adding the proper vector on node P_1 to provide the position of node P_2 (then by adding another proper vector on node P_2 to provide node P_3), without providing any algebraic expression of mesh elements. This makes the high-order mesh deformation process very competitive in terms of time cost. The details of how to obtain and add the proper vector to provide the position of the high-order nodes will be introduced in 2D mesh generation (Chapter 3).



(a) (b)

Figure 7 Diagram of curved mesh transformation

In a 3D case, the process of the deformation with Vector-Adding method is similar to the 2D case above. The mesh is deformed one element at a time through the pipe-like mesh structure as same as a 2D case. This similarity is shown in Fig. 8. For 2D, the new curved edge (dot dash) is obtained based on the two vectors *V1*, *V2*, and the existed curved edge (solid). For 3D, the new curved facet (dot dash) is obtained based on the four vectors *V1*, *V2*, *V3*, *V4*, and the existed curved facet (solid). The details of how to add the proper vectors to provide the new curved facets and to obtain the position of the high-order nodes will be introduced in 3D mesh generation (Chapter 5).

a) b)

Figure 8 Diagram of Vector-Adding model extension. a) 2D Vector-Adding; b) 3D Vector-Adding

2.2.2 Stop conditions

The earlier the deformation through the pipes stop, the larger number of straight edges remain in the mesh, which will not only decrease the total time cost of deformation but also benefit the numerical solver which uses the mesh. For this purpose, several stop conditions are provided. One is that two neighboring child nodes overlap each other. For example, in Fig. 7a), after curve *CurveS* is given, the deformation process at that "extruding pipe" stops because the edge that links nodes K_{kid} and I_{kid} is set to "break" status, and that leads to the two nodes overlapping each other. Another stop condition is, if two deformation processes from the opposite directions meet each other at the same mesh element, which means two "pipes" meet at the medial-axis area, both deformation processes will stop. For keeping the non-curved edges as much as possible to benefit the CFD solver, a Jacobian-checking process can be added as the third stop condition. If, without deformation, the target mesh element is already a positive-Jacobian element, the deformation process in that "pipe" will stop. However, a minimum number of deformed mesh layers is necessary to describe the domain close to the geometry in a satisfied precision. If a mesh element that belongs to the mesh layer whose index is smaller than the given minimum number of deformed layers, the deformation process will continue regardless of the Jacobian-checking stop conditions.

One limitation of this deformation approach is that if two "pipes" from the opposite extruding directions meet each other at a negative-Jacobian mesh element, based on the second stop condition both deformation processes stop and a negative Jacobian in the mesh element would remain. Even if both deformations operate, these two processes will have adverse effects on each other and a positive Jacobian of that mesh element cannot be guaranteed. That means if the extruding "pipes" meet at a mesh element that has a long and narrow shape in the boundary layer

area, without extra adjusting procedures being added into the routine, the high probable positive-Jacobian deduction cannot be used on that mesh element.

2.2.3 Positive Jacobian

The reason that the deformed mesh element has a high probability to be positive-Jacobian lies in the angle between computational coordinate isoclines. This will be explained in 2D mesh generation (Chapter 3) and 3D mesh generation (Chapter 5) respectively.

CHAPTER 3

2D MESH GENERATION

If geometry repair and reconstruction are not considered, the lowest level of curved viscous mesh generation hierarchy is the boundary mesh generation. In this research, the 2D mesh generation could be divided into three procedures: the boundary mesh generation procedure, the linear domain mesh generation procedure and the curved domain mesh generation procedure. The boundary mesh generation procedure provides boundary meshes for the linear domain mesh generation procedure and the curved domain mesh generation procedure. The linear domain mesh generation procedure provides linear domain meshes for the curved domain mesh generation procedure. The curved domain mesh generation procedure provides the curved mesh for the finite element solver in the application cases where the viscous boundary layer mesh is needed.

Section 3.1 introduces the 2D boundary mesh generation procedure and the meshes generated by this procedure. Section 3.2 introduces the 2D linear domain mesh generation procedure and the meshes generated by this procedure. Section 3.3 introduces the 2D curved high-order domain mesh generation procedure, the meshes generated by this procedure, and the CFD results based on the high-order mesh. In Section 3.2, the original AFSF and the improved AFSF methods are used to generate the linear mesh. In Section 3.3, Vector-Adding method is used to deform the obtained linear domain meshes, which are introduced in Section 3.2, to the curved high-order meshes. The linear and curved mesh results are analyzed in these sections.

3.1 Boundary Mesh Generation Procedure

3.1.1 Linear boundary mesh generation

The mesh quality in this phase is essential because errors, such as overly dense or sparse spacing, will be propagated up the mesh generation hierarchy[62]. A good boundary mesh can be measured using two views:

1) The curve segments with large curvature are described by sufficient nodes.
2) The growth ratio, i.e., the length of a segment divided by the length of a topologically adjacent segment[14] should deviate from unity broadly enough to keep representation efficiency, while being close enough to unity to avoid negative effects on numerical solvers.

Based on this analysis, it is reasonable to treat the mesh node spacing l_p, which is used for distributing nodes on boundaries, as a function of curvature κ and given growth ratio g_{given} :

$$l_p = f(\kappa, g_{given})$$

By such a definition, the node spacing l_p can reflect the two key features concerned (κ and g_{given}). In this research, to further extend the definition, the mesh node spacing l_p, as shown in Eq. (1), is defined as a function of g_{given}, approximated curvature κ_{apro}, maximum length limitation δ, refinement angle α_{refine}, and the first extruding length l_{extru} of 2D domain mesh generation. The advantage of this definition is that it allows l_p to reflect κ and g_{given}, which are the two key features, while taking 2D domain mesh generation (l_{extru}) into consideration. Furthermore, with a global input parameter α_{refine}, it could provide different levels of the total number of mesh nodes on boundaries.

$$l_p = f(\kappa_{apro}, g_{given}, \alpha_{refine}, \delta, l_{extru}) \qquad (1)$$

where the definition of approximated curvature κ_{apro} is the same as the one in Ref. [62]:

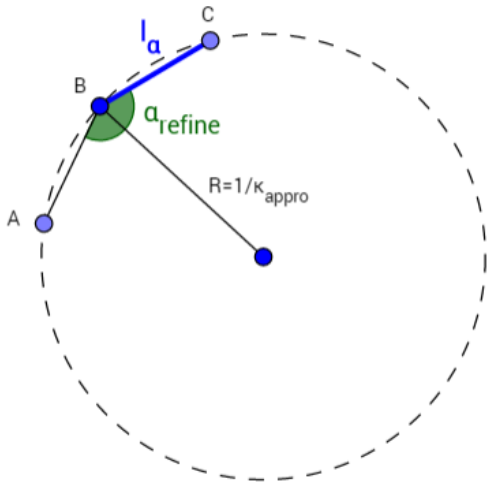


Figure 9 Approximated curvature κ_{apro}, refinement angle α_{refine} and local spacing length limitation l_α

$$\kappa_{apro} = \frac{1}{R} \qquad (2)$$

As shown in Fig. 9, *R* is the radius of the circle passing point *B* and its geometry neighbor points *A* and *C*. Since curvature κ_{apro} is used instead of the real curvature, the geometry points must be dense enough to allow κ_{apro} to represent the real curvature with a satisfied precision.

A local spacing length limitation l_α is introduced and defined as a function of radius *R* and refinement angle α_{refine}:

$$l_\alpha = f(\alpha_{refine}, R) \qquad (3)$$

where α_{refine} is angle $\angle ABC$ as a global input factor as shown in Fig. 9. Note that Eq. (3) is the basic definition of l_α. This local spacing length limitation l_α could be further adjusted in some special sections of the boundary, as discussed in the following main procedure for the boundary mesh generation. Based on Eq. (3), if node spacing l_p is a function of l_α, it is a function of α_{refine}.

The strategy of using α_{refine} to control l_p is similar to the approach described in Ref. [63]. The real expression of node spacing l_p in practice is:

$$l_p = f(g_{given}, l_\alpha, \delta, l_{extru}) \qquad (4)$$

By substituting (2) into (3) and (3) into (4), the Eq. (1) is obtained. Given this node spacing function, the domain boundary mesh generator is provided. The main procedure of the linear boundary mesh generation is as follows:

1) Calculate l_α of each discretized geometry point, and pick the break points to be the points whose l_α are smaller than their right and left neighbor points. Give l_p at each break point.
2) Take mesh quality in the medial-axis area into consideration in a refinement step (optional). Based on these break points, check l_α in each geometry point. If l_α is larger than length requirement $l_{require}$ that is calculated by g_{given} and the distance between the break point and the target geometry point, specify l_α of this geometry point as $l_{require}$. After checking all the geometry points, step 1 is repeated. At this stage, the boundaries have been separated into piecewise intervals by the break points.
3) For each interval, generate an unstructured 1D mesh. This generation is based on two l_p of the two nodes at both ends of each interval, l_α of each geometry point in this interval and growth ratio g_{given}. If κ_{apro} on each geometry node is in a satisfied precision and the curvature κ between any two neighboring discretized geometry nodes can be approximated by linear interpolation and is sufficiently precise, each node spacing should be smaller than the minimum l_α of geometry points on the interval matched to that same node spacing.
4) Combine all piecewise intervals to build a boundary mesh.

One limitation of this approach is, though the optional refinement step can improve mesh quality in medial-axis area, which is shown in the result section, it cannot guarantee the similar

size of mesh cells when nodes meet at the medial-axis area. The reason is that this refinement approach is only based on the distance between nodes but it does not take the mesh advancing directions of nodes into consideration. In the future, this step could be further developed.

3.1.2 Curved boundary mesh generation

For curved viscous mesh generation, a curved geometry is required. Base on it, a curved boundary mesh is generated. Based on the curved boundary mesh, the interior linear domain mesh is deformed. In this research, the curved geometries are described by their piecewise algebraic expressions. These algebraic expressions are obtained by building piecewise 6-degree Hermit curves with equal first and second derivatives on each input discretized geometry node. The first and second derivatives on each discretized geometry node are determined by building a Lagrange polynomial with the target node as the center of interpolation. Based on these curved geometries, the curved boundary meshes are obtained. The method used to generate linear boundary meshes introduced in the above section, is employed here to provide the curved boundary meshes. The only difference between generating a linear boundary mesh and a curved boundary mesh with this method is that, for a curved mesh, the algebraic expressions of the geometry curves are used to obtain the positions of the mesh nodes.

3.1.3 Mesh results

To verify the boundary mesh generation methodology proposed above, two examples are provided: a bat shape boundary mesh and the 30P30N airfoil boundary mesh.

3.1.3.1 Bat Shape Geometry

The wingspan of the bat is 60. The initial extruding length is 0.0005, which is used to provide l_p of the nodes at sharp corners in this boundary mesh generation procedure. Note that this initial extruding length is also used in the 2D mesh generation. The growth ratio g_{given} in this boundary mesh generation section is 1.3. The representations of parametric curves on the boundary are quadratic polynomials as same as the ones in the domain. The total number of extruding layers is 51.

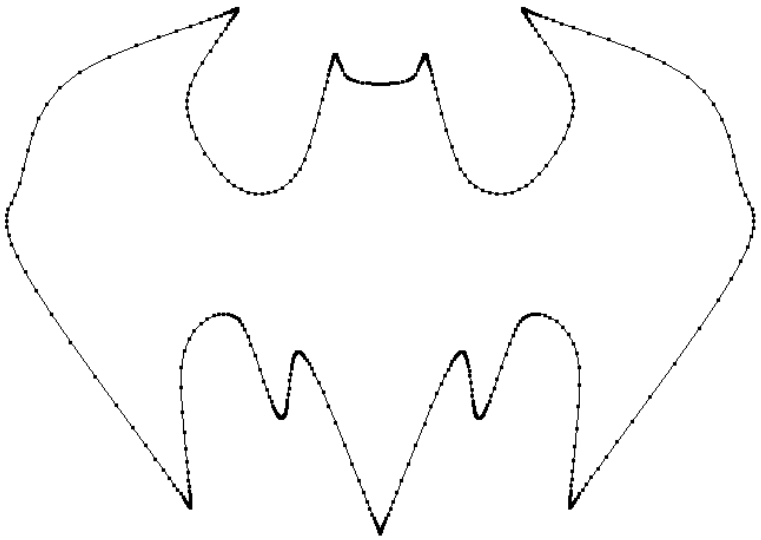


Figure 10 Bat shape linear domain boundary mesh

Fig. 10 shows the linear boundary mesh provided by the automated domain boundary mesh generation procedure. It illustrates that the boundary mesh generation method captures and describes the large curvature regions by inserting more nodes. Meanwhile, nodes are distributed efficiently and smoothly under the influence of the given growth ratio.

3.1.3.2 30P30N airfoil

30P30N airfoil is a Three-Element airfoil designed by NASA. The first extruding length in this example is 0.000005. The growth ratio g_{given} in this boundary mesh generation section is 1.3 or 1.4 in different test cases. As shown in Fig. 11, the linear domain boundary mesh has a smooth node distribution with more nodes along the large-curvature boundary sections.

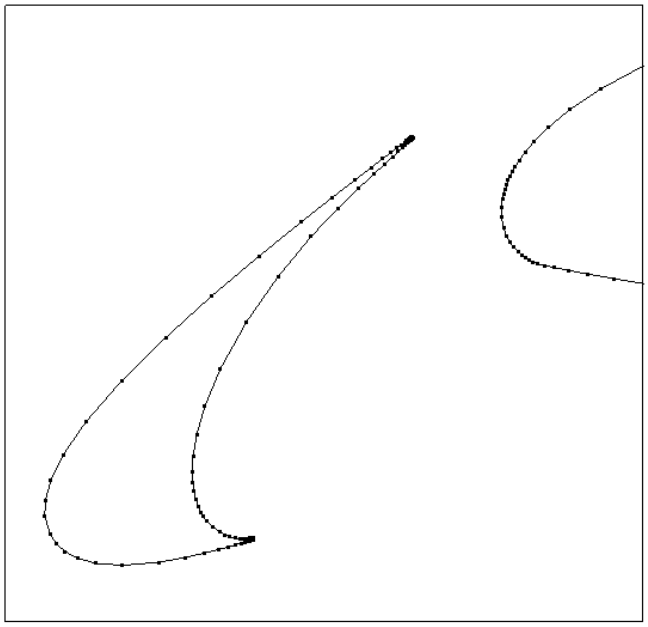


Figure 11 Close view of the slat of 30P30N airfoil quadratic boundary mesh

For curved boundary meshes, the piecewise quadratic and cubic polynomials are used to represent the piecewise curves. These results will be introduced in Section 3.3.2, which contains the curved domain mesh results.

3.2 Linear Domain Mesh Generation Procedure

Based on the obtained linear and curved boundary meshes, the linear and curved domain

meshes can be generated by AFSF method and Vector-Adding method. The mesh generator in this procedure produces unstructured linear domain meshes with the viscous mesh layer. The mesh elements in output meshes are naturally a hybrid type made of mixed quadrilaterals and triangles. This type of mesh requires vastly fewer elements than the pure triangular mesh to achieve the same degree of numerical resolution[64]. However, a pure triangular mesh can be provided by simply dividing all quadrilaterals into triangles.

The linear domain mesh generation are presented here using the Advancing-Front-Spring-Field method. Different from a classic advancing-front method, each inserted node and the nodes surrounding it have several interaction effects from a force model Spring-Field. Recall the Spring-Field force model described in Chapter 2. This force model smooths the mesh to avoid mesh quality problem caused by the advancing-front concept in medial axis area or in the sharp corner area.

In Section 3.2.1., the procedure of the linear domain mesh generation is introduced. In Section 3.2.2 and Section 3.2.3, the original AFSF method and the improved AFSF are introduced.

3.2.1 Procedure of the linear domain mesh generation

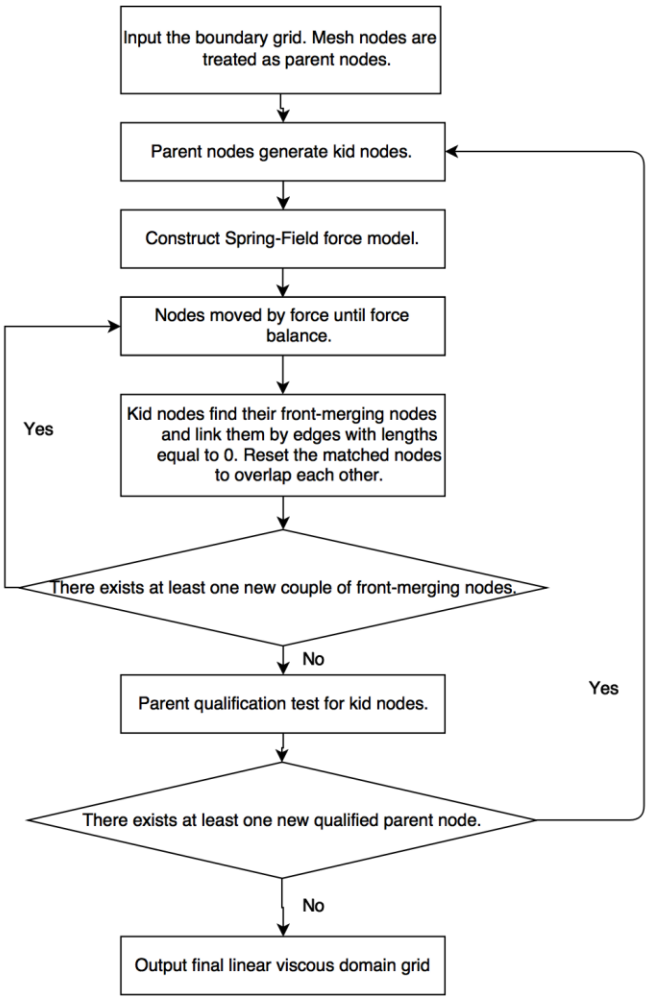


Figure 12 Procedure of the linear domain mesh generation

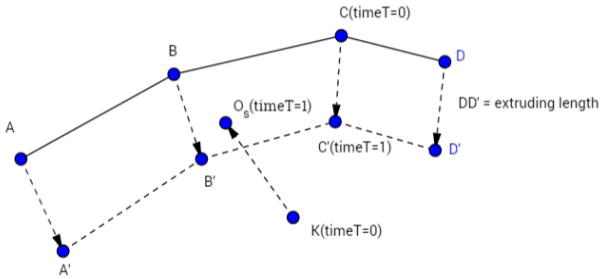


Figure 13 Front node merging sketch

Fig. 12 shows the main procedure of the linear domain mesh generation. The force

equations are solved explicitly. At the step where the spring force balances, there is no need to move all mesh nodes. Normally for a better mesh quality, three moving mesh layers are enough. The strategy of finding front-merging nodes is shown in Fig. 13. Suppose from time 0 to 1, kid node O_s moves from the position of its parent node *K* to its current position, which is obtained by adding the extruding vector on parent node *K*. A', B', C', D' are the child nodes the same as O_s. That means the edge *BC* moves from its initial position with time t=0 to $B'C'$ with time *t=1*. The distance Dis_{bc} between node O_s and the line that contains edge *BC* is a known function of *t*. Find the smallest $Dis_{bc\_smallest}$ at t_d in $t \in [0,1]$ and compare it with the other distances $Dis_{ab\_smallest}$, $Dis_{cd\_smallest}\dots$ to find the edge which has the smallest *Dis*. And if the smallest distance is smaller than a given global factor Dis_{judge} (for example, $Dis_{judge} = 0.1\overline{DD'}$), this edge is chosen to be the front-merging edge. Note that if at time t_d, the node O_s is not "on top of" the edge, the edge cannot be chosen as the front-merging edge. For this reason, in Fig. 12, edge $A'B'$ cannot be chosen as the target edge. After the front-merging edge is found, like $B'C'$ in Fig. 13, either node B' or C', whichever is closer to node O_s during time $t \in [0, 1]$, is selected to be the front-merging node to node O_s. Then the target node is linked to its front-merging node by a new edge with the status set to "break", which leads the two nodes (O_s and B') to overlap each other. This strategy is to stimulate a tissue growth circumstance: the tissue grows from the original surface layer by layer until it runs out of space.

3.2.2 Original AFSF method and mesh results

3.2.2.1 Spring models

Recall three types of spring could be designed for the Spring-Field model. They are

extrusion edge spring, neighboring edge spring, and angle spring. The three spring force models in the original AFSF method are described as follows:

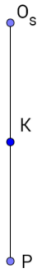


Figure 14 The edge spring model linking the parent and child nodes

The first is the edge spring model in the domain mesh extruding direction. This type of edge spring links the parent and child nodes. In Fig. 14, node *P* is node *K*'s parent node, node *K* is node O_s's parent. There are two edge springs shown in Fig. 14, *PK* and KO_s. Each of them has a given ideal length l_{ideal} and a real length l_{real}. The force model is

$$F_{extru\_eg} = K_{extru\_eg} \times \frac{l_{ideal} - l_{real}}{l_{ideal} + l_{real}}$$

where K_{extru_eg} is the extruding edge spring coefficient, which is set to be 1 in the numerical examples in the mesh result section. The ideal length l_{ideal} is a given global value that grows exponentially as the mesh layer increases. For edge KO_s, $l_{ideal}\ of\ KO_s = l_{ideal}\ of\ PK\ \times G_{ratio}$, where G_{ratio} is a given mesh extruding growth factor (e.g. 1.1,1.2,1.3… etc.). The force direction of edge spring KO_s is identical to vector $\overrightarrow{KO_s}$. The purpose of this spring type is to push the child nodes to the designed target positions, and away from their parent nodes and mesh boundaries.

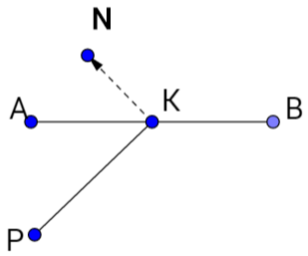


Figure 15 The edge spring model linking adjacent child nodes

The second is the edge spring model in the "neighbor" direction. This type of edge spring links the adjacent child nodes. In Fig. 15, there is only one edge spring model of this type, shown as A-K-B. The force model is

$$F_{nei\_eg} = K_{nei\_eg} \times \frac{l_{real}\ of\ AK - l_{real}\ of\ KB}{l_{real}\ of\ AK + l_{real}\ of\ KB}$$

where K_{nei_eg} is the extruding edge spring coefficient, which is set to be 1 in the numerical examples in the mesh result section. The force direction of this spring model *A-K-B* is as same as vector $\overrightarrow{KN}$, that is perpendicular to vector $\overrightarrow{PK}$, where node *P* is node *K*'s parent node. The purpose of this spring type is to dissipate the difference between lengths of neighbor-direction edges inheriting from the difference of lengths of the edges in the boundary mesh.

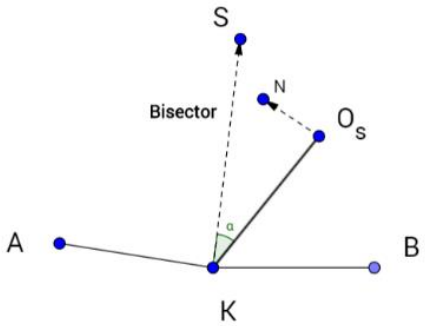


Figure 16 The angle spring model

The third is the angle spring model, as *A-K-B-*O_s shown in Fig. 16. The angle spring model contains four nodes *A, K, B,* O_s and three edges *AK, KB,* KO_s. Vector $\overrightarrow{KS}$ is the bisector of angle $\angle AKB$. Angle α is angle $\angle SKO_s$. The angle spring force can be specified by two different functions of angle α. One is a simple linear force function (linear in terms of α_{appro}). The force of angle spring is defined as

$$F_{A-K-B-O_s} = K_{ang} \times \frac{\alpha_{appro}}{Fac_A} \times Fac_{shape}$$

where K_{ang} is the angle spring coefficient, which is set to be 1 in the mesh result section. $\alpha_{appro} = 180\frac{sin\alpha}{\pi}$. $sin\alpha$ is used instead of α to decrease computation cost. Fac_A is a prescribed global coefficient used to control the stiffness of the angle spring and is set to be 20 in the mesh result section. As Fac_A decreases, the angle spring becomes stiffer. The Fac_{shape} is:

$$Fac_{shape} = min(Fac_B, \frac{l_{real}\ of\ AK + l_{real}\ of\ KB}{l_{real}\ of\ KO_s + l_{real}\ of\ KO_s})$$

It is used to distinguish anisotropic mesh elements from isotropic ones. As a mesh element becomes wider (larger $\frac{\overline{AK}}{\overline{KO_s}}\ or\ \frac{\overline{KB}}{\overline{KO_s}}$), its Fac_{shape} becomes larger, so the matched angle spring becomes stiffer. Fac_B is a prescribed global coefficient to control the range of Fac_{shape} and is set to be 50 in the mesh result section. It is used to avoid unnecessary large angle spring force caused by the highly anisotropic mesh element ($\frac{\overline{AK}}{\overline{KO_s}}\ or\ \frac{\overline{KB}}{\overline{KO_s}}$ is large). The other angle spring force function is a nonlinear one defined as:

$$F_{A-K-B-O_s} = K_{ang} \times \{tanh[Fac_C(\alpha_{appro} - Fac_{shape}) - tanh[Fac_C(-\alpha_{appro} - Fac_{shape})]\}$$

where $Fac_{shape} = min(Fac_D, \frac{Fac_E}{\frac{l_{real}\ of\ AK + l_{real}\ of\ KB}{l_{real}\ of\ KO_s + l_{real}\ of\ KO_s}})$ and Fac_C, Fac_D, Fac_E are given global coefficients used to control the stiffness of angle springs. As shown in Fig. 17, this force model

has a better control of α_{appro} due to the steep change, which could be adjusted by Fac_C. However, since the convergence speed is much slower than the first angle spring function, all the test cases in the mesh result section use the first linear function.

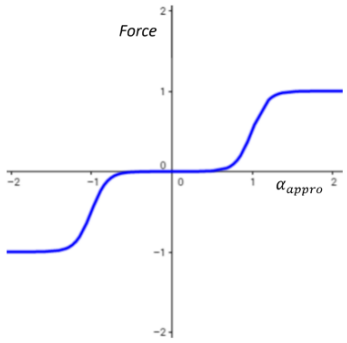


Figure 17 The angle spring model- type2

The angle spring contributes a force on node *K* as:

$$F\_K_{ang} = C_{percent} \times F_{A-K-B-O_s}$$

and a force on node O_s as:

$$F\_O_{s_{ang}} = (1 - C_{percent}) \times F_{A-K-B-O_s}$$

where $C_{percent}$ is a given global coefficient. The force direction of the angle spring on node O_s is the same as vector $\overrightarrow{O_sN}$ and the force direction of the angle spring on node K is the same as vector $\overrightarrow{NO_s}$. The purpose of this spring type is to keep the extruding direction of node O_s close to the direction of the bisector $\overrightarrow{KS}$ to benefit mesh quality.

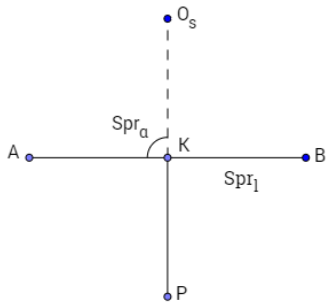


Figure 18 A spring model diagram

Note that if node K in Fig. 18 is on the front mesh layer, there will be no child node O_s of node *K* because node *K* is the youngest generation. On the other hand, when the two front layers meet, node O_s could be found from the opposite front layer, the edge spring in that position will be activated but the status is set to break (length=0) to force the node O_s to overlap node *K*. Therefore, the nodes from different front layers will have interaction effects caused by Spring-Field force models. By these interactions, the improvement of mesh quality in the medial-axis area has been made, which is a common challenge to advancing front methods. However, since in the original AFSF method, the node-edge structure only allows one node to have four nodes linked to it with edges, such as shown in Fig. 18, the node-overlapping strategy could lead to a robustness issue when multiple nodes overlap in one position. The improved AFSF method has solved this robustness issue, which will be introduced in Section 3.2.3.

3.2.2.2 Mesh results

To verify the original AFSF method, two examples are provided: a bat shape geometry and 30P30N airfoil. The first is the bat shape geometry. The linear domain mesh is generated based on the previous obtained bat shape linear boundary mesh. Two cases are tested with inward and outward extruding directions, respectively. The wingspan of the bat is 60. The initial extruding

length in both cases is 0.0005. The growth ratio in the domain mesh extruding direction is 1.3. The total number of extruding layers is 51.

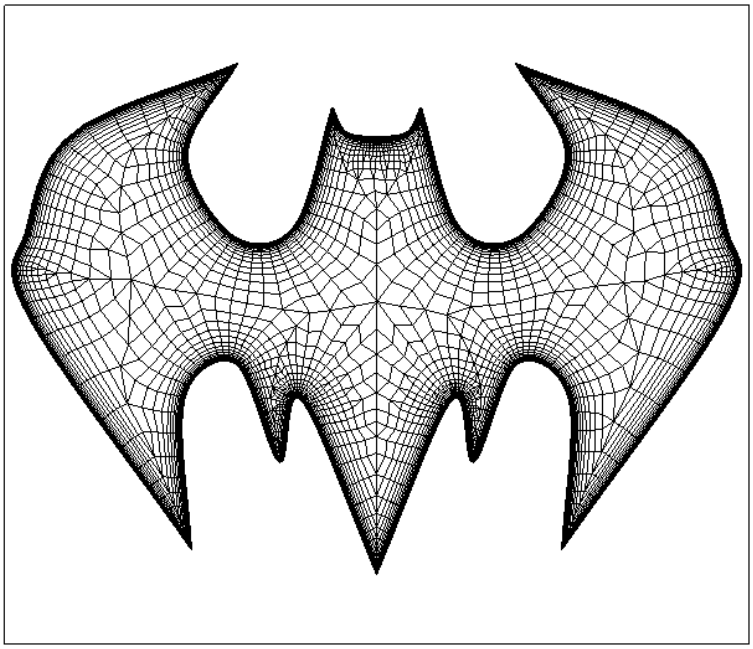


Figure 19 The bat shape inward extruding mesh with mixed elements

Fig. 19 shows the bat shape inward extruding mesh with mixed elements. It is a hybrid mesh. Most of the mesh elements are quadrilaterals, which could benefit the numerical solvers by decreasing computational cost. It also illustrates that the mesh quality is good in the medial-axis area.

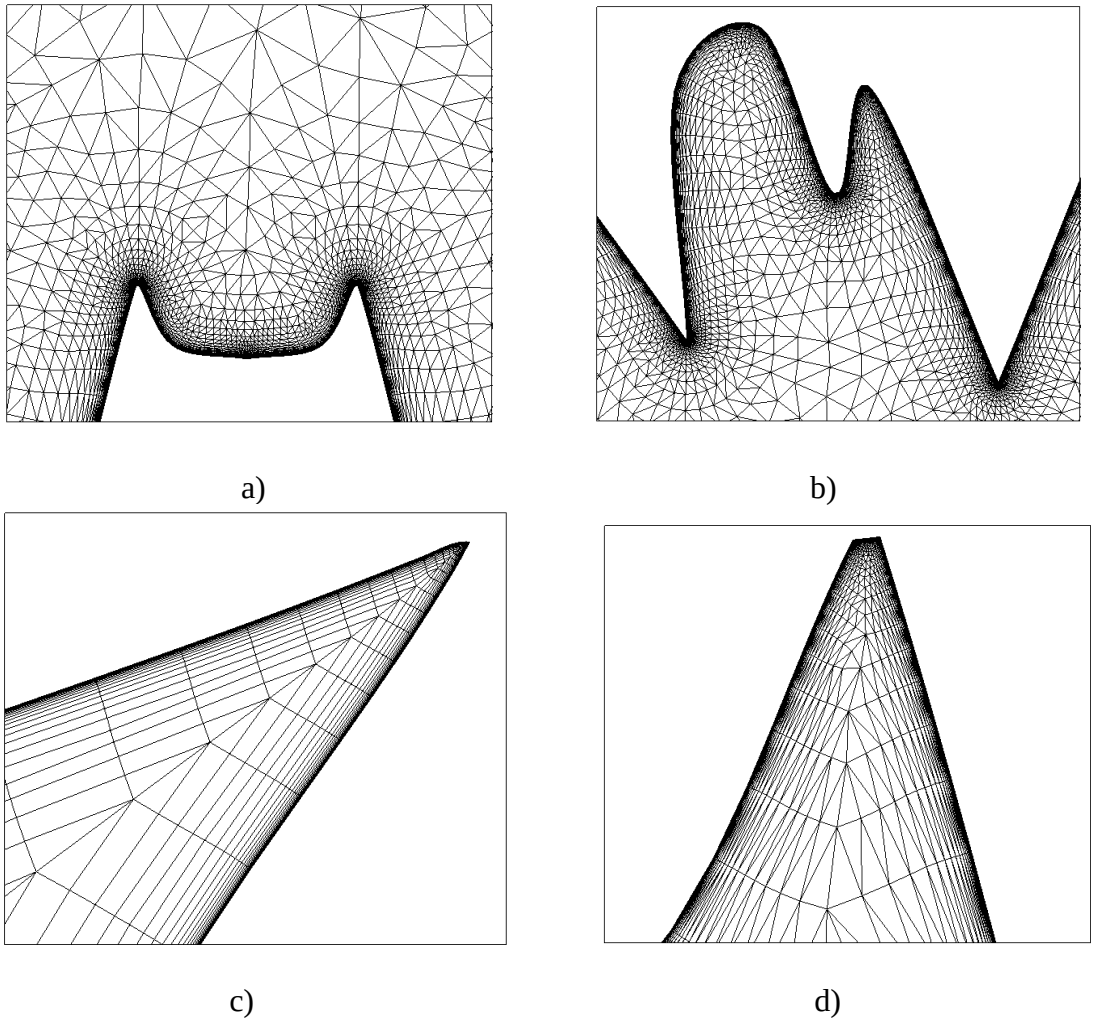
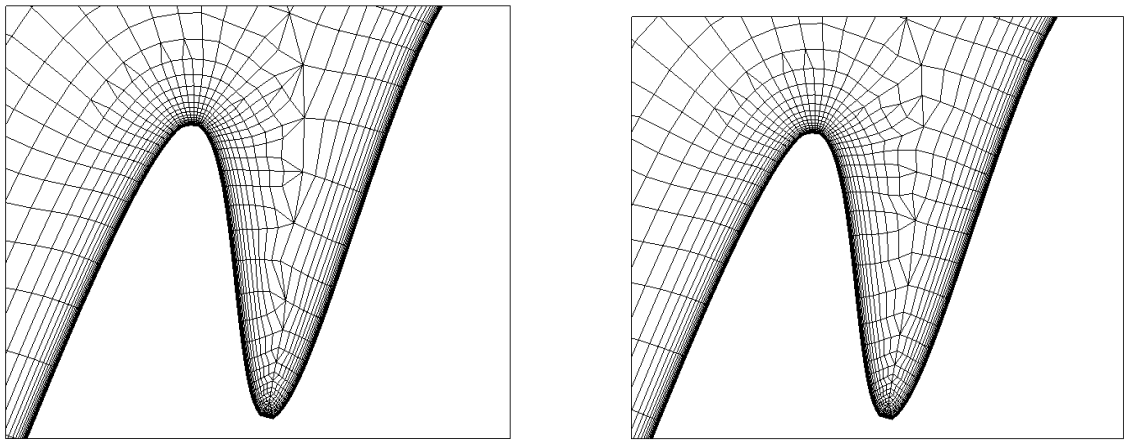


Figure 20 Close view of the bat shape inward and outward extruding meshes. a) head part of the outward mesh with pure elements; b) tail part of the outward mesh with pure elements; c) wing part of the inward mesh with mixed elements; d) head part of the inward mesh with mixed elements

Fig. 20 shows details of the linear domain viscous meshes as intermediate outputs. The meshes grow smoothly, and at medial-axis area, they are in satisfactory quality. In addition, there is no obvious transition areas between the boundary layer mesh and the domain interior mesh, which supports the good mesh qualit



a) b)

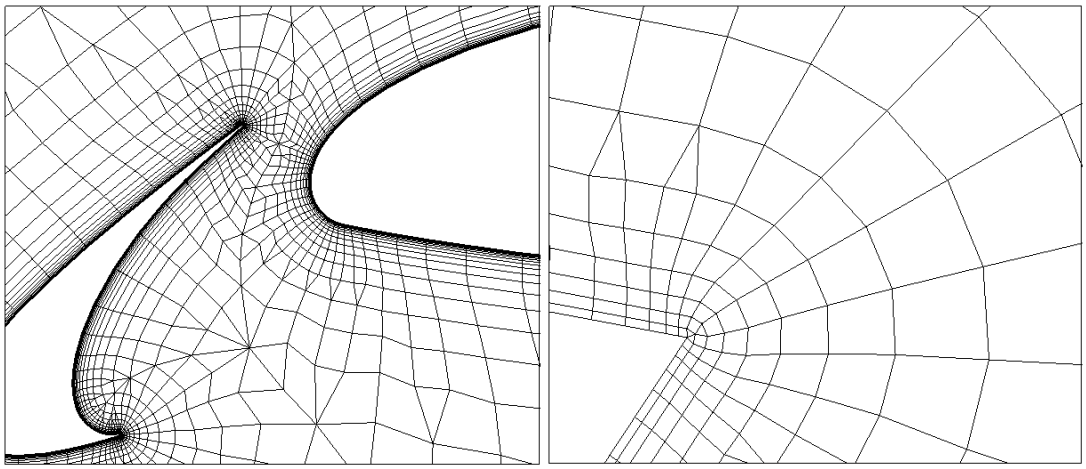
Figure 21 A part of the bat shape of the inward extruding meshes. a) Without the refinement step; b) With the refinement step described in Section 3.1

Fig. 21 shows the comparison between parts of the inward linear domain meshes with and without the refinement step described in Section 3.1. It illustrates that the mesh quality is improved due to a smoother change in cell sizes with the refinement step in the advancing front closing area. Note that the mesh in Fig. 21b) is an unfinished one. The whole mesh with the refinement step cannot be obtained by the same input coefficients of the unrefined one. A lack of robustness of the current code implemented based on the original AFSF method causes this situation and the input coefficients need to be changed to obtain a whole mesh in the refinement step.

Second, three 30P30N airfoil cases are tested. The first extruding length in all cases is 0.000005. The growth ratio g_{given} of the input boundary mesh is 1.4, 1.3 and 1.4, respectively. Note the boundary meshes used here are obtained with the previous boundary mesh generation procedure. The growth ratio in the domain mesh extruding direction is 1.4, 1.3 and 1.3, respectively. The total number of extruding layers is 51. The following figures in this section are all from case 1.

Fig. 22 shows details of the hybrid linear viscous mesh, and Fig. 23 shows the whole view.

These figures illustrate that the number of quadrilaterals is much larger than that of triangles, which could benefit the numerical solvers by decreasing computational cost.



a) b)

Figure 22 Close view of 30P30N airfoil linear mesh with mixed elements, a) the slat zone; b) the trailing edge of the main-element

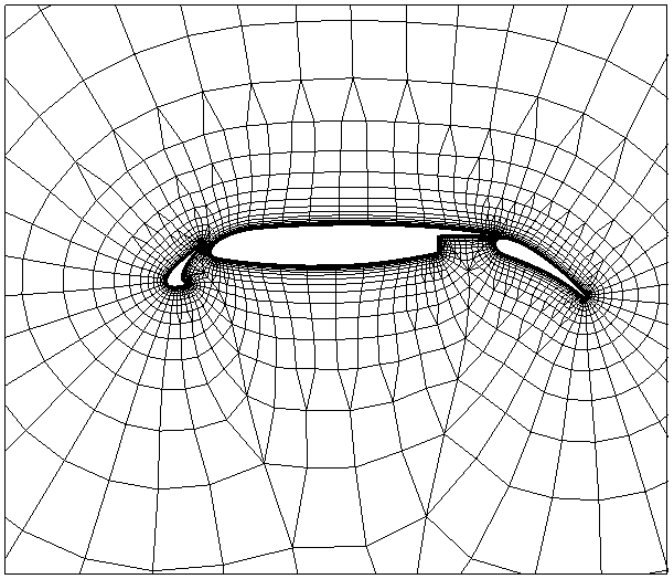


Figure 23 30P30N airfoil mesh with mixed elements

Table 1 shows an obtuse triangle distribution, defined in linear triangular meshes, which are provided by the linear viscous domain mesh generation procedure. O-tri indicates obtuse triangles. Since triangles with large obtuse angles in meshes have a negative effect on numerical solvers, Table 1 indicates that these meshes do not only look good but have a fine quality, which provides a strong support to this high-order viscous mesh generation methodology.

Table 1 Triangle distribution in terms of obtuse angle value

Mesh case name	Total number of triangles	O-tri with $\alpha \in [95^{\circ}, 105^{\circ}]$	O-tri with $\alpha \in [105^{\circ}, 115^{\circ}]$	O-tri with $\alpha \in [115^{\circ}, 125^{\circ}]$	O-tri with $\alpha \in [125^{\circ}, 180^{\circ}]$
30P30N in case 1	21661	1431	57	0	0
30P30N in case 2	31734	1011	105	1	0
30P30N in case 3	28425	745	66	0	0
Bat shape outward	45850	1121	124	3	0
Bat shape inward	31493	614	35	12	0

Table 2 Time cost of the linear domain mesh generation

	Bat inward	Bat outward	Quadratic 30P30N(case1)	Cubic 30P30N(case1)
Time cost (unit: second)	11	13	6	6

Table 2 shows the time cost of four high-order mesh generation cases in this section. These four cases are performed using Intel®Core(TM) i7-3632QM 2.20GHz. The Jacobian-checking process is not counted for the time cost because different methods of Jacobian-checking lead to different time cost. Note that most time cost is in the force balancing process. Hence, this mesh generation method highly relies on force models and their input parameters, such as angle spring and edge spring coefficients. For example, if the non-linear angle spring model is used in the 30P30N cases, the time cost could be tenfold.

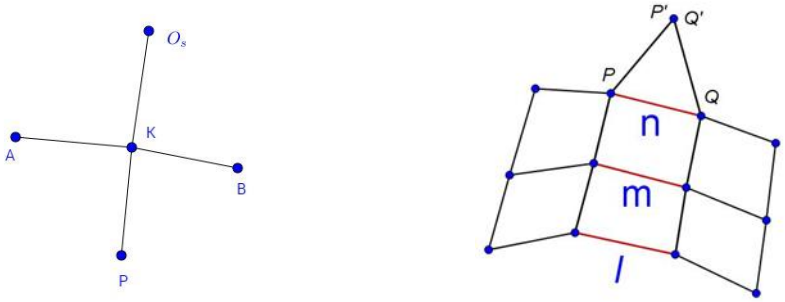
3.2.3 Improved AFSF method and mesh results

To increase the robustness of the AFSF method and to serve the mesh movement and mesh morphing, two major adjustments are made to the original AFSF method, which cause an increase of robustness as expected. They are described in the following section 3.2.3.1.

3.2.3.1 Two main adjustments

One adjustment is changing the advancing-front framework from node-based to edge-based. The advancing-front algorithm in AFSF method is a point based approach as shown in 20 (a). Suppose nodes *A*, *K* and *B* belong to mesh layer i, node *P* belongs to mesh layer i-1 and node O_s belongs to mesh layer i+1. Then node *K* is the next generation of node *P* and node O_s is the next generation of node *K*. Based on this framework, the maps of points to points, points to edges, etc. are given. Different from this framework, in the improved AFSF method, an edge-based advancing-front framework is employed as shown in Fig. 24b). Edge m is the next generation of edge *l* and edge *n* is the next generation of edge m. Since node *P'* and node *Q'* overlapped each

other, there is no offspring of edge *n*. The advantage of this framework is that it can significantly simplify the maps and the implementation of codes. And the "family tree" of the edge procedurally matches the "Pipe" structure used for high-order mesh generation as shown in Figure 25. This match benefits the implementation of high-order mesh generation in programming.



a) b)

Figure 24 Diagrams of the advancing-front method. a) diagram of the point based advancing-front; b) diagram of the edge based advancing-front

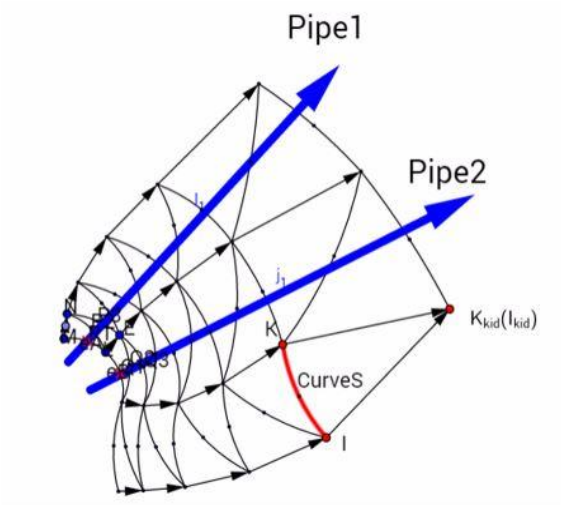


Figure 25 The pipe-like mesh structure

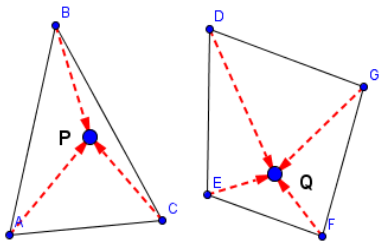


Figure 26 The robustness issue diagram

The other adjustment is adding one more node layer framework as a “super node” layer. In the previous AFSF method, the front-nodes-merging situation faces a robustness issue especially when the mesh extrudes inwards such as shown in Fig. 26. This robustness issue is described as follows. In Fig. 26, node *A*, *B* and *C* all have one child node and their child nodes should overlap each other. So do nodes *D, E, F* and *G*. The previous strategy in the original AFSF method to deal with overlapped nodes from opposite extruding directions is: an edge is generated between these two nodes and the edge state is set to break to force the two nodes to overlap. But in Fig. 26, there are multiple nodes overlapping from different extruding directions, thus the previous strategy cannot suit this case well. To overcome this barrier and to improve the robustness of AFSF method, a new type of node named “super node”, is created. Accordingly, the original mesh node is called the basic node. The basic-super node system has the following rules:

1) Each basic node has a super node attached to it.
2) Each basic node shares the same coordinate value with its super node.
3) One basic node can only map to one super node. One super node can map to multiple basic nodes.
4) If several basic nodes overlap, only one super node is active in that group. This super node will absorb all basic nodes at this overlapped location and the other super nodes are

deactivated.

5) Edges link to basic nodes. So do all the spring models.

Generally, basic nodes are responsible for the interaction with other mesh elements such as edges, quad-like structures and springs. Super nodes are responsible for gathering, broadcasting, and changing mesh values such as node coordinates. The super node does not only act as the representative of a group of overlapped basic nodes, but also as a representative of one single basic node if that is all it contains. Take the cases in Fig. 26 as an example, now node *A*, *B* and *C* just need to be absorbed into one super node (such as node *A*'s super node). Then they will overlap each other, without producing any new nodes, which is a process that works the same for nodes *D, E, F* and *G*. The robustness obtains improvements not only by a better overlapped-nodes strategy but also by a clarifying code structure because the operations on mesh nodes now can be divided into two layers. Note that the 3D case is a different story. In 3D cases, the system of nodes and super nodes increases the difficulty level of implementation of the AFSF method, especially on the mesh topology changing.

After the above adjustments, several mesh-moving cases have been tested. Some results show the AFSF method needs to be further adjusted to benefit mesh movement. These further adjustments are not described here but are described in Chapter 4, which introduces the 2D mesh movement and morphing by the improved AFSF method.

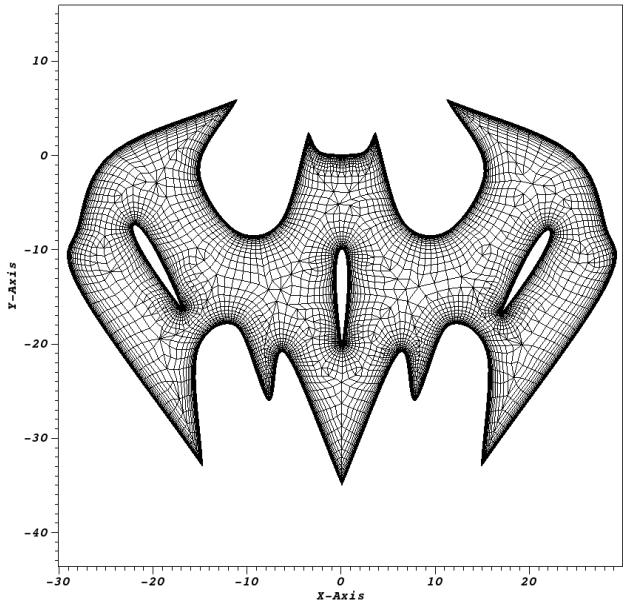
After all these adjustments, several mesh generation cases have been tested. The results are shown in the following section 3.2.3.2.

3.2.3.2 Mesh Results

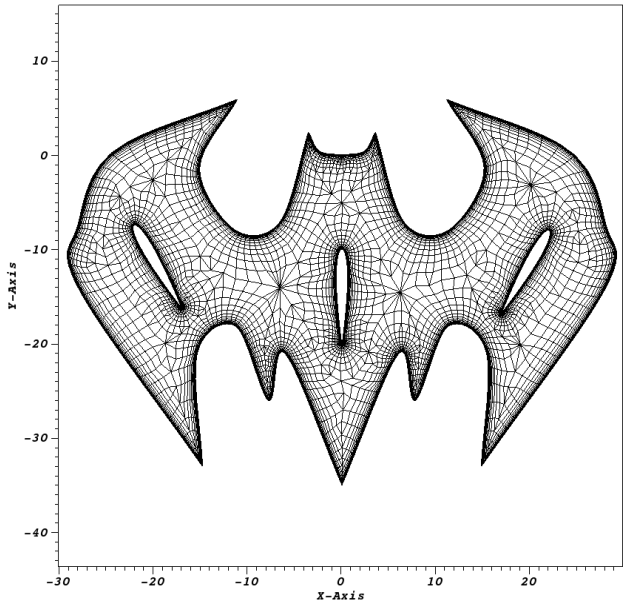
In this section, all the triangular meshes are generated from mixed-element meshes by simply cutting the quads into triangles.

Recall that by the original AFSF method, a bat shape inwards-extruding mesh cannot always be generated due to the lack of robustness, which is shown in the mesh results section in Section 3.2.2. By the improved AFSF method, now this mesh generation approach can not only handle the original bat shape mesh case with variable input factors such as first extruding length and the domain mesh growth ratio, but also can handle more complex geometries as shown in Fig. 27 and Fig. 28.

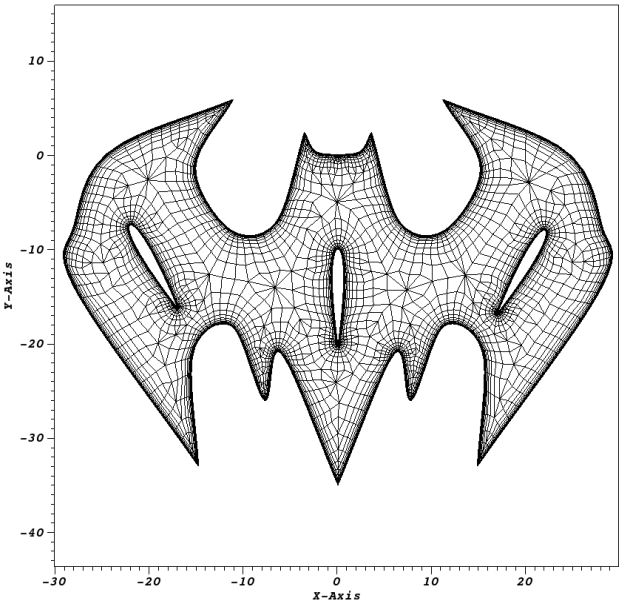
Six cases have been tested. First, three mixed-bat-airfoil cases are tested as shown in Fig. 27. The wingspan of the bat is 60 units. The initial extruding length of the first mesh layer is 1×10^{-4} units in all cases. The boundary mesh and the domain mesh growth ratios are the same. And the values for each case are 1.3, 1.4 and 1.5. The last three are SimCenter-logo cases, where the geometry is obtained from the reference[65]. The initial extruding length of the first mesh layer is 5×10^{-5} units in all cases. The boundary mesh and the domain mesh growth ratios are the same. And the values for each case are 1.3, 1.4 and 1.5. Fig. 28 shows the SimCenter logo case with a growth ratio of 1.3.


10
0
-10
-20
-30
-40
Y-Axis
-30
-20
-10
0
10
20
X-Axis

a)


10
0
-10
-20
-30
-40
Y-Axis
-30
-20
-10
0
10
20
X-Axis

b)



c)

Figure 27 Mixed-bat-airfoil mixed-elements meshes. a) with growth ratio 1.3; b) with growth ratio 1.4; c) with growth ratio 1.5

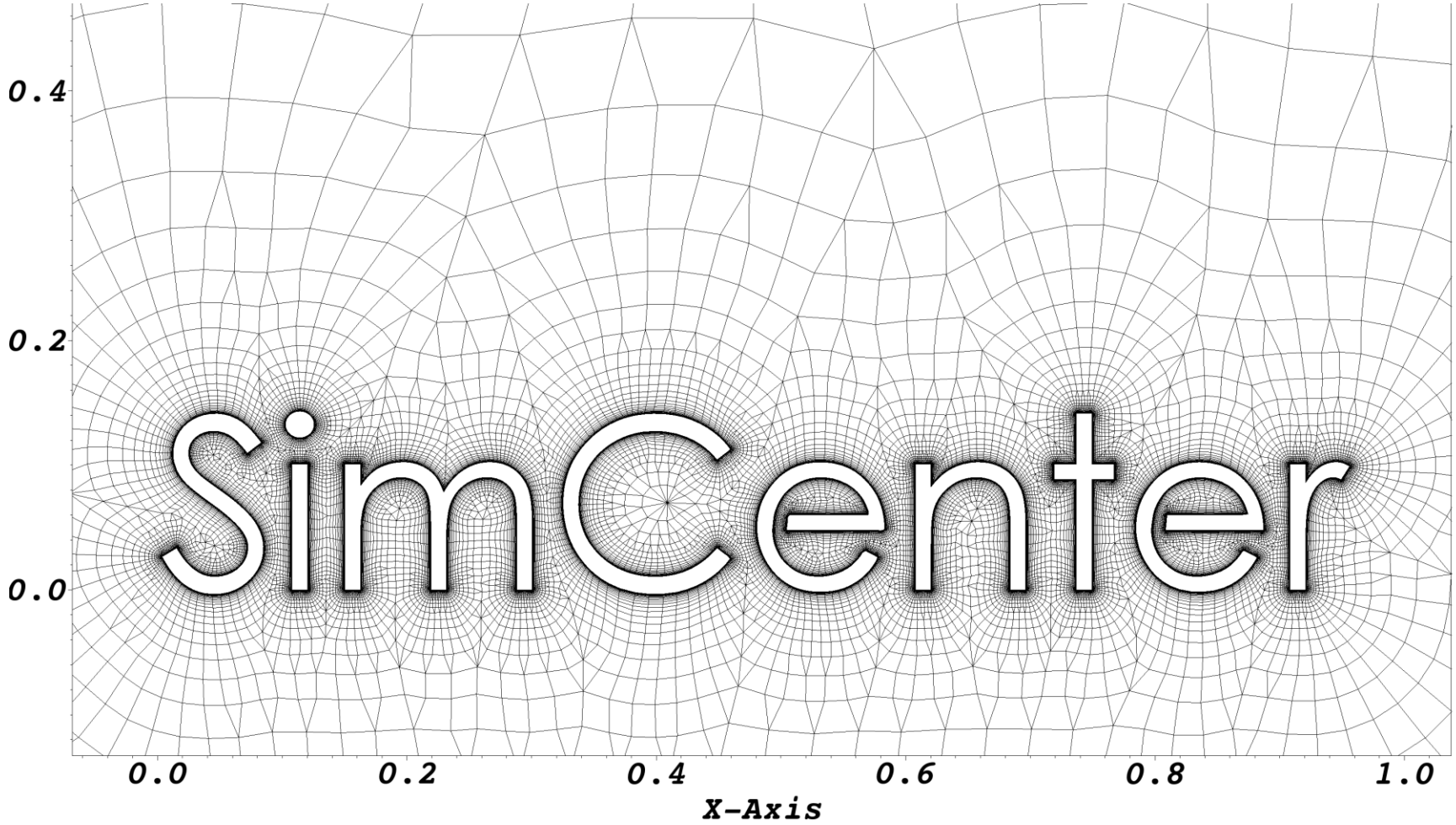


Figure 28 SimCenter logo mixed-elements mesh with Growth ratio 1.3

Table 3 shows the obtuse triangle distribution for variable triangular mesh cases. The triangular meshes are obtained by simply cutting quads of mixed-element meshes into triangles. In all six cases in Table 3, there is no angle bigger than 135 degrees. It indicates that the improved AFSF method preserves the high mesh quality.

Table 3 Obtuse triangles distribution for multiple cases

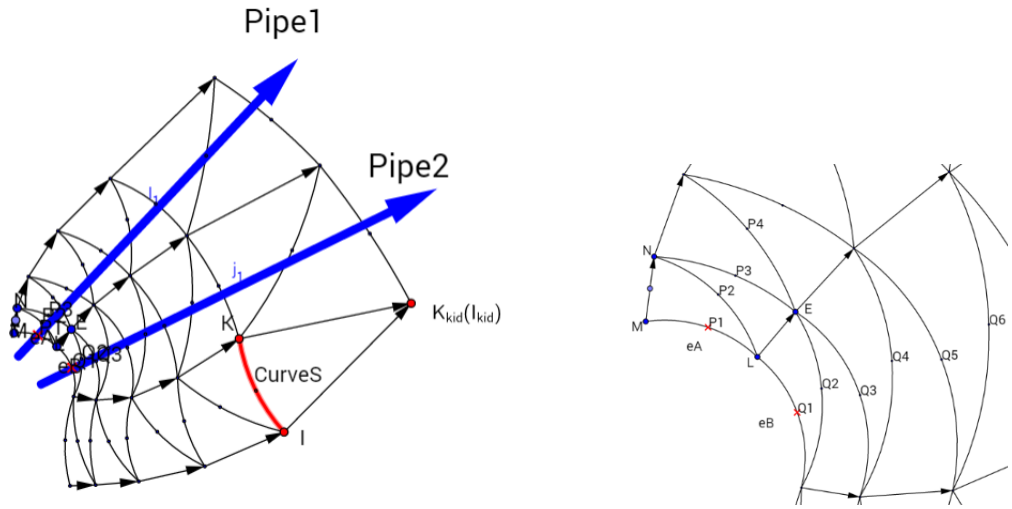
Mesh case name	Total number of triangles	O-tri with $\alpha \in [95^{\circ}, 105^{\circ}]$	O-tri with $\alpha \in (105^{\circ}, 115^{\circ}]$	O-tri with $\alpha \in (115^{\circ}, 125^{\circ}]$	O-tri with $\alpha \in (125^{\circ}, 135^{\circ}]$
Bat-airfoil-1.3	71731	13777	7953	1167	13
Bat-airfoil-1.4	47618	9703	5514	774	26
Bat-airfoil-1.5	35692	7792	3981	576	7
Sim-logo-1.3	89058	17127	334	74	2
Sim-logo-1.4	63013	13410	399	85	12
Sim-logo-1.5	49378	10933	721	119	14

Since the node distribution highly relies on the spring models, different spring model coefficients lead to different obtuse triangles distribution. Generally, the coefficients that lead to a larger angle spring force lead to a better-averaged mesh quality in terms of obtuse angles, but sometimes several triangles with a large obtuse angle (140 to 150 degrees) are generated in the front-edge-merging area. In consideration of an extension to 3D cases, it is valuable to provide a

solution depending on spring models only, though these large angles could be simply fixed by a Delaunay re-triangulation.

3.3 Curved Mesh Deformation Procedure

The curved domain mesh is generated by deforming the previous generated linear domain mesh with Vector-Adding method. Recall that the Vector-Adding method deforms the mesh elements one after another through the "pipe-like" mesh structure with stop conditions, as shown in Fig. 29. There are several stop conditions of the deformation process provided to benefit the numerical solver as described in Chapter 2. An unstop condition is also provided to keep a minimum number of deformed mesh layers to describe the domain close to the geometry in a proper precision.



a) b)

Figure 29 Diagram of curved mesh transformation. a) The pipe-like mesh structure; b) The zig-zag trace of the high-order nodes

3.3.1 Positive Jacobian

Fig. 30 shows how to build a parametric quadratic triangle $\Delta ACA'$ with a high probability of positive-Jacobian quality with Vector-Adding method. First, based on nodes *A*, *B*, and *C*, which are given by the previous described boundary mesh generation method, a quadratic curve $\widehat{ABC}(\ x = f(\xi), y = g(\xi)\ )$ is built. Supposing $Vec1 = \begin{Bmatrix} X_v \\ y_v \end{Bmatrix}$, which is given by the previous described AFSF linear mesh generation method, the algebraic expression of parametric quadratic triangle $\Delta ACA'$ is given as: $\begin{Bmatrix} x = f(\xi) + X_v\eta \\ y = g(\xi) + y_v\eta \end{Bmatrix}$, within a certain range of (ξ, η). By specifying this expression, it is easy to find the position of node B'. Note that in general the position of node B' could be obtained by simply adding the vector $\frac{1}{2}\overrightarrow{AA'}$ on node B without building any parametric curve or mesh element. The position of the node B' obtained in this way is the same as calculating the position of the node B' through the expression of the parametric triangle $\Delta ACA'$. Since the final output mesh provided to the finite element solver includes only the nodes' positions without any algebraic expression information, in practice, there is no need to find the parametric expression of the triangle $\Delta ACA'$.

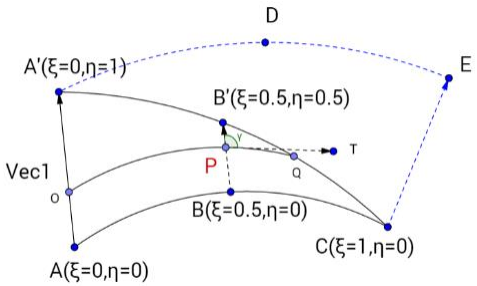


Figure 30 Vector-Adding sketch

For the quadrilateral mesh element, the Vector-Adding method follows the same routine. For example, in Fig. 30, to find the position of node *D* in quadrilateral $ACEA'$ based on boundary curve $\widehat{ABC}$, two non-curved edges AA' and *CE* and the position of node *B* are needed. The position of node *D* = the position of node *B* + $\frac{1}{2}\overrightarrow{AA'}$ + $\frac{1}{2}\overrightarrow{CE}$.

In the following, an explanation is provided on the property of producing the high probability of positive-Jacobian in triangles. Based on the algebraic expression of $\Delta ACA'$, the ξ isolines are linear and parallel to *Vec1*. If an η isoline $\widehat{OPQ}$ is drawn to pass the node *P* and a tangent line *PT* is drawn on node *P* based on the η isoline, the direction of vector $\overrightarrow{PT}$ is $\begin{Bmatrix}\frac{\partial x_p}{\partial \xi}\\ \frac{\partial y_p}{\partial \xi}\end{Bmatrix}$. If the direction of ξ isoline is associated with increasing ξ, the direction of the ξ isoline that passes node *P* is $\begin{Bmatrix}\frac{\partial x_p}{\partial \eta}\\ \frac{\partial y_p}{\partial \eta}\end{Bmatrix}$. If angle γ is the angle between the tangent line of the η isoline and the ξ isoline that pass node *P*, the Jacobian on node *P* can be expressed as $\begin{vmatrix}\frac{\partial x_p}{\partial \xi} & \frac{\partial x_p}{\partial \eta}\\ \frac{\partial y_p}{\partial \xi} & \frac{\partial y_p}{\partial \eta}\end{vmatrix} = \left\|\begin{matrix}\frac{\partial x_p}{\partial \xi}\\ \frac{\partial y_p}{\partial \xi}\end{matrix}\right\|_2 \left\|\begin{matrix}\frac{\partial x_p}{\partial \eta}\\ \frac{\partial y_p}{\partial \eta}\end{matrix}\right\|_2 sin\gamma$.

Hence, if in $\Delta ACA'$ each point has $\gamma \in (0, \pi)$, $\Delta ACA'$ is a positive-Jacobian triangle. That implies if the angle between any tangent line on curve $\widehat{ABC}$ and *Vec1* is larger than 0 and smaller than π, triangle $\Delta ACA'$ is positive-Jacobian. Recall that in the AFSF linear domain mesh generation method, node A' is designed to extrude away from curve $\widehat{ABC}$ almost perpendicularly, and the same property applies to node *E*. Therefore, an assumption is introduced: the angle between the tangent line of curve $\widehat{ABC}$ on node *A* and *Vec1* (or between the tangent line of curve $\widehat{A'B'C}$ on node C and edge CE) is bigger than $\frac{\pi}{4}$ and smaller than $\frac{3\pi}{4}$. If this is true and the statement that the

angle of any two of tangent lines on curve $\widehat{ABC}$(or on $\widehat{A'B'C}$) is smaller than $\frac{\pi}{4}$ is also true, the angle between any tangent line on curve $\widehat{ABC}$ and Vec1(or between any tangent line on curve $\widehat{A'B'C}$ and edge *CE*) is between 0 and π. Hence, $\Delta ACA'$ (or $\Delta EA'C$) is a positive-Jacobian triangle.

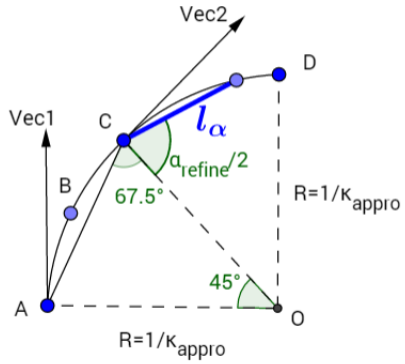


Figure 31 The picked edge in G2

The following deduction is to approach the statement that the angle of any two tangent lines on curve $\widehat{ABC}$ is smaller than $\frac{\pi}{4}$ by reduction to absurdity. The target hypothesis is that, on curved edge $\widehat{ABC}$ there is an angle of two tangent lines equal to or larger than $\frac{\pi}{4}$. All $\widehat{ABC}$s satisfying this hypothesis belong to group *G1*. In *G1*, all $\widehat{ABC}$s that have the maximum smallest curvature of the whole curve $\widehat{ABC}$ are grouped to *G2*. In *G2*, the edge which has the smallest length of $\overline{AC}$ is picked and that type of edge should be like curve $\widehat{ABC}$ shown in Fig. 31. Recall that in Section 3.1, κ_{apro} is an approximated curvature. Hence, another assumption is introduced: κ_{apro} on each geometry node is sufficiently precise and the curvature κ on any position of the curve between two neighboring discretized nodes can be approximated by linear interpolation and is also sufficiently precise. If this second assumption is true, each linear edge should be smaller than the minimum $l_\alpha = f(\alpha_{refine}, \frac{1}{\kappa_{apro}})$ on the same edge. In Fig. 31, the geometry *OABC* is an eighth of

a circle with a radius R=$\frac{1}{\kappa_{apro}}$. Vector vec1 and vec2 are the parts of the tangent line of curve $\widehat{ABC}$ at node *A* and *C* respectively. The angle between them is 45°. Since the edge $\widehat{ACD}$ is a quadratic curve and node *C* is the middle point of this curve, $\angle ACD = \frac{3\pi}{4} = 135°$. Then if the given $\alpha_{refine} > 135° = \angle ACD$, $l_{\alpha\_nodeC} < \overline{CD} = \overline{AC}$. That means edge *AC* should not exist because in Section 3.1 it is not allowed to generate any node spacing on an interval longer than the minimum l_α on the same interval. Hence, the target hypothesis is invalid which means on curved edge $\widehat{ABC}$ there is no angle of two tangent lines equal to or larger than $\frac{\pi}{4}$. With the first assumption, this statement leads to a positive-Jacobian $\Delta ACA'$.

For quadrilaterals, the reason of the highly probable positive-Jacobian is the same. The angle γ between the ξ isoline and the η isoline should be in the range of 0 to π, which is the same reason in the triangle case. For the quadrilateral, it is even safer to claim that any angle γ between ξ and η isolines in the mesh element is in that range. And this two-Vector-Adding strategy can also be used for the triangles. For example, in Fig. 30 with the two-Vector-Adding strategy, $\Delta ACA'$ and $\Delta CEA'$ share the same algebraic expression with quadrilateral $ACEA'$. Still, in consideration of linear computational coordinate transformation, the order of nodes in each mesh element in the output mesh for the solver need to be picked carefully to maintain the positive-Jacobian quality. Compared with the zig-zag trace of the middle nodes *P1, P2, P3…* and *Q1, Q2, Q3…* as shown in Fig. 29b), generated by one-Vector-Adding strategy, this two-Vector-Adding strategy provides a straighter trace.

Even though this Vector-Adding method cannot mathematically guarantee all positive-Jacobian elements without the assumptions, it shows a promising potential to reach this all positive-Jacobian goal by satisfying the two assumptions, with the support of the boundary mesh

generator and the linear domain mesh generator.

Note in general, a finite element CFD solver only takes the coordinates of the high-order nodes as input instead of the algebraic expressions of the high-order mesh elements. If the solver wants to maintain the positive-Jacobian quality, two requirements for the solver should be satisfied. The first is that the number of mesh nodes used for the CFD solver to rebuild each parametric element should be equal to or larger than the number of coefficients of high-order expression of the same mesh element. The second is that by a simple linear coordinate transformation, the computational coordinate $(\xi_{mesh}, \eta_{mesh})$ used in the mesh generation process can be transferred to the coordinate $(\xi_{solver}, \eta_{solver})$ used in the solver for each matched node in each mesh element. If both requirements are satisfied, regardless of the method of CFD finite element solver used to rebuild a parametric mesh element, the algebraic expression should be the same as the one in the curved mesh generation part and hence the positive-Jacobian quality is maintained.

3.3.2 Mesh results

This section shows two examples with a bat shape geometry and 30P30N airfoil geometry. In this section, all the final curved meshes are deformed from the linear domain meshes generated by the original AFSF method. These linear meshes have been shown in Section 3.2.1. The curved meshes generated based on the linear meshes, which are generated by the improved AFSF method, will be introduced in Chapter 5.

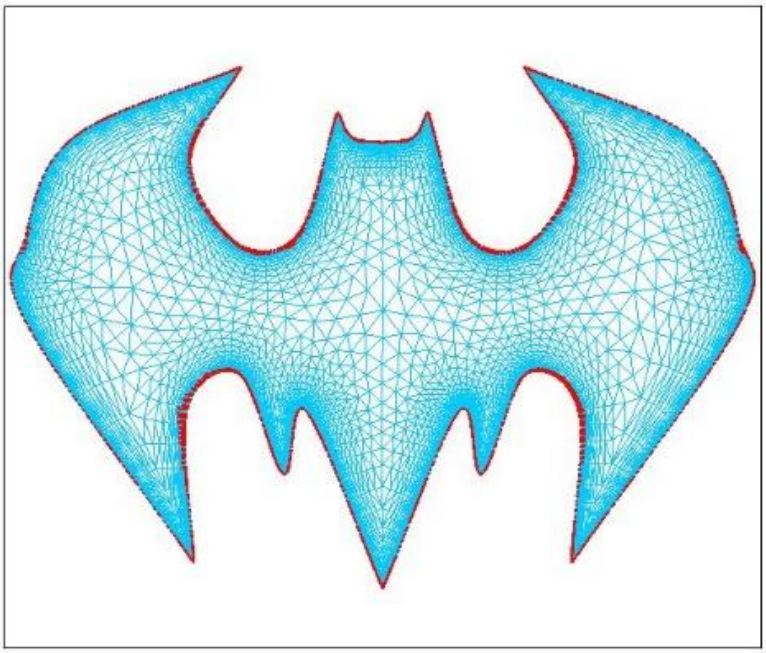


Figure 32 The bat shape inward extruding curved mesh with pure elements

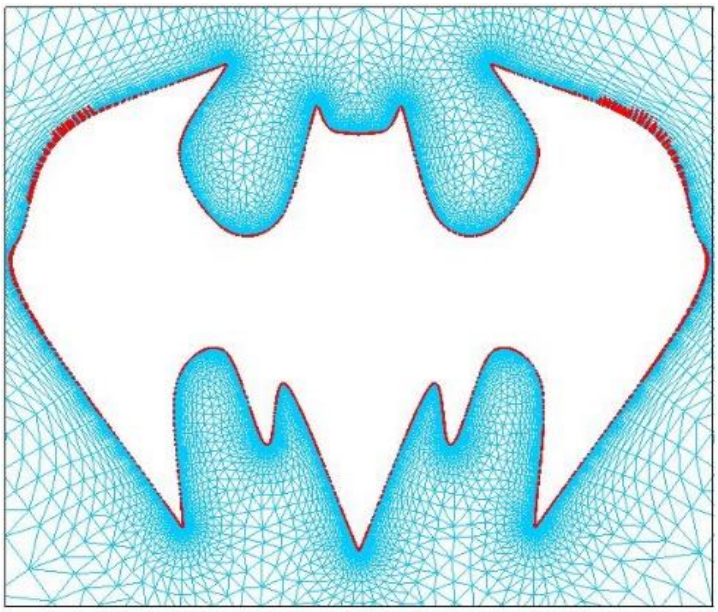
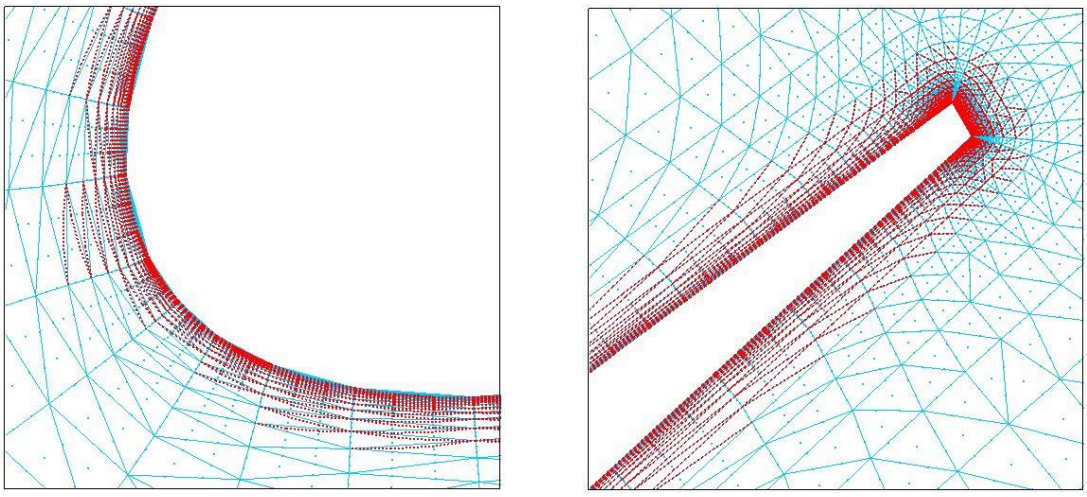


Figure 33 The bat shape outward extruding curved mesh with pure elements

First, let us look at the bat shape example. Two cases are tested with inward and outward extruding directions, respectively. The representations of parametric curves on the boundaries and

in the domain are both quadratic polynomials. Good-looking curved domain meshes are obtained as shown in Fig. 32 and Fig. 33. In Fig. 32, after deformation, the percentage of curved edges is 40%, as shown in red dotted lines. In Fig. 33, after deformation, the percentage of curved edges is 31%, as shown in red dotted lines. In Fig. 32 and Fig. 33, all mesh elements are positive-Jacobian, which indicates that the deformation approach works well.

Second, let us look at the 30P30N example. Three cases are tested with both piecewise quadratic and cubic polynomials to represent the mesh boundaries. These three cases use different linear domain meshes which are generated in Section 3.2.1. All the figures in this section are from case 1. All triangular meshes in the three cases show all positive-Jacobian triangles. Note that the final cubic viscous mesh in case 3 is tested with the finite element CFD solver and the results are shown in Section 3.3.3.



a) b)

Figure 34 Close view of the cubic curved mesh and the linear mesh, a) leading edge of the slat; b) trailing edge of the slat

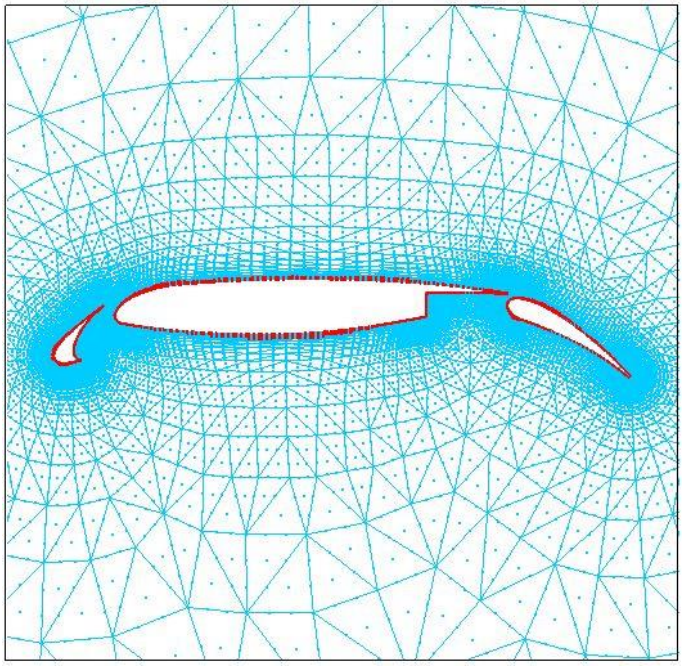


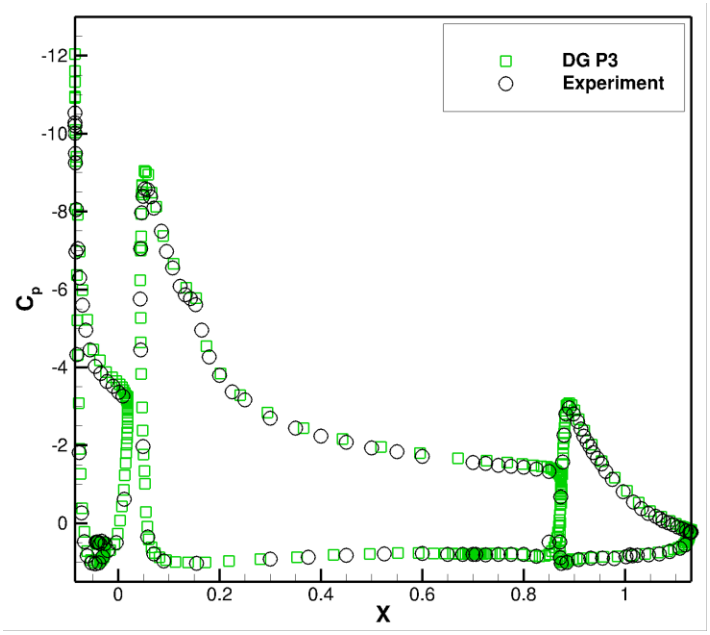
Figure 35 Distant view of cubic deformation of 30P30N airfoil

Fig. 34 presents a comparison of the linear mesh before deformation and the cubic curved mesh after deformation. The red dotted lines are cubic curved edges while the blue solid lines are non-curved edges. It illustrates that both the Vector-Adding method and the stop conditions of deformation work well. Fig. 35 presents the entire cubic deformation of the linear mesh. In Figs. 30 and 31, all the blue solid edges, which do not have matched red dotted edges, are the remaining non-curved edges in the final high-order viscous mesh. In this cubic case, the percentage of deformed edges to total edges is 36%. It illustrates that the 64% remaining non-curved edges in the final curved mesh could benefit the numerical solvers.

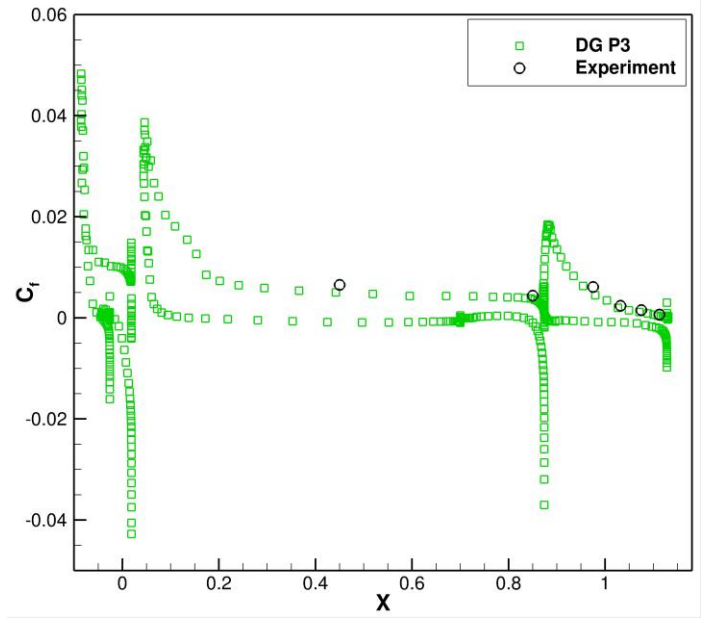
The time cost of Vector-Adding deformation without the Jacobian-checking part for all the five cases are all less than 1 second, which is negligible compared with the time cost of the matched linear domain mesh generation by the original AFSF method.

3.3.3 CFD Results

The following CFD Result is obtained based on the curved high-order mesh in case 3 from the above curved meshes results section.



a)



b)

Figure 36 The C_p and C_f distributions of the 30P30N airfoil. a) C_p distribution; b) C_f distribution

The final cubic triangular mesh in case 3 is used. The total numbers of mesh nodes and triangular elements are 128746 and 28425, respectively. The mesh is tested using a high-order discontinuous Galerkin finite element solver that solves the Reynolds Averaged Navier-Stokes equations coupled with the modified one-equation Spalart-Allmaras turbulence model. Details about the discretization method can be referred to in Ref. [20]. In the test case, the freestream Mach number is 0.2, the angle of attack is 16.2 degrees and Reynold number is 9 million. Fig. 36 shows comparisons of the surface pressure and skin friction distributions using a fourth-order DG discretization with experimental data. Between the CFD result and the experimental data, the average absolute difference of C_p is 0.13, and the maximum absolute difference of C_p is 1.09. The average absolute difference of C_f is 0.0008, and the maximum absolute difference of C_f is 0.0014. It is depicted that the present numerical solutions agree well with the experimental data. Fig. 37 shows the Mach number contours around the multi-element airfoil. It is observed that the slat wake that persists all the way over the flap is well captured using the current mesh, and in addition, the solution in the flap wake region is also quite smooth. Streamlines around the multi-element airfoil configuration are displayed in Fig. 38, with the close-up views on the slat and the flap as well as the flap cove on the main element. As seen in these figures, the streamlines show high curvatures near the gap between the slat and the main element. Recirculation zones near the slat lower surface and the flap cove are also clearly shown.

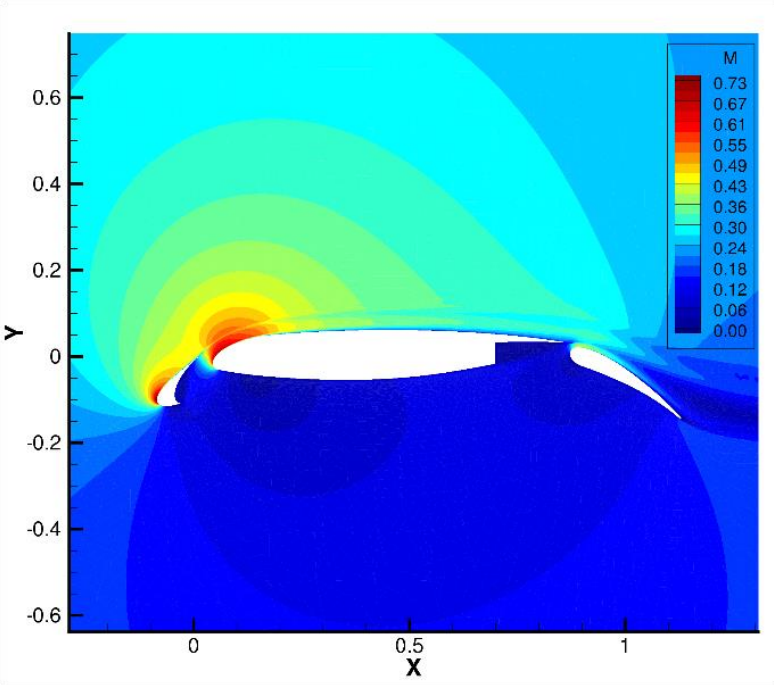
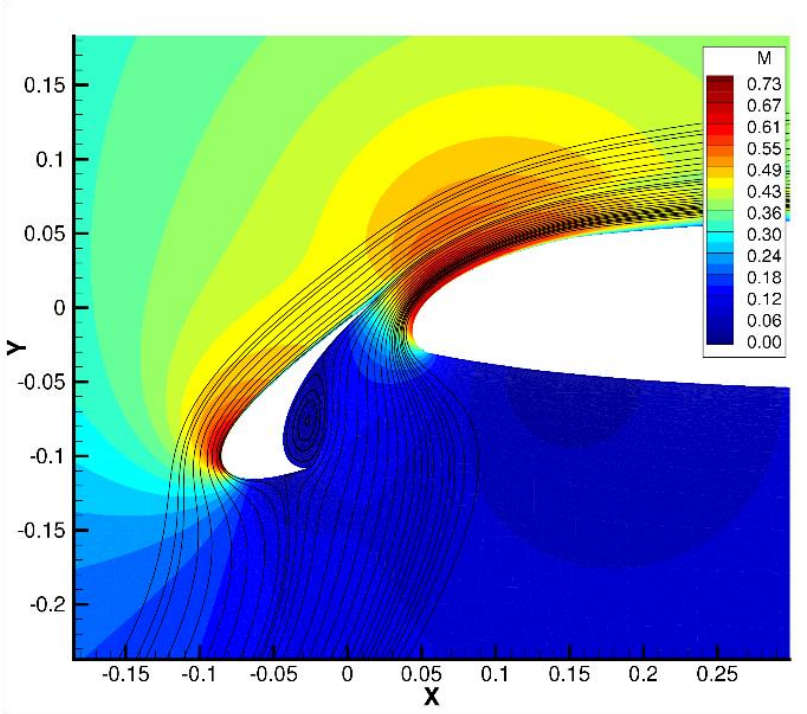
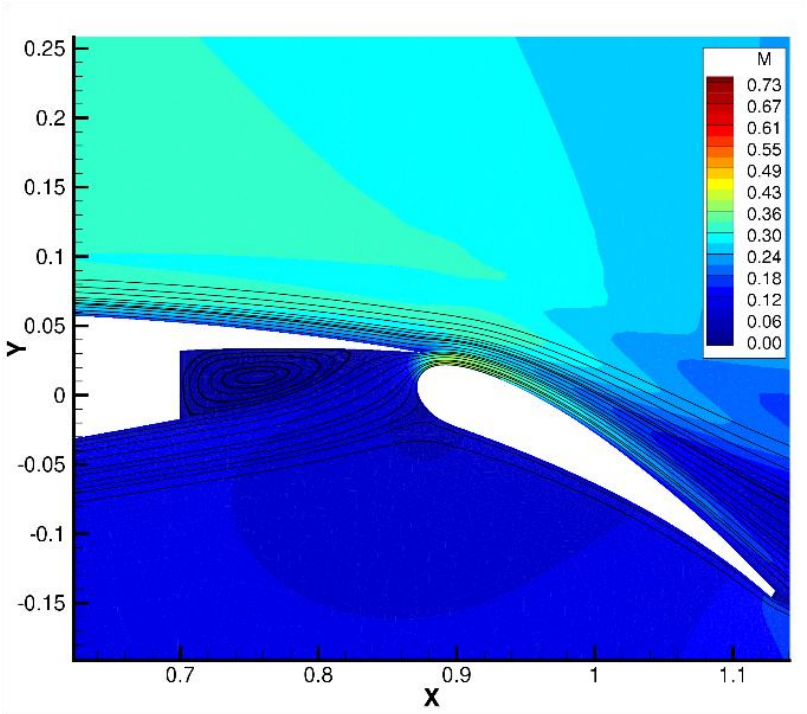


Figure 37 Mach number contours around the 30P30N airfoil

The numerical results indicate that the mesh generated by the Spring-Field force model and Vector-Adding deformation method is promising for a higher-order scheme to achieve well-resolved flow fields for arbitrarily complex geometries.



a)



b)

Figure 38 Streamlines around the 30P30N airfoil. a) The slat zone; b) The flap zone

CHAPTER 4

2D MESH MOVEMENT

As a spring model, the "Spring-Field" used in the AFSF method is readily extensible to mesh movement or morphing. In this Chapter, the improved AFSF method used in the application of mesh movement and mesh morphing is introduced. In Chapter 3, the two main adjustments for the improved AFSF method are introduced. This Chapter will focus on more adjustments. These further adjustments are made for the mesh movement and mesh morphing cases. The main adjustments focus on the spring models. After introducing these adjustments, several cases are tested: mesh movement cases and mesh morphing cases with large deformations. Finally, the mesh results are tested as a part of airfoil shape optimization with a finite element high-order solver.

4.1 Further Adjustments in the Improved AFSF Method

Except for the two main adjustments described in Section 3.2.3, further adjustments are made for mesh movement purpose.

First, the way to adjust the mass of the nodes in spring models changes. To solve the oscillation issue which a typical force model generally faces, the mass of the node is added into the spring force model. To approach the force balance, the mass of an oscillating node increases. If force balance is not achieved, ultimately the enormous mass will annihilate the oscillation numerically. This model performs well in mesh generation because the force balance position of a

node will always be in the range of the oscillation. So, by decreasing the oscillation amplitude, the node can attain its balancing position. But for mesh movement cases, especially the cases with large boundary motion, the balancing position is not always in that oscillation range. In this case, should the original strategy be taken, the node could finally end in a position far away from the force balancing position. To solve this problem, we invoke the following strategy: When the spring force drags the node to its ideal position, the mass will first be constant. Then the mass starts to increase after a certain number of iterations n, if there is an oscillation. This strategy will work if we assume the node is close to its ideal position after the first certain numbers of iterations n.

Second, an adjustment is made to the relationship between the angle spring model and the edge spring models. In a big motion case, the angle spring may disturb the node movement if the node is too far from the target position where it should end. However, in the improved AFSF method, the angle spring force is blocked until the node moves close to its target position, which is judged by the ratio of the ideal length of the extrusion edge (linked to this node) to its real length. We also separate the extrusion edge springs from the neighboring edge springs and angle springs. Though the two groups share the same time step, now they have their own characteristic movement distance and maximum movement distance limitation, by which the spring models decrease decoupling and the cooperation becomes more manageable.

Third, an adjustment is made to the extrusion edge spring model to accelerate the convergence process, and to accommodate the compressed mesh cases. The extruding spring force, which is the main force to drag the node to its target location in large boundary motion cases, is adjusted as shown in the force equation Eq. 1

$$F_{extru\_eg} = C_{aniso} \times K_{extru\_eg} \times \frac{l_{ideal}-l_{real}}{l_{ideal}} \quad (1)$$

Where l_{ideal} is used instead of $l_{ideal} + l_{real}$, which is used in the original AFSF method, to amplify the force value difference generated by different l_{real}. An anisotropic coefficient C_{aniso} is introduced into the equation. In one edge spring, C_{aniso} is different for the two nodes attached to this spring to provide a proportional compression of edges. In the mesh results section, for the node in lower mesh layer, $C_{aniso} = \frac{1}{G_{ratio}}$. For the node in upper mesh layer, $C_{aniso} = 1$.

Fourth, the neighboring edge spring model changes. adjustment to the neighboring edge spring is that the neighboring edge spring strengthens if the aspect ratio of the attached mesh element is smaller than a certain given global factor. Take Fig. 39a) as an example, if $\frac{l_{real}\ of\ AK + l_{real}\ of\ KB}{l_{real}\ of\ KP + l_{real}\ of\ KP}$ is smaller than a given constant global factor, the neighboring edge spring force updates to

$$\mathrm{F_{nei\_eg}} = \mathrm{Fo_{nei\_eg}} \times \frac{\mathrm{l_{real}\ of\ AK + l_{real}\ of\ KB}}{\mathrm{l_{real}\ of\ KP + l_{real}\ of\ KP}} \times \mathrm{Fac_A} \qquad (2)$$

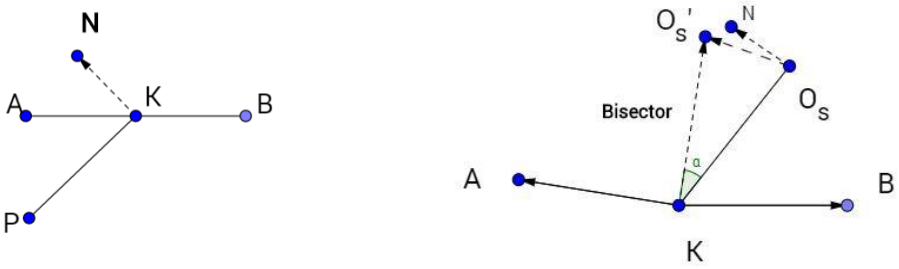
where Fac_A is a given constant global coefficient and $Fo_{edge\_neighbor}$ is the original force of neighboring edge spring obtained by the original force formula. The aim of this adjustment is to obtain a smoother interior mesh in the mesh movement cases with large motions under the larger neighboring edge spring force.

Last, the change is made to the angle spring model. The angle spring force function is changed to

$$F_{ang} = K_{ang} \times f(Angle_{advanced}) \qquad (3)$$

where $f(Angle_{advanced})$ is a polynomial function of $Angle_{advanced}$, with the same sign as α_{appro} in Section 3.1. $Angle_{advanced}$ is the sum of absolute value of α_{appro} and Fac_{shape}. Take

Fig. 39b) as example. Fac_{shape} is defined as $\mathrm{Fac_{shape}} = \min(\mathrm{Fac_B}, \frac{\mathrm{l_{real\ of\ AK}}+\mathrm{l_{real\ of\ KB}}}{\mathrm{l_{real\ of\ KO_s}}+\mathrm{l_{real\ of\ KO_s}}})$ as same as the one in the original AFSF method. The change is also made to the direction of force in the angle spring model. For the previous angle spring, if the mesh element aspect ratio is extremely large, the angle spring force attached to that mesh element will be extremely large too. And the force on node O_s has a perpendicular direction $\overrightarrow{\mathrm{O_sN}}$ to the extrusion edge KO_s as shown in Fig. 39b). By the influence of this force direction and other factors, especially the constant mass in the first several iterations, the large force possibly leads node O_s to do a zig-zag motion as shown in Fig. 40. And that motion leads to an unwanted lengthy edge KO_s. One way to avoid this is to decrease the local max moving distance limitation of angle spring, but that will cause a slower convergence. Another way is to provide a better edge-angle spring models system, which could be an area of future research. The current solution for this problem is, for the angle spring such as *A-K-B-*O_s, the force direction on node O_s changes to $\overrightarrow{\mathrm{O_sO_s'}}$, where O_s' is on the bisector and $\overline{\mathrm{KO_s}} = \overline{\mathrm{KO_s'}}$. And the force direction on node *K* changes to either $\overrightarrow{\mathrm{KA}}$ or $\overrightarrow{\mathrm{KB}}$, which depends on the sign of angle α. This strategy shrinks the average extra relative length of extrusion edges to less than 5% in the viscous mesh layer. Note that, if the absolute value of angle α is bigger than $90°$, which indicates the mesh element attached to it has a bad quality or even a negative-area, the force direction will still be obtained by the previous edge-norm strategy to avoid node O_s ending below the line *A-K-B* in an improper force-balancing position.



a) b)

Figure 39 The spring model diagram. a) the neighboring edge spring; b) the angle spring

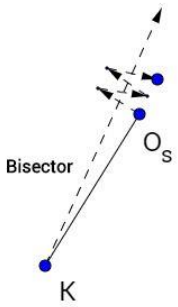


Figure 40 The zig-zag motion diagram

An extra benefit of adjusting the AFSF method for mesh movement is that the method can still cover mesh generations. That indicates that by this method, the mesh could be generated first, then under the same mesh topology, the mesh could be moved.

4.2 Mesh Results

For high-order mesh generation, the curved geometries and their piecewise functions are required. They are obtained based on the method described in Section 3.1. This special-structure linear mesh is obtained by the improved AFSF method mentioned in Section 3.2. The high-order

meshes are obtained by Vector-Adding method described in Section 3.3. Note the interpolation points for curved domain meshes are not moved by spring force or another approach. They are regenerated layer-by-layer based on boundary interpolation points through the pipe-like linear mesh structure. Since this process is managed by the Vector-Adding method, which costs much less time compared with the linear mesh movement cost, the main computational costs for this high-order mesh movement approach would be the linear mesh movement costs. That time cost strongly relies on the spring force model and the damping strategy. Adjusting spring models and damping strategy for time efficiency are not discussed in this research. In this section, all the triangular meshes are generated from mixed-element meshes by simply cutting the quads into triangles. All the meshes, either original or after movement, have positive Jacobian mesh cells with cubic curves (P3), which is enforced by a harsh Jacobian-checking process with 900 checking points in each triangle.

Recall that, in Section 3.2, the robustness of the linear mesh generation is improved by the improved AFSF method, which enables some linear mesh with complex geometries be generated. Now let us look at how the improved AFSF method performs with Vector-Adding method on mesh movement and mesh morphing. Benefited from the improved AFSF method, there is no need for intermediate moving steps for a relatively large movement distance. All the mesh movement and morphing cases are moved in one step in this section, and only the final shapes or destinations of the boundaries are needed.

The first group of test cases is motions combined with boundary translations and rotations, as shown in Fig. 41. The test geometry is NACA 30P30N airfoil, with an initial extruding length 5×10^{-5} and a domain mesh growth ratio 1.3. The initial airfoil is located around the origin. The first test case is shown in Fig. 41. (a) and (c), where the boundaries rotate 90 degrees with a rotation

center at (0,0), then move with a vector of <-10,10>. The second test case is shown in Fig. 41. (b) and (d), where the boundaries rotate negative 90 degrees with a rotation center at (0,0), then move with a vector of <10, -10>.

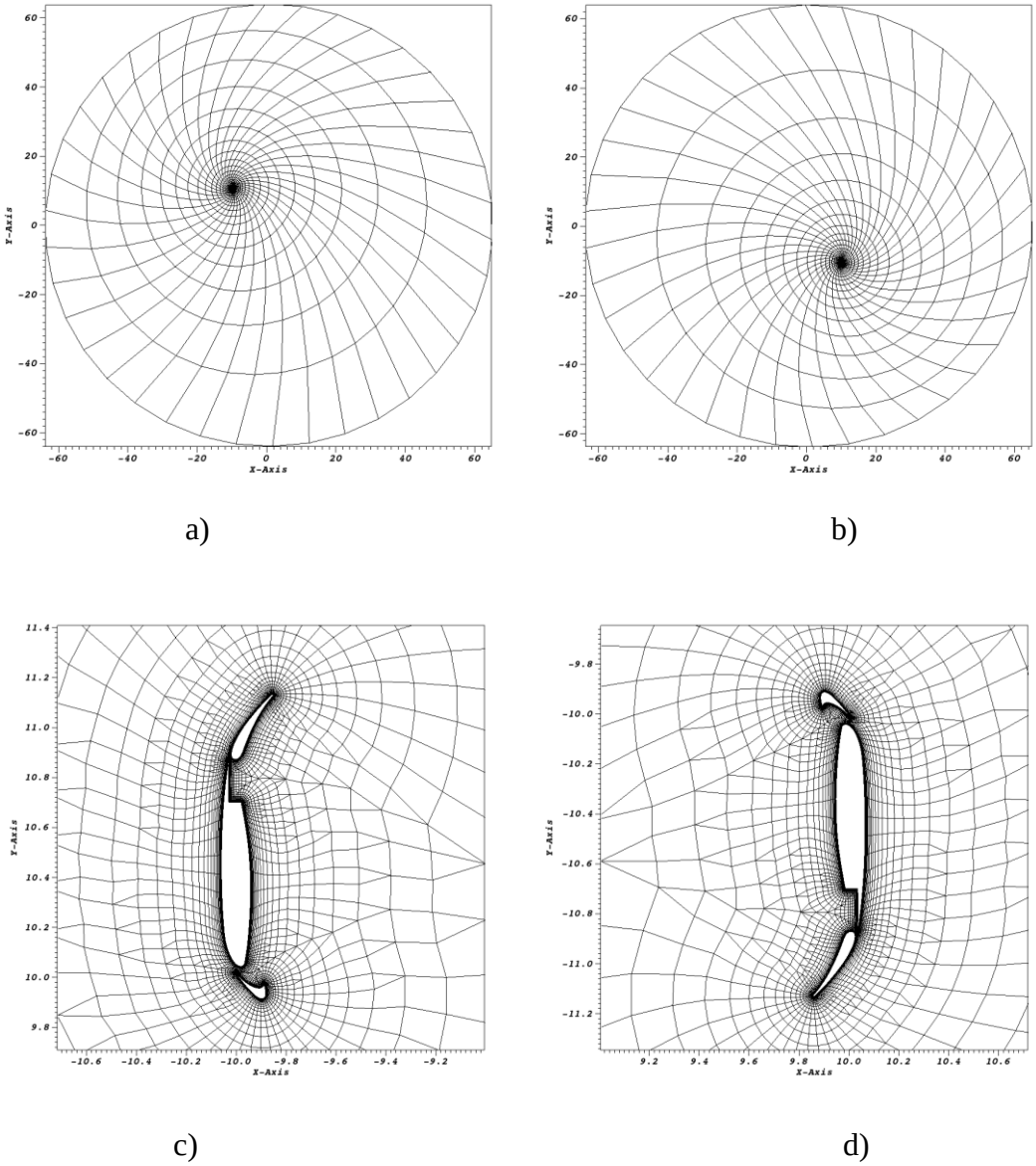
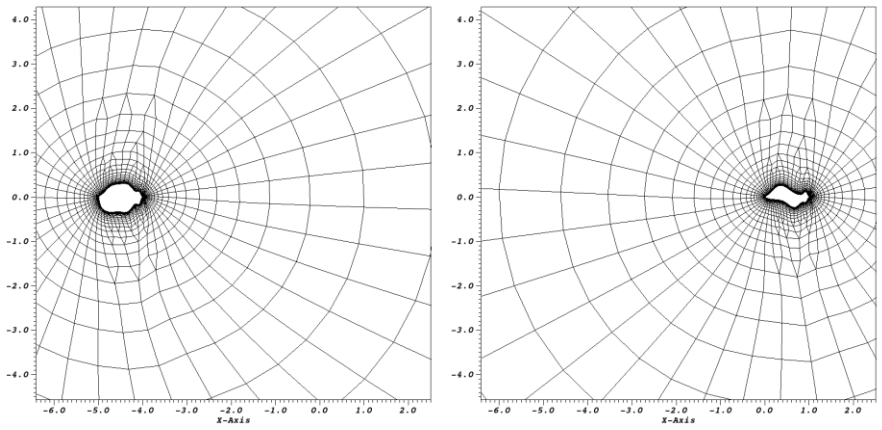
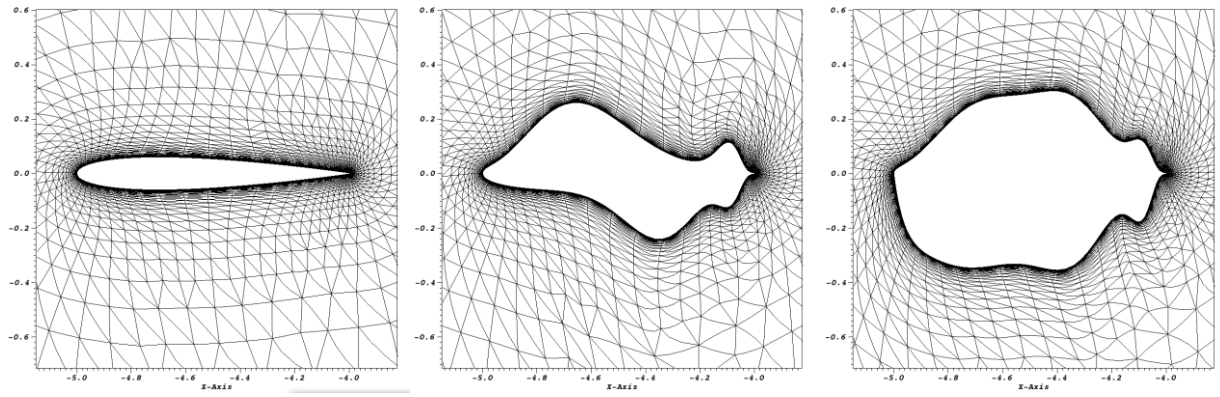


Figure 41 30P30N translation and rotation cases. a) Case 1 whole view in linear mixed-elements mesh; b) Case 2 whole view in linear mixed-elements mesh; c) Case 1 close view in linear mixed-elements mesh; d) Case2 close view in linear mixed-elements mesh

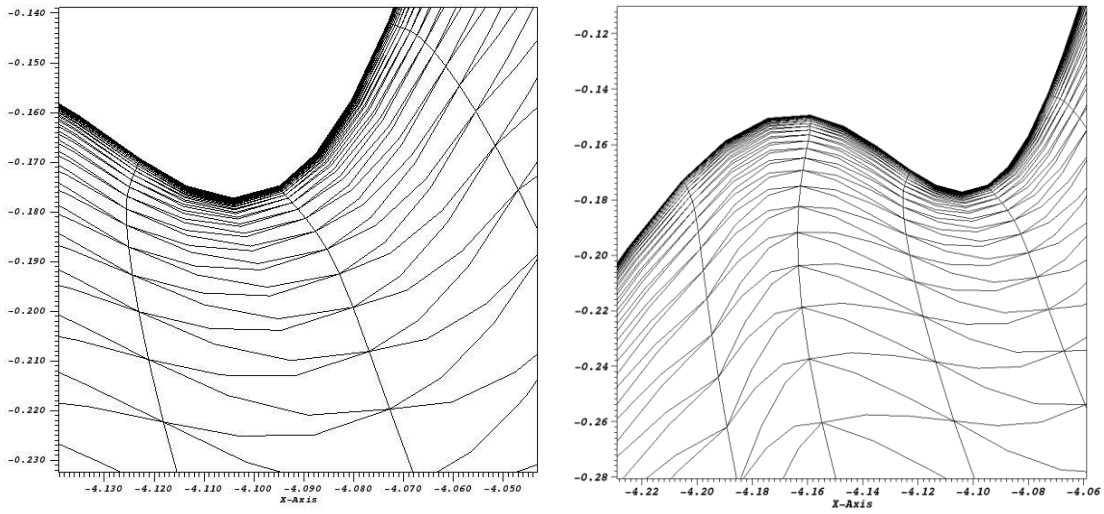
The second group of test cases is boundary morphing. The test geometry is NACA0012 with the same initial extruding length and growth ratio as 30P30N cases. The mesh boundary morphs with the Hicks-Henne bump functions[66]. This airfoil first changes to a larger size. Then it moves to the left with a five-time-body-length distance as it becomes larger, as shown in Fig. 42. The Fig. 42c) shows that, with such a relatively large mesh morphing, the boundary layer mesh still keeps the perpendicular feature to the airfoil boundary, which attributes to the angle springs.



a)



b)



c)

Figure 42 NACA0012 moving and changing shape cases. a) NACA0012 positions in linear mixed-elements meshes; b) Three size changing phases of NACA0012 in curved triangular meshes. From left to right, original, phase 1 and phase 2; c) A close view of two convex and concave areas of NACA0012 after morphing

The mesh quality differences between original meshes and meshes after movements are shown in Table 4, in terms of obtuse triangles distribution. For all cases, there is no triangle with an angle bigger than 150 degrees.

Table 4 Obtuse triangles distribution for mesh movement cases

Grid case name	Total number of triangles	O-tri with $\alpha \in [95^{\circ}, 105^{\circ}]$	O-tri with $\alpha \in [105^{\circ}, 115^{\circ}]$	O-tri with $\alpha \in [115^{\circ}, 125^{\circ}]$	O-tri with $\alpha \in [125^{\circ}, 135^{\circ}]$	O-tri with $\alpha \in [135^{\circ}, 150^{\circ}]$
30P30N original	19139	4556	2488	352	0	0
30P30N case1	19139	3939	2681	1277	6	0
30P30N case2	19139	4570	3244	1649	59	0

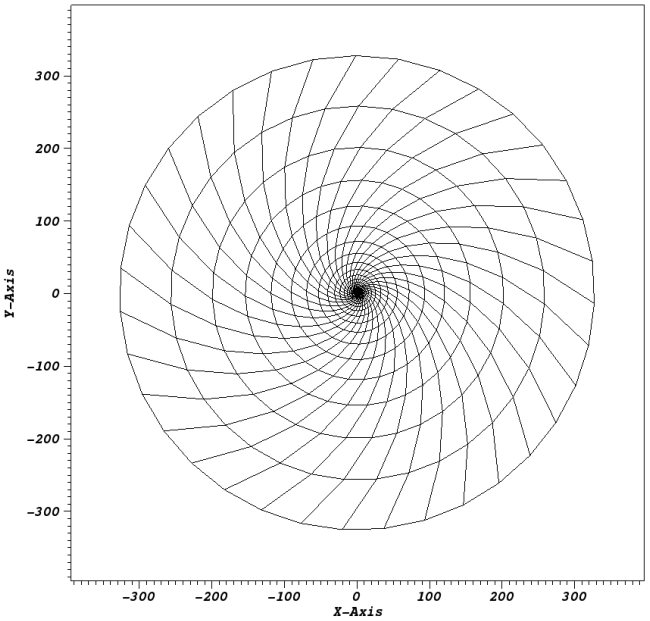
NACA0012 original	6459	1400	686	140	0	0
NACA0012 phase1	6459	1509	885	625	160	33
NACA0012 phase2	6459	1561	1004	655	237	24

These figures and the table indicate that the mesh movement method has promising performance. It is worth noting that, the mesh nodes can either move together or move layer by layer (not restricted to one layer at a time). For some more complex cases, such as a combined motion of 180-degree rotation and a large translation, the latter strategy is more effective than the former one in avoiding mesh movement failing. Whatever the case is, the latter strategy can always add several iterations of moving all nodes together to smooth the whole mesh after the layer-by-layer movement.

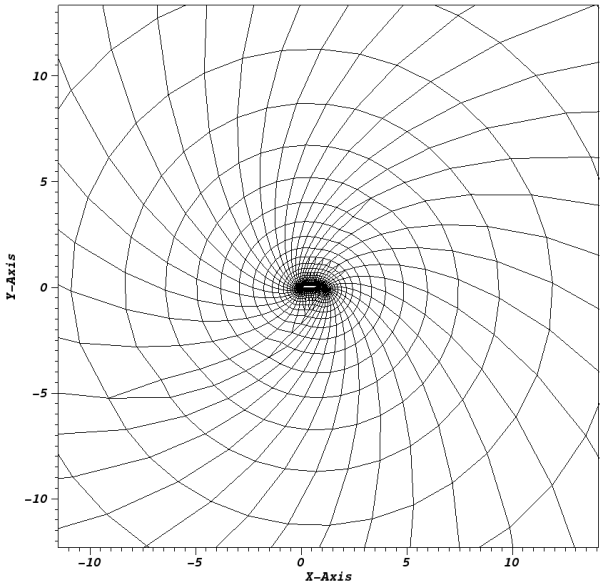
4.3 CFD Results

A cubic 30P30N triangular curved mesh is used as shown in Fig.43e). It is obtained by three steps. First, rotate the original airfoil boundary mesh with 180 degrees; Second, generate the domain mesh (now the airfoil is upside down); Third, rotate the boundary mesh with 180 degrees and move the domain mesh, while keeping the nodes on the far-field boundary standing still. This triangular mesh is obtained from the mixed-element mesh as shown in Fig.43a)-d). From Fig.43 (d) we can tell that after the 180-degree rotation, the mesh around the flap zone keeps almost the

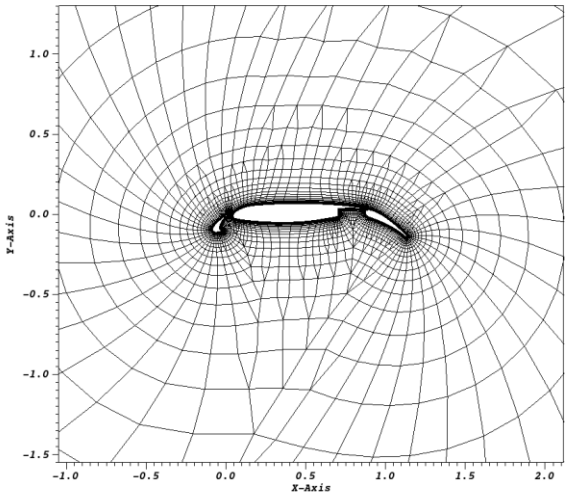
same shape as the mesh before the rotation. That attributes to the improved AFSF method, especially the spring models in the improved Spring-Field. The total numbers of mesh nodes (not including high-order nodes) and triangular elements are 16330 and 30937, respectively.



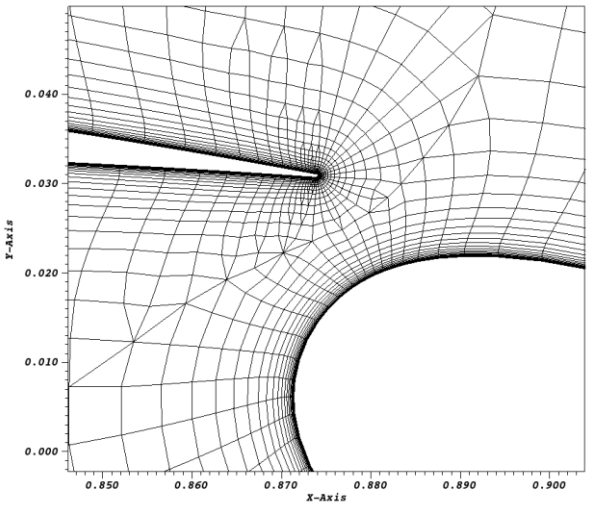
a)



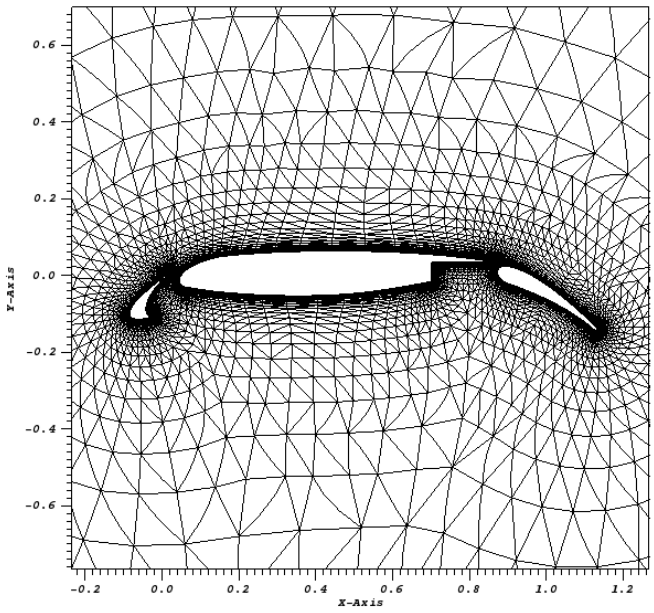
b)


1.0
0.5
0.0
-0.5
-1.0
-1.5
Y-Axis
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
X-Axis

c)


0.040
0.030
0.020
0.010
0.000
Y-Axis
0.850
0.860
0.870
0.880
0.890
0.900
X-Axis

d)



e)

Figure 43 The 30P30N airfoil curved mesh with 180-degree rotation: a) the mixed-element linear mesh in the whole view; b) the mixed-element linear mesh in a close view; c) the mixed-element linear mesh in a closer view; d) the flap zone in the mixed-element linear mesh; e) triangular curved mesh in a closer view

The grid is tested by using a high-order stabilized finite element method to solve for the Reynolds Averaged Navier-Stokes equations coupled with the Spalart-Allmaras turbulence model. Greater details concerning the methodology may be found in Ref. [65]. Fig. 44 shows a comparison of the *Cp* distribution using a fourth-order spatial discretization with experimental data. It is shown that the present numerical solution is in good agreement with the experimental data, which implies a good mesh quality under such large rotating degrees.

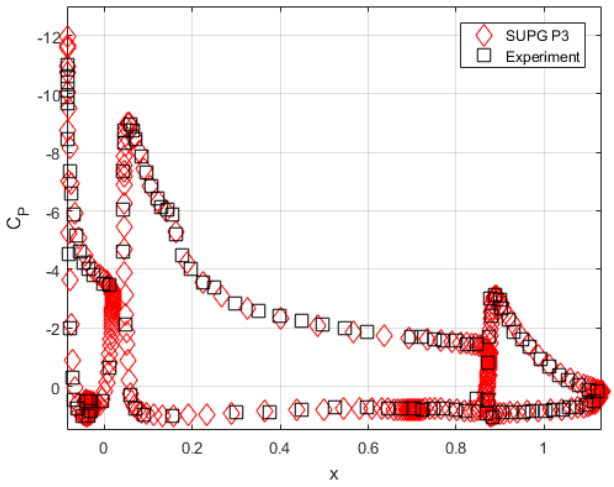


Figure 44 The *Cp* distribution of the 30P30N airfoil

4.4 Optimization

In this section, gradient-based optimizations of the NACA0012 airfoil are conducted. High order fully unstructured meshes are used for the viscous computations of turbulent flows. A stabilized finite element formulation (Streamline Upwind Petrov Galerkin, or SUPG) is used to discretize the compressible Navier-Stokes equations[67] with linear elements, and the Spalart-Allmaras turbulent model is employed. The sensitivity derivatives are calculated using the central finite-difference method. For optimizations with many design variables, adjoint-based sensitivity analysis can be feasible for both steady[68] and unsteady[69] problems.

The baseline case is a computation of the flow over the NACA0012 airfoil at a free-stream Mach number of 0.4, an angle of attack of 2°, and a Reynolds number of 5 million. The grid used for this computation consists of approximately 4,000 points. The spacing at the wall is 5e-5 of the

chord length. The airfoil is parameterized with the Hicks-Henne bump functions[66]. Two sample optimization cases are tested below.

Case 1 attempts to obtain a specified pressure distribution with 8 control points

$$I = \frac{1}{2}\int_{\Gamma}\left(p - p^{*}\right)^{2} d\Gamma$$

Here p is the current pressure distribution, and p* is the pressure distribution corresponding to the target geometry. After 15 design cycles, the cost function is reduced from 1.5e-02 to 4.3e-12. It is found that the target pressure distribution is obtained along with the recovery of the objective geometry.

Case 2 aims to decrease the drag-lift ratio (or equivalently increase the lift-drag ratio), and the cost function is given by

$$I = \frac{C_D}{C_L}$$

Here C_D is the drag coefficient and C_L represents the lift coefficient. The number of design variables is 16. After 15 design cycles, the drag has been reduced from 0.005 to 0.004, and the lift has been increased from 0.224 to 0.409. Fig. 45. shows the pressure distributions before and after the optimization.

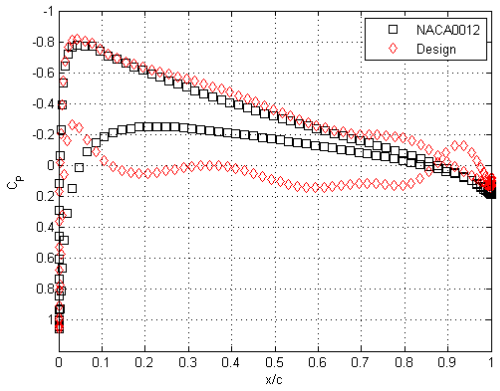


Figure 45 Initial and final pressure distributions

Fig. 46. shows the computational mesh obtained by linear elasticity as a comparison to the current approach. It is obvious that there is a mesh quality issue in viscous layers. In contrast, the mesh movement with current approach produces quality viscous computational mesh, as seen in Fig. 47.

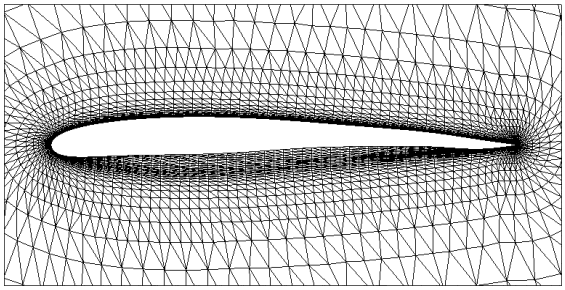


Figure 46 Modified NACA0012 airfoil mesh with linear elasticity

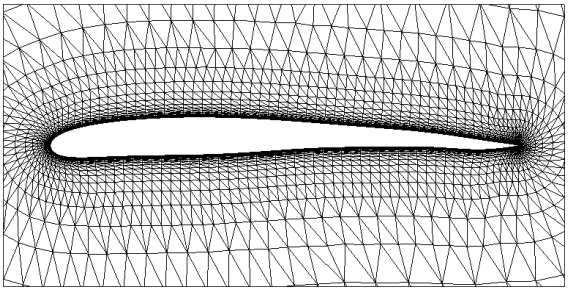


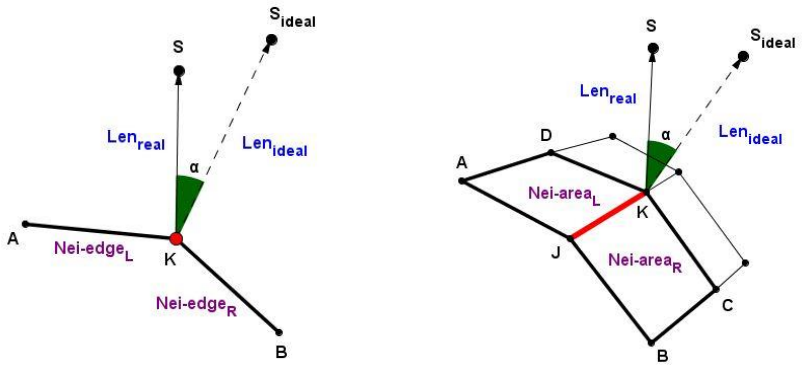
Figure 47 Modified NACA0012 airfoil mesh with AFSF method

However, for mesh movement with the current approach, a small perturbation ($1 \times 10^{-10}\ level$) of the bump function on geometry boundaries leads to a relatively large average displacement difference ($1 \times 10^{-5}\ level$) for the whole mesh, which leads to a relatively large difference of the target function. Hence, the sensitivity derivatives are inaccurate and fail the high-order cases. In the future work, the spring model could be improved to decrease the displacement difference, or this mesh moving approach could combine with other methods to satisfy the sensitivity derivatives.

CHAPTER 5

3D MESH GENERATION

The key to extending the presented 2D curved mesh generation method to 3D geometries lies in the extension of the Spring-Field force model for linear domain mesh generation and the Vector-Adding method for curved domain mesh generation.

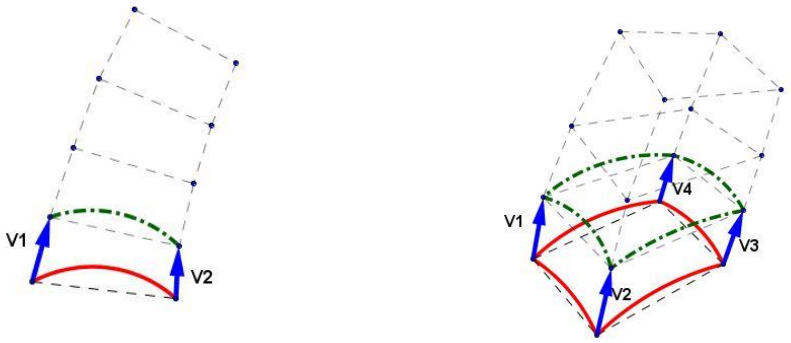


a) b)

Figure 48 Diagram of spring model extension. a) 2D spring models; b) 3D spring models

The Spring-Field force model is constructed by many individual spring units. And each of them could contain several types of spring as shown in Fig. 48. In the 2D cases, this spring unit contains three types of springs: the extrusion edge spring, the neighboring edge spring, and the angle spring. Such a spring system could suit the 3D cases as well, only some adjustments may be needed. While the extrusion edge spring and the angle spring are almost the same in 3D as in 2D, the neighboring edge spring is slightly distinct from its 2D version: in 2D, it is a function of the

lengths of the two neighboring edges; in 3D, it could be a function of the areas of the two neighboring facets, as shown in Fig. 48b). Besides these three types of springs, a new type of spring named tension spring is introduced in 3D cases. In this process, the advancing front approach in 3D is very similar to its 2D version, except that the facets are involved in advancing in 3D. The force balance process has no difference.



a) b)

Figure 49 Diagram of Vector-Adding model extension. a) 2D Vector-Adding; b) 3D Vector-Adding

As discussed in Chapter 3, the Vector-Adding method relies on the linear mesh with the pipe-like structure. If the 3D linear volume mesh is obtained by the AFSF method, the linear mesh should contain the pipe-like structure as shown in Fig. 49. The 3D curved mesh generation process can then follow a similar procedure as the one in 2D but requiring three (prism) or four (hexahedra) vectors in the generation of a new curved mesh element, instead of one or two vectors in the 2D Vector-Adding method. When the shape of the piecewise facet and the angles between the facet and the vectors are well adjusted, the Jacobian positivity can be achieved.

To obtain the linear and curved domain mesh with the AFSF method, the input linear and curved boundary meshes are needed. The way to obtain these boundary meshes is introduced in

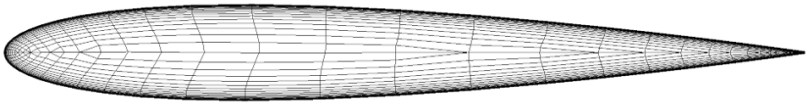
Section 5.1. In Section 5.2, the AFSF method used for linear domain mesh generation in 3D cases is introduced and the mesh results are presented. In Section 5.3, the Vector-Adding method used for curved domain mesh generation in 3D cases is introduced and the mesh results are presented.

5.1 Boundary Surface Mesh Generation

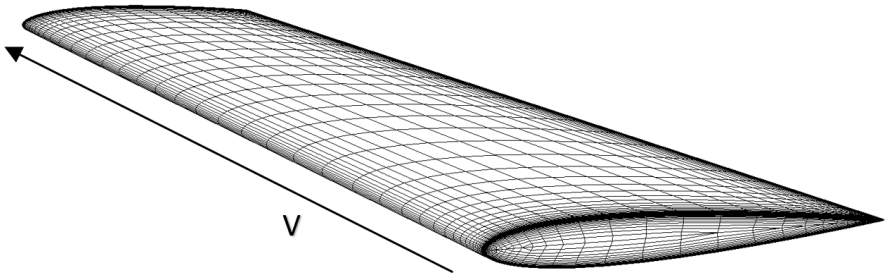
For most 3D domain mesh generation procedures, a prerequisite to the discretization of a 3D computational domain is the discretization of the boundary surface representation. In this paper, the boundary surface meshes are obtained by two methods. One of them involves the following steps:

1) Generate the 2D hybrid domain meshes by the previously described 2D AFSF method. For example, the 2D airfoil mesh shown in Fig. 50a).

2) Extrude the generated 2D mesh with a vector $V=<0,0,s>$, where s is a variable to define the extrusion length. The distribution of mesh nodes on the vector V is not even. This distribution depends on the 2D domain growth ratio and the first extrusion length through the direction of the vector V. Normally this extrusion length has the same value as the first extrusion length in the 2D domain meshes. Take Fig. 50 as an example, the mesh shown in Fig. 50b) is the closed boundary mesh after this extrusion step. The value of the first extrusion length in the direction of vector V in Fig. 50b) is same as the value of the first extrusion length in Fig. 50a).



a)



b)

Figure 50 Boundary surface mixed-elements mesh generation diagram. a) An airfoil 2D plane-surface mesh; b) A closed boundary surface mesh

After the steps described above, a closed boundary surface mesh is generated. This method is used for all airfoil-test cases.

Another method to provide the boundary meshes for the 3D mesh generation uses the commercial software package Pointwise. Pointwise is used for generating linear boundary surface mesh for the remaining test cases, which are only designed for testing the linear mesh generation.

The reason for providing curved boundary surface meshes using the first method is that it is the easiest way to generate a valid curved boundary surface mesh with positive Jacobian mesh

elements. However, the 2D AFSF mesh generation method mentioned in Section 3.1 has not been extended to 3D surface mesh generation yet, which means it can only handle 2D domain mesh generation in a plane. Since such a limitation exists, this extrusion method may be the simplest way to provide the qualified 3D curved boundary surface meshes for testing the 3D volume mesh generation approach. It can provide curved surface meshes while maintaining the positive Jacobian mesh elements on all surface facets of the mesh. This is important for 3D curved mesh generation using the 3D Vector-Adding method, which will be described in Section 5.3.

5.2 Linear Domain Mesh Generation

The linear domain mesh generation procedure for 3D meshes is similar to the procedure for 2D meshes. The mesh is generated layer by layer using the advancing front method, and a spring model named Spring-Field is used to smooth the mesh. This linear mesh generation method is named Advancing-Front-Spring-Field method with the abbreviation AFSF. The main procedure of the linear mesh generation will be discussed in Section 5.2.1, and the Spring-Field model will be discussed in Section 5.2.2. Same as the 2D AFSF method, this 3D AFSF method provides a hybrid mesh as the raw output of this linear mesh generator. Different than the raw output mesh of the 2D AFSF method, the 3D output mesh could contain several non-standard types of mesh elements. For example, the 2D raw output mesh is a hybrid mesh which only contains triangles and quadrilaterals. If the numerical solver can handle this type of hybrid mesh, there is no need to change the mesh topology. But in the 3D raw output mesh of the AFSF method, there could be several non-standard types of mesh elements such as a mesh element containing 7 points with a quadrilateral as the bottom facet and a triangle as the top facet. There are few, if any, solvers that can handle this type of mesh element. Hence, the mesh topology must be changed to serve the

general numerical solvers. There are two ways of changing the mesh topology. One is converting all mesh elements into tetrahedra. Another is converting all mesh elements into the standard types of mesh elements, namely hexahedral, prism, pyramid and tetrahedral. These final meshes could be obtained by cutting or assembling the raw non-standard mesh elements. However, in this research only the operation of cutting is used. The strategy of obtaining a pure tetrahedral mesh is described in Section 5.2.3 and Section 5.2.5. The strategy of obtaining a hybrid mixed-elements mesh is described in Section 5.2.5.

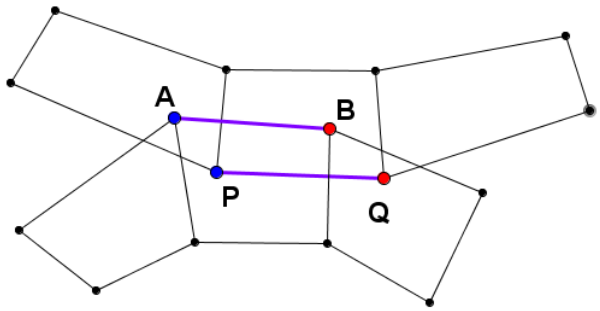


Figure 51 2D front node closing sketch

In 2D advancing-front methods, the front-closing only needs to control the front-node-closing. If the front nodes overlap, the front edges will automatically match. Take the example in Fig. 51. If node *A* finds the matched node *P*, node *B* finds the matched node *Q* and the two matched couples overlap, edge *AB* and edge *PQ* will automatically overlap. If node A and B find the same node *P* to merge with for the front-node-closing, edge *AB* will disappear and edge *PQ* may find another edge to overlap. Hence there is no need to find a special strategy of changing the topology of the mesh to let the front edges match. But 3D situations are completely different. Take the example in Fig. 52. If node *A* finds the matched node *P*, node *B* finds the matched node *Q*, node

D finds the matched node *R*, and the three matched node couples overlap, the related edges do not automatically match, let alone the related facets such as triangle *PQR* and quad *ABCD*. Hence, a special strategy of matching the front edges is needed. Note, if the front-edges match, the front facets will automatically match. This special strategy is introduced in Section 5.2.4. Unfortunately, the strategy has some limitations. For example, it only works for the pure triangular front surface, and the facets which overlap each other must have similar sizes. This strategy cannot handle the 3D front-closing case as shown in Fig. 52, which contains quads and triangles.

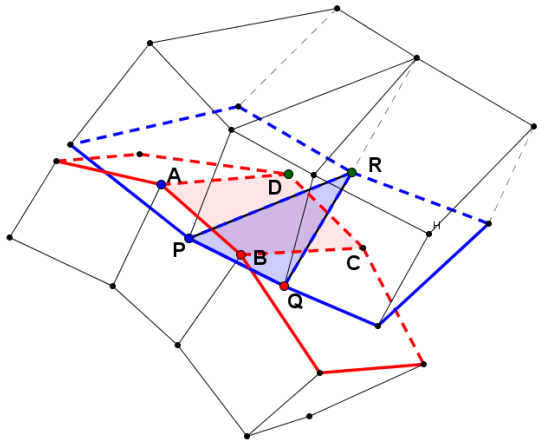


Figure 52 3D front node closing sketch

5.2.1 Procedure of the linear viscous domain mesh generation

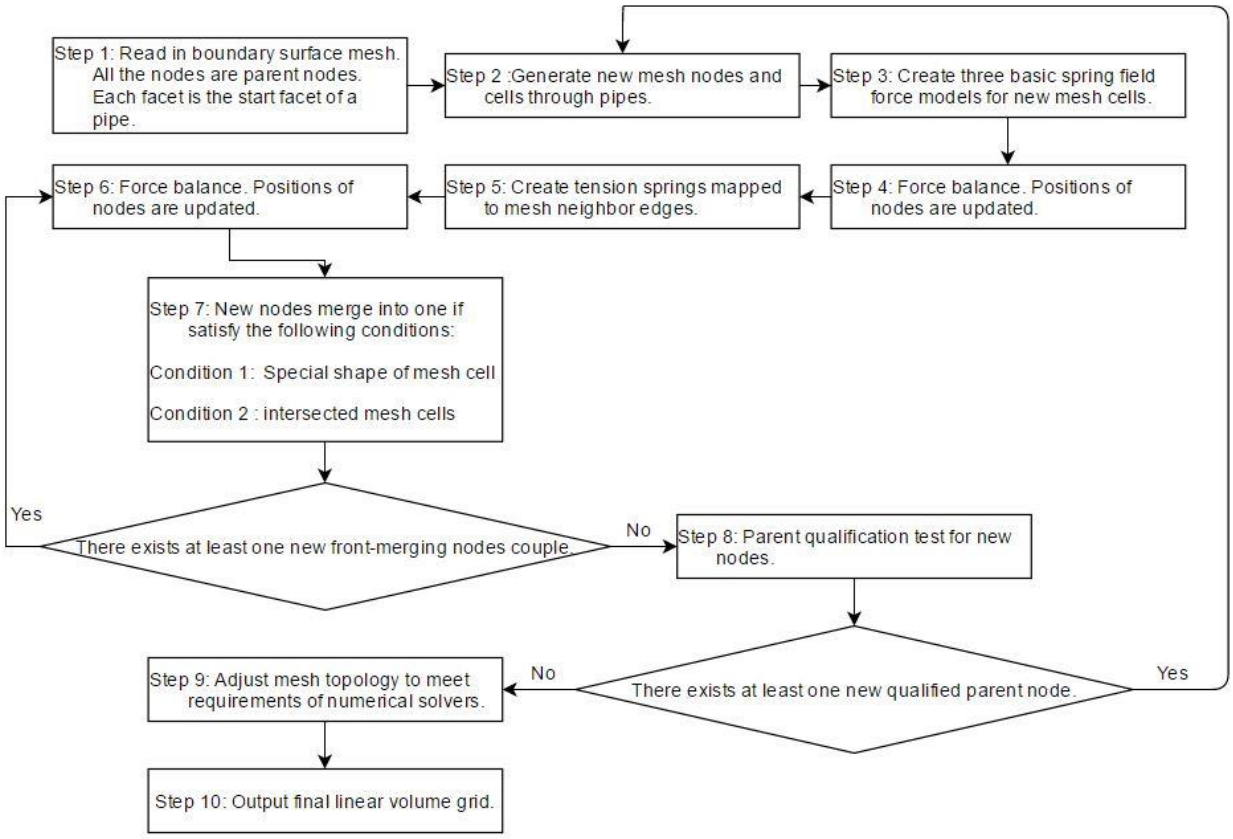


Figure 53 Procedure of the linear domain mesh generation

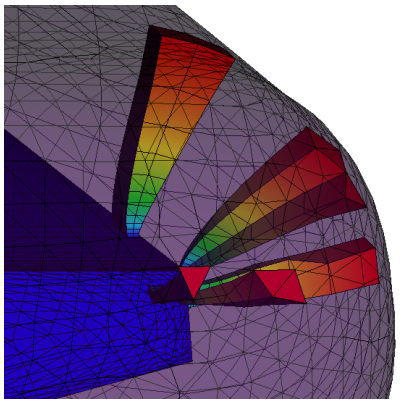


Figure 54 Mesh cells generation diagram

Fig. 53 shows the main procedure of the linear viscous domain grid generation. In step 2, for the previous 2D approach, each parent node could have multiple child nodes. Currently, the 3D approach only allows the parent node to have one child node, which simplifies several other steps in the procedure. In the future, the strategy of multiple child nodes could be added in an effort to better represent the domain around a corner. The generation of mesh cells through "Pipes" is shown in Fig. 54. As new mesh cells are created layer by layer, the lengths of pipes increase. In step 3, there are three basic spring types: the extrusion edge spring, the neighboring edge spring and the angle spring. The concept of these spring types is inherited from the 2D approach. These spring types will be discussed in Section 5.2.2 together with a new spring type named the tension spring type, which is shown in step 5 in Fig. 53. In step 4, the force equations are solved explicitly. There is no need to involve all springs during force balance. Normally the springs related to the top three advanced mesh layers are enough for a better mesh quality. Step 7 contains the merging operations of mesh nodes. There are two types of front-node merging. One is the merging of the nodes, which are connected to each other by an edge. They are neighboring nodes which generally extrude almost in the same direction. Another is the merging of the mesh nodes which are not linked by an edge. Generally, they extrude from different boundaries.

There are several conditions for judging if the front neighboring nodes should merge. For efficiently representing the domain, the length ratio of a neighboring edge to its attached extrusion edge should be above a certain threshold. If this ratio is smaller than a specified global factor, the neighboring edge breaks and the two neighboring nodes attached to this edge are merged. The other triggering condition applies if a mesh cell is twisted to become a negative-volume mesh element. The conditions of determining if the front nodes without a linked edge should merge will be introduced in Section 5.2.4. Since the mesh cells are generated through the pipes layer-by-layer,

this front-closing strategy stimulates tissue growth, i.e. the tissue grows from the original surface layer by layer until it runs out of space. The method of finding front-merging nodes not linked by an edge is also described in Section 5.2.4. In step 9, since the neighboring nodes can merge with each other, several non-standard mesh cell types could be generated. The strategy of how to adjust topology by cutting these mesh cells to suit numerical solvers will be discussed in Section 5.2.3 and Section 5.2.5.

5.2.2 Spring-Field force model

In the 3D mesh generation approach, all three basic spring types are inherited from 2D approach: the extrusion edge spring, the neighboring edge spring and the angle spring.

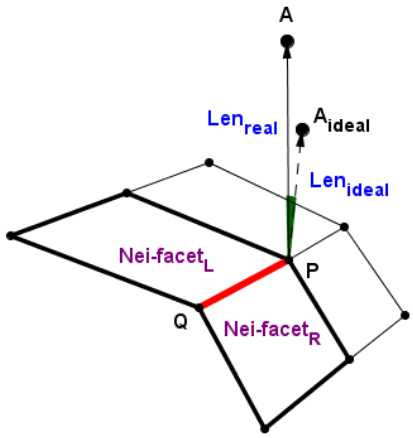


Figure 55 The extrusion edge spring diagram

The extrusion edge spring is mapped to the edge that links parent and child mesh nodes from different layers. It applies forces on two nodes. The purpose of this spring type is to push the child node to the designed ideal position, and away from their parent nodes and mesh boundaries,

as shown in Fig. 55. Node P is the parent node of node A. Edge PA maps to one extrusion edge spring with the force model on node A as:

$$\overrightarrow{F_{extru\_eg}} = K_{extru\_eg} \times C_{aniso} \times \left(\frac{\|\overline{PA}\|_{ideal} - \|\overline{PA}\|_{real}}{\|\overline{PA}\|_{real}}\right) \times \overrightarrow{PA}$$

where K_{extru_eg} is the extrusion edge spring coefficient, which is set to be 2 in the numerical examples in the mesh results section. $\|\overline{PA}\|_{real}$is the length of edge PA. The definition of ideal length $\|\overline{PA}\|_{ideal}$ is the same as the one in the 2D method. It is a given global factor which grows exponentially as the mesh layer increases. The expression of the spring force model on node P is similar with a opposite force direction $\overrightarrow{AP}$ and different C_{aniso}. C_{aniso} is a anisotropic coefficient with the definition as same as the one in the 2D method to provide a proportional compression of edges[55]. For node Pin the lower mesh layer, $\mathrm{C_{aniso}} = \frac{1}{G_{ratio}}$, where growth ratio is a given global mesh extrusion growth factor (e.g. 1.1,1.2,1.3, etc.). For node A in the upper mesh layer, $C_{aniso} = 1$.

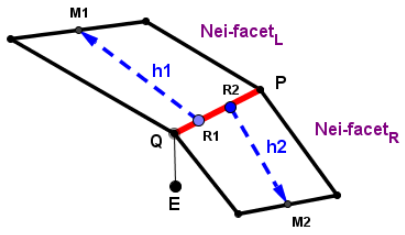


Figure 56 The neighboring edge spring diagram

The neighboring edge spring is mapped to the edge that links neighbor nodes on the same layer. It applies a force on two nodes. The purpose of this spring type is to dissipate the difference between the sizes of neighbor mesh cells which are inherited from the difference between the sizes

of their ancestor mesh cells on the mesh boundaries. This difference between neighbor mesh cells for 2D is the difference between the lengths of neighboring edges, and for 3D is the difference between the areas of neighbor facets, as shown in Fig. 56. Node *Q* is the neighboring node of node *P*. Edge *QP* maps to one neighboring edge spring which tries to push edge *QP* towards facet Nei_facet_L, since the area of Nei_facet_L is larger than that of Nei_facet_R. Neighboring edge *QP* maps to one neighboring edge spring with the force model on node *P* and *Q* as:

$$\overrightarrow{F_{nei\_eg}} = K_{nei\_eg} \times C_{aniso} \times \left(\frac{h_1}{h_1 + h_2}\|\overline{QE}\|_{ideal} \times \frac{\overrightarrow{R_1M_1}}{\|R_1M_1\|} + \frac{h_2}{h_1 + h_2}\|\overline{QE}\|_{ideal} \times \frac{\overrightarrow{R_2M_2}}{\|R_2M_2\|}\right)$$

where K_{nei_eg} is the edge spring coefficient, which is set to be 0.25 in the numerical examples in the mesh results section. $\overline{R_1M_1}$ and $\overline{R_2M_2}$ are maltitudes of the quadrilaterals at each side of edge *QP*. $\overline{R_1M_1}$ or $\overline{R_2M_2}$ is an altitude if the corresponding mesh cell is a triangle, where *M1* or *M2* should be the corresponding vertex. $\|\overline{QE}\|_{ideal}$ is the ideal length of the edge *QE*. Its role in the formula is a base unit of distance. By this definition, $Force_{edge\_neighbor}$ is equal to some percentage of $\|\overline{QE}\|_{ideal}$. Coe_{Aniso} is a coefficient used to distinguish anisotropic mesh cells to isotropic ones. The definition of C_{aniso} is $C_{aniso} = \min(1, \frac{2\|\overline{QE}\|_{ideal}}{h_1+h_2})$. If the mesh cell Nei_facet_L is in the boundary layer and is anisotropic in terms of the direction $\overrightarrow{R_1M_1}$, C_{aniso} is small. That will cause F_{nei_eg} to be small, which means F_{nei_eg} will cause less perturbation of the node *P* and *Q*. In practice, there is no need to calculate each value of factor in the neighboring edge spring model, such as $h_1, h_1 + h_2$, etc., during each iteration in the force balancing process. Some of them could use the initial value without updating in each iteration, or could be calculated using dot product or cross product without the need for the square root function.

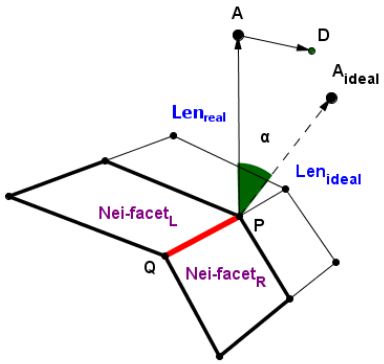


Figure 57 The angle spring diagram

The angle spring model is like the one in the 2D method. It applies forces on two nodes. The purpose of this spring type is to keep the extruding direction of the target node close to the weighted average normal of the facets around this target node's parent node, so that mesh quality can be better. In Fig. 57, node *A* is the target node and *P* is its parent node. Vector $\overrightarrow{PA_{ideal}}$ defines the ideal extruding direction, which is calculated based on the normals of the facets around node *P*. The force of the angle spring is a function of the deviation angle α. Angle α is defined as the angle between the real extruding direction $\overrightarrow{PA}$ and the ideal extruding direction $\overrightarrow{PA_{ideal}}$ of node *A*. The force model of the angle spring is defined as:

$$F_{ang} = K_{ang} \times C_{percent} \times f(\alpha_{appro}) \times \|\overline{PA}\|_{ideal}$$

where K_{ang} is the angle spring coefficient, which is set to be 1 in the mesh results section. $\|\overline{PA}\|_{ideal}$ is the ideal length of extrusion edge *PA* as the distance measure. $f(\alpha_{appro})$ is a polynomial function of α_{appro} defined as $f(\alpha_{appro}) = 8{\alpha_{appro}}^3 + 0.2\alpha_{appro}$, where $\alpha_{appro} = sin\alpha$. $Sin\alpha$ is calculated through the cross product of vectors $\overrightarrow{PA_{ideal}}$ and $\overrightarrow{PA}$ instead of calculating α to decrease the computation cost. For the angle spring force on node *A*, the direction

is the same as vector $\overrightarrow{AD}$. $\overrightarrow{AD}$ is in the plane APA_{ideal} and is perpendicular to $\overrightarrow{PA}$. For the force on node *P*, the direction is the same as vector $\overrightarrow{DA}$. $C_{percent}$ is the percentage of the force distributed to the two nodes (in Fig. 57 nodes *A* and *P*). The sum of $C_{percent}$s of the two nodes is 1. In the mesh results section, $C_{percent}$ is 0.5 for the both nodes.

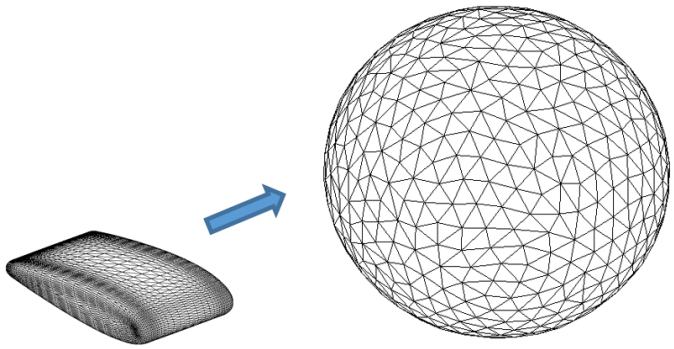


Figure 58 The shape change diagram

In addition to these three-basic type of springs, there is a new spring type introduced for 3D linear mesh generation. This spring is mapped to each neighboring edge and has force on two nodes. The purpose of this spring type is twofold. One is to control the angles in neighbor facets (defined as facets enclosed by neighboring edges) and the other is to simulate surface tension to smooth the mesh. In Fig. 57, the angle spring linked to node *P* can only control the angle between the extrusion edge and the neighbor facets around node *P*. The neighboring edge spring mapped to *QP* pushes *QP* to move in parallel to its original position. So, it cannot control the angle between the neighboring edges either. If there is a small perturbation that causes a rotation of the neighboring edge *QP* on the plane of neighbor facets, there is no force to deliberately rotate the edge back. In practice, under the three-basic type of spring forces, this small perturbation seems to be amplified as mesh layers advance, and a large consequent perturbation could lead to large

obtuse angles in neighboring facets. Hence an extra spring type is needed to control the angles between neighboring edges. In addition, by building and mapping such springs to each neighboring edge, all the neighbor facets on the same mesh layer mimic a stretched elastic membrane. These new springs will act like surface tension, which make the mesh smoother. As shown in Fig. 58, this spring type helps the mesh layer change its shape from the shape of the object to a sphere faster. Hence, we name this spring type as a tension spring type. As shown in Fig. 57, one tension spring maps to neighboring edge *QP* with the force model on node P as:

$$\overrightarrow{F_{tension}} = K_{tension} \times C_{aniso} \times \left(\frac{\|\overline{QP}\|_{ideal} - \|\overline{QP}\|_{real}}{l_{charac}}\right) \times \|\overline{QE}\|_{ideal} \times \frac{\overrightarrow{QP}}{\|\overline{QP}\|_{real}}$$

where $K_{tension}$is the tension spring coefficient, which is set to be 0.25 in the numerical examples in the mesh results section. $\|\overline{QP}\|_{ideal}$ must be given after first force balancing step (Step 4 in Fig. 53), which means tension springs are not created at the same time with the other three spring types. $\|\overline{QP}\|_{ideal}$ is set to be $0.9\|\overline{QP}\|_{real}$ in the mesh results section. The role of $\|\overline{QE}\|_{ideal}$ is the distance measure. By this definition, $Force_{tension\_spring}$ is equal to some percentage of $\|\overline{QE}\|_{ideal}$. l_{charac} is a characteristic length. It is defined as the maximum length of neighboring edges linked to node P. C_{aniso} is a coefficient used to distinguish anisotropic mesh cells to isotropic ones. The definition of C_{aniso} is $C_{aniso} = \frac{\|\overline{QE}\|_{ideal}}{l_{charac}}$. The expression of the force model on node *Q* is the same as on node *P* except that the force direction for node *Q* is $\overrightarrow{PQ}$.

5.2.3 Topology adjustment method 1

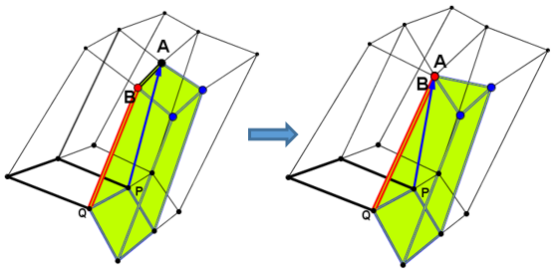


Figure 59 The nodes-merging diagram

As the mesh layer advances, several mesh cells change their element type to increase the efficiency of node distribution. This leads to the creation of several non-typical mesh elements. As shown in Fig. 59, the aspect ratio of edge *BA* to edge *BQ* is smaller than the given global factor Coe_{shrink} (this factor is 0.65 in the mesh results section). Hence nodes *B* and *A* merge with each other and the green hexahedral changes its shape to a non-typical mesh element, which has a quadrilateral on bottom but a triangle on top. Generally, this type of mesh cell will not be recognized by numerical solvers. The mesh topology around the node *A*/*B* needs to be changed to erase several mesh cells and re-generated some mesh cells of standard mesh element types. Another situation which requires a mesh topology adjustment is where the advancing-front facets merge. In 2D cases, if the two nodes of a neighboring edge merge with two nodes of another neighboring edge from the opposed extruding direction, these two neighboring edges automatically merge. This is more difficult in 3D, because when the advancing fronts collide, the facets may not align even though there are no hanging nodes. The mesh topology adjustment method used for this is discussed in Section 5.2.4.

In this section, we only focus on the topology adjustment to provide a pure tetrahedral mesh. The pre-condition of this adjustment is that the input boundary surface mesh in Step 1 as shown in Fig. 53 must be a triangular mesh. If the input surface mesh is a mixed mesh contains triangles and quadrilaterals, the pipes based on surface quadrilaterals must be divided into two pipes based on triangles. This operation currently has not been added into the linear mesh generation approach. The method of the topology adjustment, or topology re-generation, has the following steps:

1) For all pipes, boundary surface facets are their bottom facets. Set all boundary surface facets as their top triangular facets. Pick mesh layer 1 (the boundary input mesh layer) as the starting mesh layer *X*.

2) Between mesh layer *X* and *X*+1, pick one extrusion edge *L*, which links parent node *P* and its child node *A*. Create n tetrahedra around edge *L* based on the n top triangular facets of n pipes around node *P*. Update the n top tetrahedral facets of the matched n pipes.

3) Pick another extrusion edge and repeat the same operations until all extrusion edges between mesh layer 1 and 2 are chosen. If there is a merging node group which contains several extrusion edges, do operations in Step 2 on these extrusion edges simultaneously.

4) Repeat step 2 and 3 on the extrusion edges over mesh layer *X*+1, *X*+2, *X*+3… until all extrusion edges of the whole mesh are chosen.

As shown in Fig. 60, based on extrusion edge *PA*, six green tetrahedra (solid line) are generated. Then based on extrusion edge *QB*, six orange tetrahedra (dash line) are generated, among which there are two tetrahedra based on the new generated green triangular facets. As shown in Fig. 61, if the merging nodes group *A* & *B* contains two extrusion edges *PA* and *QB*, ten

tetrahedra will be generated simultaneously. Note that the order of picking extrusion edges can affect the final tetrahedral mesh quality significantly. The current strategy is to start with the extrusion edge which has the smallest deviation angle from its ideal extruding direction. The extrusion edges around anisotropic mesh elements are also picked first. If there is a tetrahedron with a larger dihedral angle, switch the newest chosen extrusion edge with the related extrusion edge which constructs this tetrahedron.

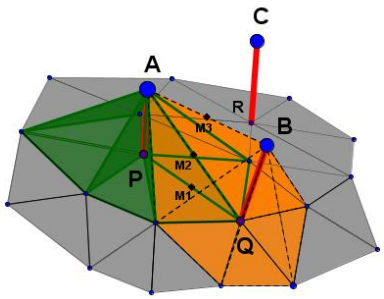


Figure 60 The Tet topology generation diagram

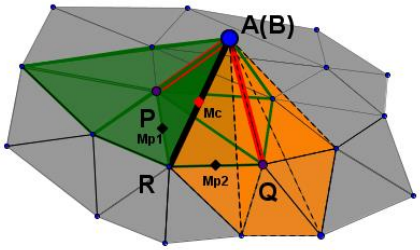


Figure 61 The special Tet topology generation diagram

5.2.4 3D front closing

3D front closing is a challenging research topic. Not only do the front nodes have to find matched nodes to overlap, but the front edges also need to find matched edges to overlap. For merging the 3D front nodes, two methods are used.

The first one is similar to the method used in the 2D front closing. If an extrusion edge has an intersection point with a front facet, the front node which is the top node of this extrusion edge will find a vertex of the front facet to overlap. The following paragraph describes how to find the right front facet for each extrusion edge, if there is any, which is the key to this front closing method.

The facets around the top front node of an extrusion edge are defined as the attached front facets of this extrusion edge. For each extrusion edge, first find several front facets which are close to this extrusion edge and are not its attached front facets with the help of k-d tree. The k-d tree library used here was created by Matthew Kennel[70]. Define the group of these picked front facets as group *A*. As simulating the tissue growth, the initial length of this extrusion edge is 0, which means its top front node overlaps with its bottom node. Then extrude the top front node by a distance *l* defined as $l = Ideal\_Len_{extru\_edge}/n$, where $Ideal\_Len_{extru\_edge}$ is the ideal length of this extrusion edge and n is a given total number of growth steps. Note as the extrusion edge increases its length, the positions of all the front facets also change. Then detect if there is any intersection point between this extrusion edge and the facets in group *A* at this growth step. If there is one facet has an intersection point, based on the position of this intersection point to pick one vertex of this facet as the front-merging node to the top front node of this extrusion edge. Repeat this growth step for n times.

The second method is that after the mesh topology changing, the two facets which attach to the same new front-merging edge either overlap each other or construct an angle α. If the angle α is smaller than a global given factor, for example, $\frac{\pi}{2}$, the edge is qualified for checking the front-node merging and the nodes around this new front-merging edge will be checked. Take Fig. 62 as an example, edge *BC* has four attached facets from two opposite extruding directions. Two attached facets overlap as shown in the brown color. The other two are triangle *ABC* and triangle *PCB*. If the angle α between ΔABC and ΔPCB is smaller than $\frac{\pi}{2}$ (or another given global value), the operation of finding the couple of the front nodes to merge is activated. Rotate node *P* by angle α with the rotation axis *BC* to obtain a node P' on the plane *ABC*. Note that node P' may be located outside ΔABC. Then check the distance between node P' and all the nodes in node group *AA* which is defined as the group of node *A* and its neighboring nodes. Find node $node_{target}$ in node group *AA* which has the minimum distance with node P'. Node $node_{target}$ is the front-merging node for node P'. Then overlap nodes P' and $node_{target}$. This strategy only works if the triangle *ABC* and the triangle *PCB* are similar in size. If one of them is distinctively larger, the mesh quality could be inferior even with the help of the Spring-Field force model. After merging nodes *A* and *P*, edge *PB* and *CP* may become the qualified edges for checking the front node merging. This process is repeated until no front edge qualified the checking conditions. The reason to add this method into the 3D front-closing process is that it decreases the number of "sharp corners" in the mesh, which decreases the difficulty of 3D front-closing for the first 3D front-closing method. Still, the robustness could be undermined due to a high level of difficulty.

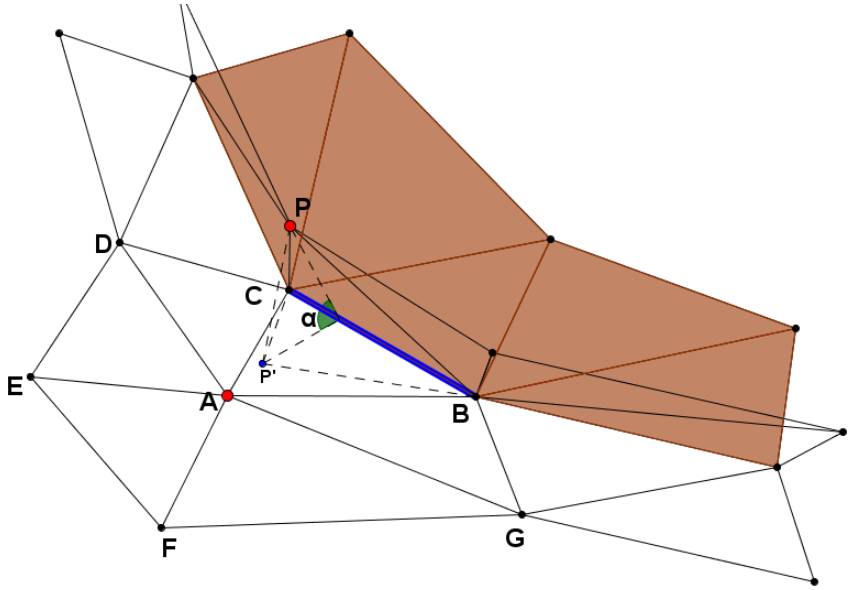
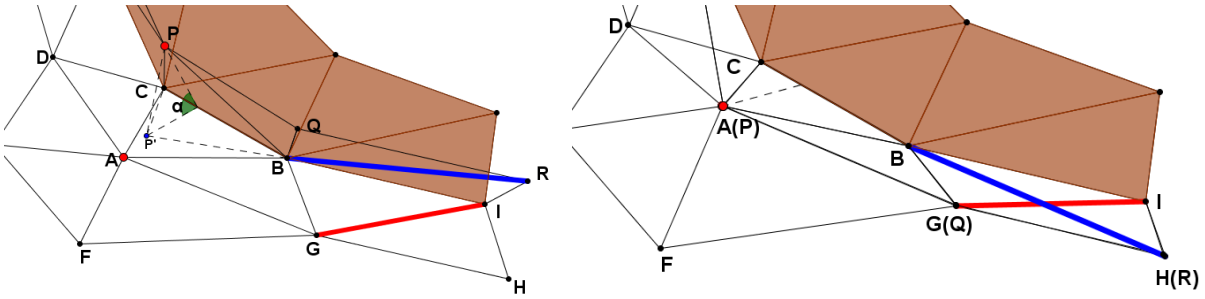


Figure 62 The 3D front-closing diagram

While the front nodes are merging, the front edges should also merge. Unfortunately, sometimes the front edges cannot merge unless the mesh topology changes. Take the case in Fig. 63 as an example, after node *Q* merges with node *G*, node *R* merges with node *H*, edge *BR* or edge *GI* must be swapped to merge with the other one. After the qualified edges are swapped, the edges like edge *BC* should be checked again to find the front nodes qualified for merging. Repeat these steps until no new front nodes qualified for merging could be found.



a) b)

Figure 63 The edge swapping diagram. a) before front-node merging; b) after front-node merging.

Only two cross-edge situations are taken into consideration for swapping edges. The first one is shown in Fig. 63. The second one is shown in Fig. 64.

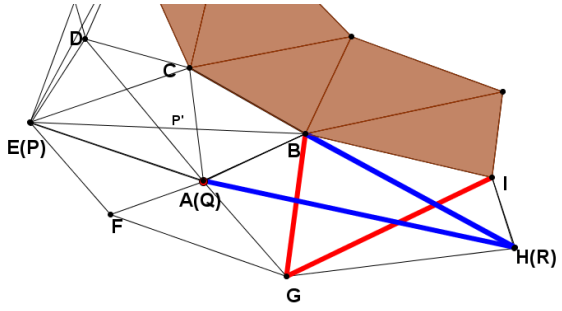


Figure 64 The edge swapping diagram 2

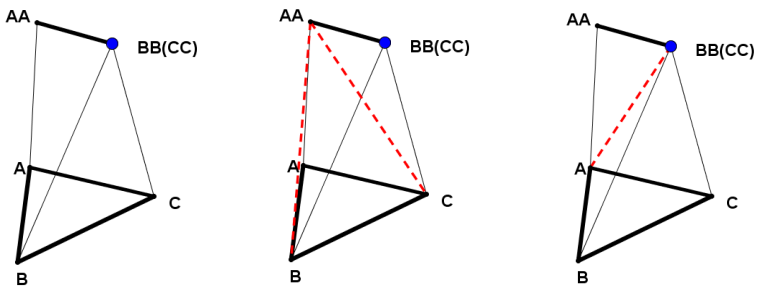
5.2.5 Topology adjustment method 2

The method of cutting the obtained non-standard mesh elements into standard mesh elements in Section 5.2.3 can only output a pure tetrahedral mesh. If a mixed-element output mesh is needed, another topology adjustment method should be developed.

After the front-node merging and the front-edge merging, the topology of the front facets, which are triangles, is fixed and will not change in the future. The type of non-standard mesh element is shown in Fig. 65a). It is a mesh cell with a triangle as the bottom facet and an edge as the top. This type of mesh element is generated due to the merging of node *BB* and node *CC*. The reason of the merging could be a relatively small edge ratio described in Section 5.2.3. The merging could also be caused by the situation where node *BB* and *CC* share the same front-merging

node *X* which extrudes from another direction; in such case, node *BB* merges with node *X*. Then node *CC* merges with node *X* which leads to node *BB* merging with node *CC*.

There are two ways to cut this non-standard mesh cell into standard mesh elements. One is as shown in Fig. 65b). The mesh cell is cut into one tetrahedral and one triangle. The other is shown in Fig. 65c). The mesh cell is cut into two tetrahedral. If this non-standard mesh cell does not have neighboring mesh cells, the job is done. The challenging part of this mesh topology adjustment is that, the mesh cell has neighboring mesh cells. If new cutting-edge *BAA* or the new edge *ABB* is created by the cutting operation, the neighboring mesh cell which share the side quad with this non-standard mesh cell must be cut to match the new edge *BAA* or *ABB*. Besides the non-standard mesh cells, another situation in which the mesh topology must be adjusted is that the mesh cells relate with the swapped-edge. Recall the cases in Fig. 63 and Fig. 64. After the edge swapping, the generated prisms in the front mesh layer are extremely twisted with respect to the original mesh topology. These mesh cells must be fixed. In the following paragraph, the method to cut these mesh elements to match each other will be discussed.



a) b) c)

Figure 65 The non-standard mesh element type diagram

If there is a non-standard front mesh cell $PQC - P'C'$ as shown in Fig. 66, all the front mesh cells around node C' could be cut by the cutting edges which link to node C' as shown in red and blue dash lines (for convenience, this operation will be referred to as "round-cutting"). All the obtained standard mesh cells (tetrahedra and pyramids) around node C' have the matched edges and facets. Repeat this "round-cutting" operation on all the non-standard mesh elements and the mesh cells involved with the swapped-edge, the final mixed-elements mesh could be obtained. Note if the round-cutting operation first acts on node C' then acts on node B', the cutting edge on the facet $BCC'B$ will be CB' instead of BC'. The final mixed-elements mesh is obtained based on these cutting edges. There is no need to do this round-cutting on each front node. But if doing this round-cutting on each front node and repeat this on each front mesh layer for each extrusion step, the final obtained mesh will be a pure tetrahedral mesh.

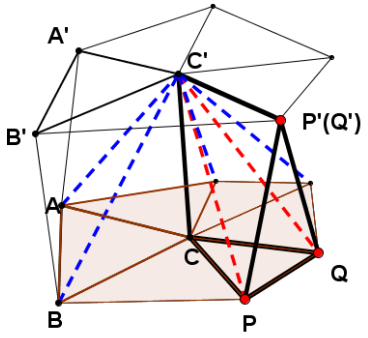


Figure 66 The round-cutting diagram

To improve the mesh quality, a ranked node list for round-cutting is provided. In some cases, the ranking of the node list will significantly affect the final mesh quality in terms of the poorly shaped tetrahedra. Since this strategy is still under development, the details are not discussed in this dissertation.

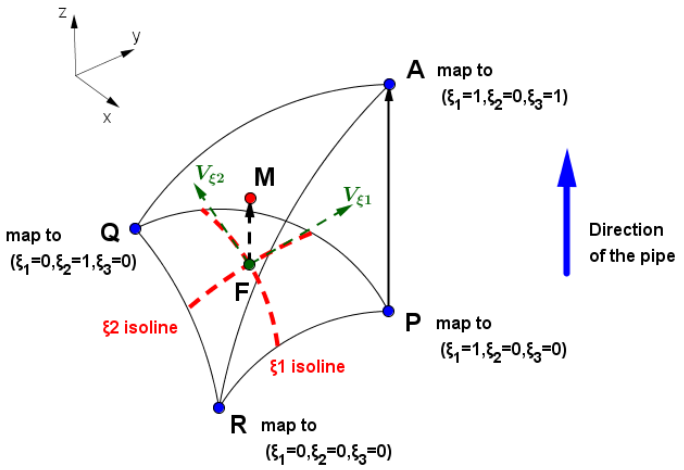


Figure 67 Positive Jacobian diagram

The curved domain mesh generation process for 3D meshes is like the one for 2D meshes. In 3D cases, the mesh is deformed one element at a time through the pipe-like mesh structure (such as the ones in Fig. 54) as same as 2D cases. The mesh is deformed by the Vector-Adding method. By this method, the deformed curved mesh has a high probability to be an all-positive-Jacobian-element mesh which can be attributed to the same mechanism demonstrated in the 2D Vector-Adding method. The key is still the range of the angle between the isolines. As shown in Fig. 67, suppose the parametric expression of the base curved facet *PQR* is given as $\overrightarrow{Sur}_{(\xi_1,\xi_2)}$ and this curved facet is a positive-Jacobian mesh element, where ξ_1 and ξ_2 are computational coordinates. *PA* is an extrusion edge with $\overrightarrow{PA} = \vec{V}(x_{PA}, y_{PA}, z_{PA})$, which is constant. The parametric tetrahedron *PQRA* is created as: $\overrightarrow{Sur}_{(\xi_1,\xi_2)} + (1-\xi_1)\xi_3\vec{V}$. Based on this expression, the Jacobian of a random node *M* in Tet *PQRA* is

$$Jacob_M = \begin{vmatrix} \frac{\partial x_M}{\partial \xi_1} & \frac{\partial x_M}{\partial \xi_2} & \frac{\partial x_M}{\partial \xi_3} \\ \frac{\partial y_M}{\partial \xi_1} & \frac{\partial y_M}{\partial \xi_2} & \frac{\partial y_M}{\partial \xi_3} \\ \frac{\partial z_M}{\partial \xi_1} & \frac{\partial z_M}{\partial \xi_2} & \frac{\partial z_M}{\partial \xi_3} \end{vmatrix} = \left(\frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_1} - \xi_{3_M} \vec{V} \right) \cdot \left[\frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_2} \times \left(1 - \xi_{1_M}\right)\vec{V} \right] = \left(1 - \xi_{1_M}\right)\vec{V} \cdot \left[\left(\frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_1} - \xi_{3_M} \vec{V} \right) \times \frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_2} \right] = \left(1 - \xi_{1_M}\right)\vec{V} \cdot \left[\frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_1} \times \frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_2} \right].$$

Suppose there is a node F on curved facet PQR with computational coordinate $(\xi_{1_F}, \xi_{2_F}) = (\xi_{1_M}, \xi_{2_M})$. We have $\left[\frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_1} \times \frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_2} \right] = \left[\frac{\partial \overrightarrow{Sur}_{(\xi_{1_F}, \xi_{2_F})}}{\partial \xi_1} \times \frac{\partial \overrightarrow{Sur}_{(\xi_{1_F}, \xi_{2_F})}}{\partial \xi_2} \right]$. As shown in Fig. 67, $\overrightarrow{V_{\xi 1}}$ is on the tangent line of ξ_2 isoline through node *F* and $\overrightarrow{V_{\xi 2}}$ is on the tangent line of ξ_1 isoline through node *F*. Hence $\overrightarrow{V_{\xi 1}} = \frac{\partial \overrightarrow{Sur}_{(\xi_{1_F}, \xi_{2_F})}}{\partial \xi_1}$ and $\overrightarrow{V_{\xi 2}} = \frac{\partial \overrightarrow{Sur}_{(\xi_{1_M}, \xi_{2_M})}}{\partial \xi_2}$. Then $Jacob_M = \left(1 - \xi_{1_M}\right)\vec{V} \cdot \left[\overrightarrow{V_{\xi 1}} \times \overrightarrow{V_{\xi 2}}\right]$. So, if the direction of $\overrightarrow{V_{\xi 1}} \times \overrightarrow{V_{\xi 2}}$ points towards the upper mesh layer, $Jacob_M$ should be positive because vector $\overrightarrow{PA}$ is close to its ideal extruding direction which is obtained based on the normal of linear triangle PQR and the normal of its neighbor facets. Recall the assumption that the input boundary mesh is smooth and contains all positive-Jacobian facets. If curved facet *PQR* is one of boundary surface facets, $\overrightarrow{V_{\xi 1}} \times \overrightarrow{V_{\xi 2}}$ should satisfy this condition so Tet *PQRA* should be a positive-Jacobian mesh cell. Hence the new curved triangular facet *AQR* is positive-Jacobian and the cross product of the tangent vectors of the isolines on facet *AQR* should point to the next layer, which benefits the positive-Jacobian tetrahedra generation through the pipe.

In practice, similar to the previous 2D approach, there is no need to calculate the parametric expressions of tetrahedra first to provide high-order nodes. High-order nodes can be obtained by

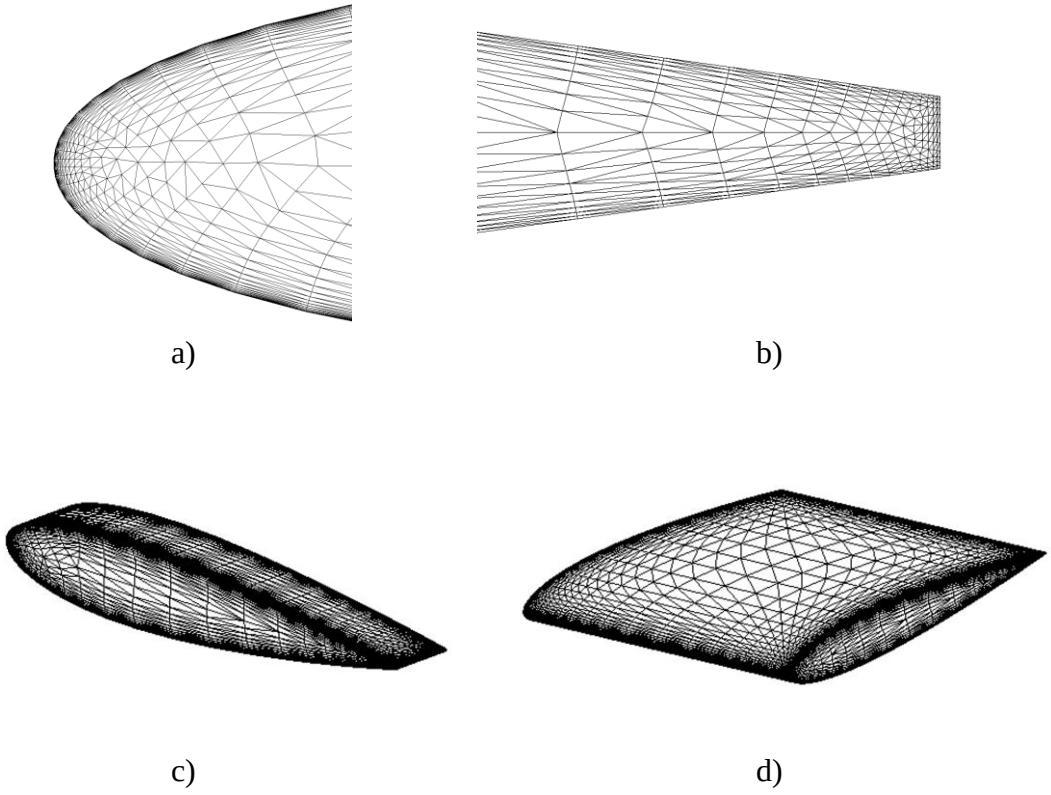
Vector-Adding. Take Fig. 60 for example, high-order node *M2* is obtained based on high-order node *M1*, which is the midpoint of edge *PQ*, adding vector $0.5\overrightarrow{PA}$. High-order node *M3* is obtained based on high-order node *M2* adding vector $0.5\overrightarrow{QB}$. If two child nodes *A* and *B* merge with each other as shown in Fig. 61, the operation is slightly different. The position of high-order node *Mc* is obtained by averaging position *Mc1* and *Mc2* (not shown in Fig. 61). Position *Mc1* equals to the position of high-order node *Mp1* adding $0.5\overrightarrow{PA}$. Position *Mc2* equals to the position of high-order node *Mp2* adding vector $0.5\overrightarrow{QB}$. Note in this situation the parametric expression of the new tetrahedra around edge *RA* is different and the previous parametric expression is not suitable anymore. The Vector-Adding strategy for mesh cells which are not tetrahedra is similar to the one for tetrahedra. The only difference is the vector becomes a weighted vector combined by multiple extrusion vectors mapped to related extrusion edges, the same as the Vector-Adding strategy used for quadrilateral in the 2D approach. (See Section 3.3)

5.4 Mesh Results

To verify the 3D high-order viscous mesh generation approach proposed above, several examples are provided. 5.4.1 describes the input boundary surface meshes. 5.4.2 provides several results of linear and curved 3D mesh generation. The Jacobians of the pure parabolic tetrahedral meshes in the NACA0012 cases are checked using a subroutine in the numerical solver, which will be described in Section 5.6 CFD results. The checked result shows that all the checked mesh elements have positive-Jacobians.

5.4.1 3D NACA0012 airfoil boundary surface mesh

Based on the linear and curved 2D NACA0012 mesh generated by the previous 2D mesh generation method, the 3D boundary surface meshes are obtained as the input meshes for 3D mesh generation. The first extrusion lengths in the 2D and 3D mesh generations are 1×10^{-3} or 1×10^{-4}. The growth ratios in the 2D boundary mesh generation, in the 2D domain mesh generation, and in extending this 2D airfoil to obtain the 3D boundary surface mesh are same, which are 1.3 or 1.4 in different cases. Fig. 68a) and b) show a close view of the 2D airfoil mesh with $Growth_{ratio} = 1.4$. Fig. 68c) and d) show the final 3D surface meshes obtained by extruding 35 more layers along the Z axis, which is perpendicular to the "2D" airfoil.



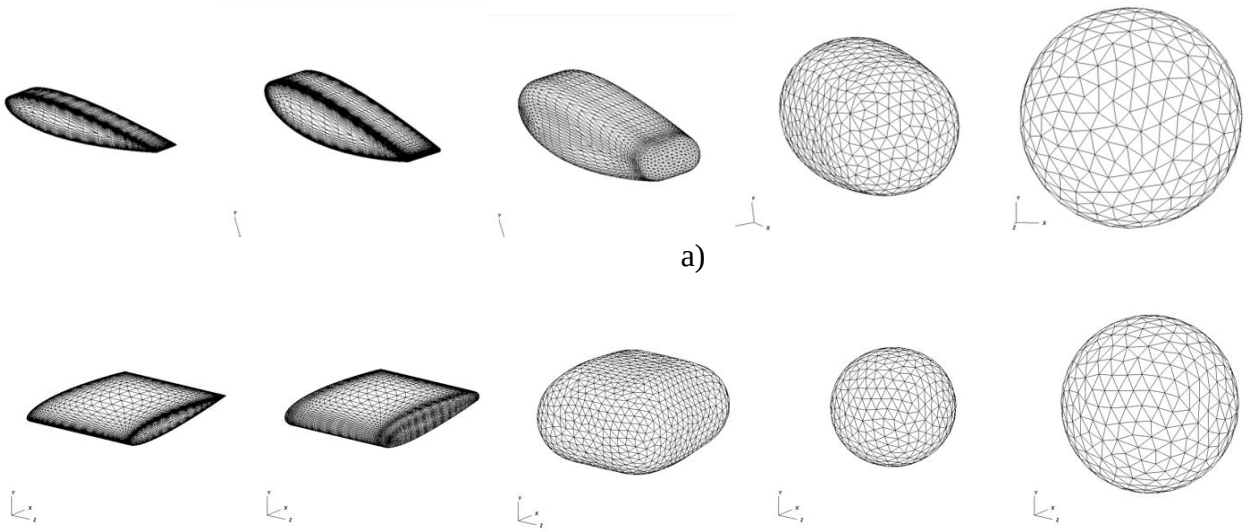
a) b)

c) d)

Figure 68 Details of NACA0012 2D domain mesh. a) the head zone; b) the tail zone; c) 3D boundary surface mesh with wall distance 0.0001; d) 3D boundary surface mesh with wall distance 0.001

5.4.2 3D NACA0012 Mesh results

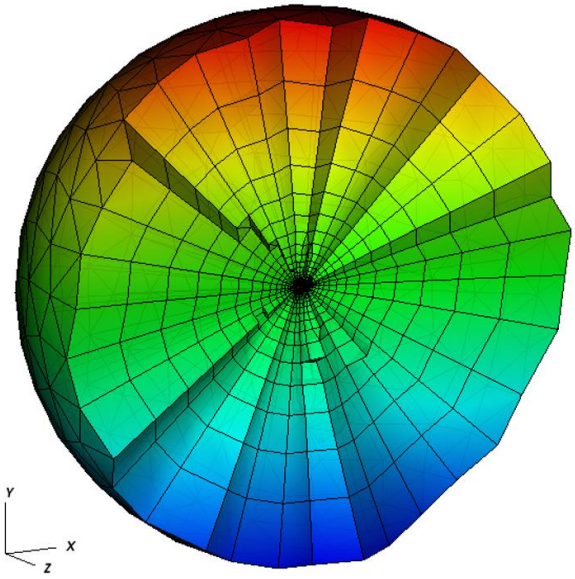
The 3D initial extrusion lengths of test cases are 1.3, 1.4 and 1.5. The total number of mesh layers is 45. Fig. 69 shows the shapes of different mesh layers. During the extrusion process, the total number of nodes decreases and the mesh deforms from anisotropic to isotropic, which implies that the neighboring edge springs and the tension springs are functioning as desired.



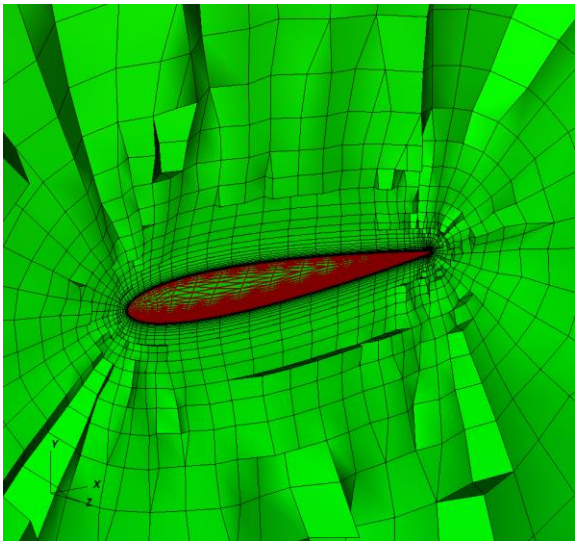
a)

b)

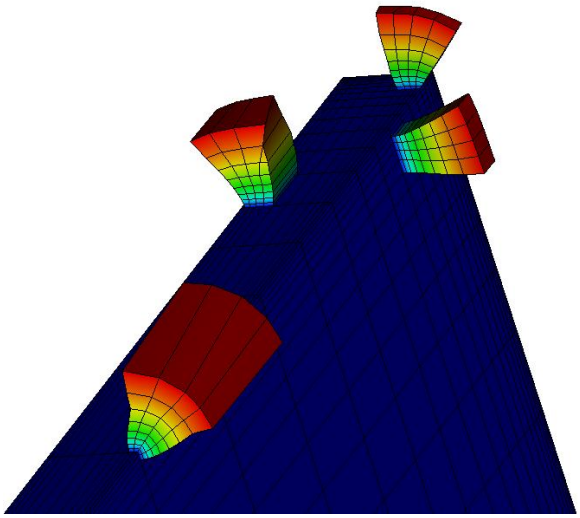
Figure 69 The deformation of the mesh layer. a) with the first extrusion length 0.0001; b) with the first extrusion length 0.001. From left to right, the mesh layer index is 10, 20, 30 and 45



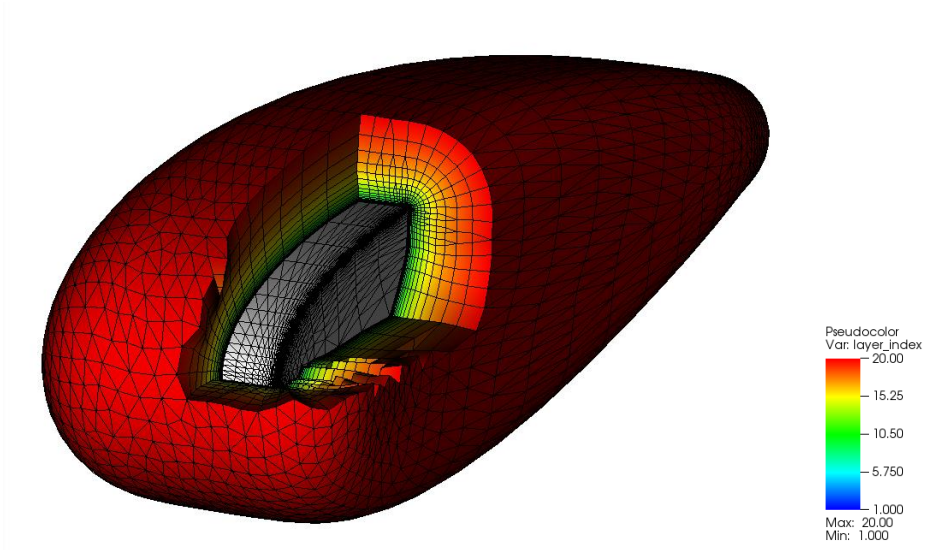
a)



b)



c)

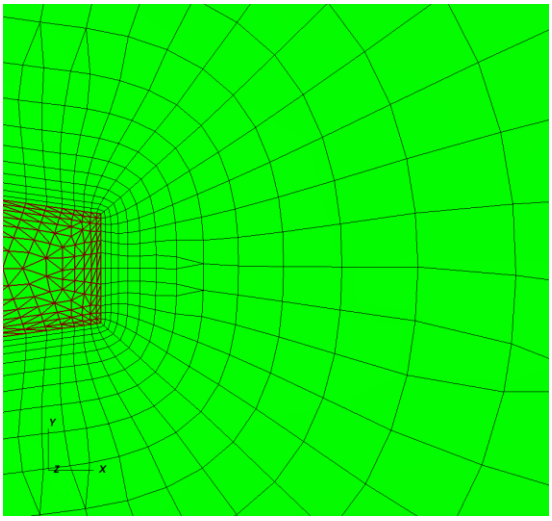


d)

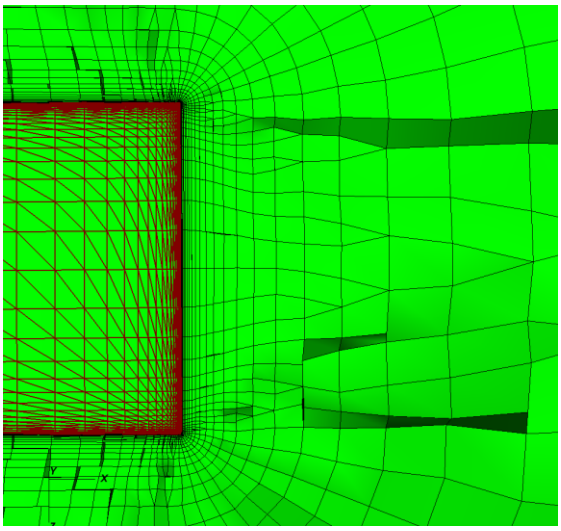
Figure 70 3D volume mesh with growth ratio 1.4. a) The whole view of the mesh cut through plane Z=0; b) the close view of the mesh cut through plane Z=0; c) The "pipes" extrude from boundary surface mesh. d) The 3D domain mesh with 20 mesh layers

Fig. 70 shows several mesh details. Fig. 70a) is a whole view of the mesh structure and Fig. 70b) is a closer view. The mesh in a) and b) are generated from a pure triangular boundary

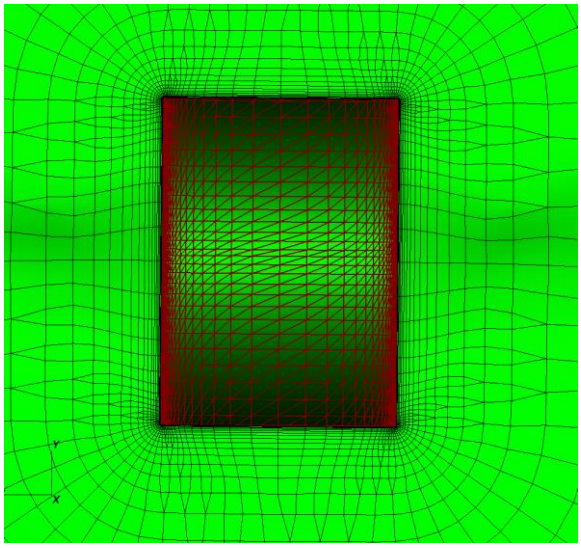
mesh with a first extrusion length 0.0001. The mesh structure is like pipes extruding from the boundary surface, as shown in Fig. 70c). Fig. 70c) shows the contour of the index of mesh layers, where the boundary mesh is a mixed-elements one. It illustrates how the pipes grow and how the mesh elements adjust their shapes and extruding direction by the forces of the springs.



a)

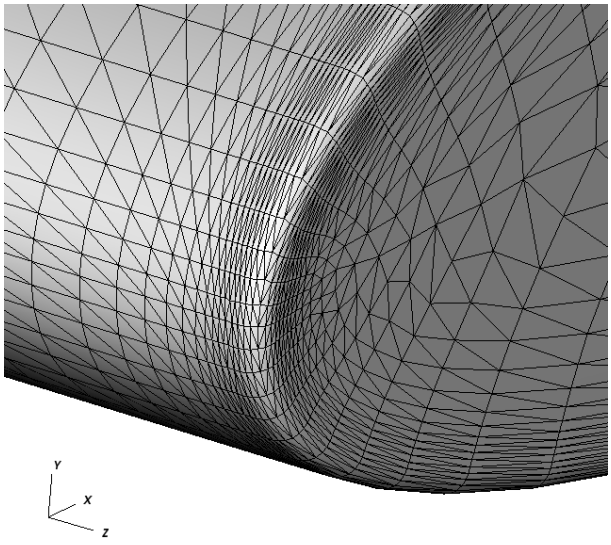


b)

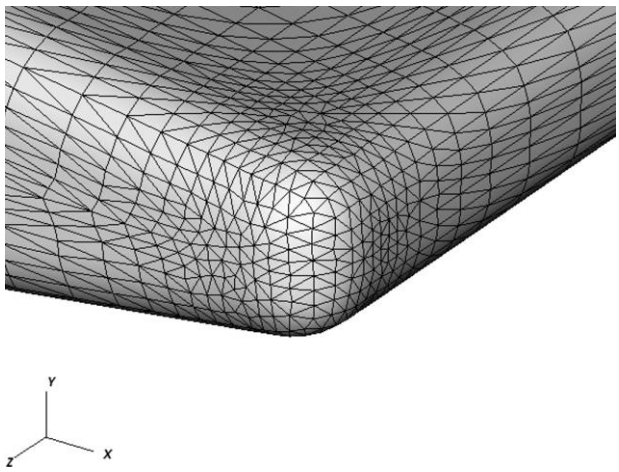


c)

Figure 71 Close view of the three views of the NACA0012 mesh with growth ratio 1.4. a) Z=0 slice; b) Y=0 slice; c) X=0.5 slice



a)



b)

Figure 72 Close view of the head and the tail zone of NACA0012 of the 10th mesh layer. a) the head zone; b) the tail zone

Fig. 71 shows a close view of three views of the mesh. These slices look similar to the 2D meshes generated by the previous 2D method. By force of springs, the extruding directions of nodes adjust layer by layer to fill in the empty corner space. As shown in Fig. 71, the neighboring edges collapse (break) for an efficient distribution of nodes, which benefits the solver. Fig. 72 shows the head and tail zone of the mesh with a first extrusion length 0.0001 in the 10th mesh layer. It illustrates that as the mesh layers increase, the node distribution is smoother but still can keep the anisotropic zones, which matches the mechanism behind Spring-Field model. Table 5 shows the neighbor triangle distribution based on the max angle in each triangle. The neighbor triangle is defined as the triangles created layer by layer which are the child triangles of the input boundary surface triangles. Note all data shown in Table 5, 6 and 7 come from the pure tetrahedral test cases and the total number of mesh layers is 45. The first value after the case index in Table 5, 6 and 7 is the first extrusion length of the input boundary surface mesh. The second value after the case index in Table 5, 6 and 7 is the first extrusion length of the 3D volume mesh.

Table 5 Triangle distribution in terms of max angle alpha

Grid case name	Total number of triangles	max $\alpha \in [60^{\circ}, 90^{\circ}]$	max $\alpha \in (90^{\circ}, 125^{\circ}]$	max $\alpha \in (125^{\circ}, 140^{\circ}]$	max $\alpha \in (140^{\circ}, 180^{\circ}]$
Case 1: 1e-3,1e-3	144632	89743	54886	3	0
Case 2: 1e-4,1e-4	237632	124395	113191	46	0
Case 3: 1e-4,1e-5	340382	164218	176122	42	0

Table 6 Extrusion edge distribution in terms of deviation angle alpha

Grid case name	Total number of extrusion edges	$\alpha \in [0^{\circ}, 10^{\circ}]$	$\alpha \in (10^{\circ}, 20^{\circ}]$	$\alpha \in (20, 30^{\circ}]$	$\alpha \in (30^{\circ}, 180^{\circ}]$
Case 1: 1e-3,1e-3	75858	41844	43441	125	0
Case 2: 1e-4,1e-4	124697	71930	72572	136	0
Case 3: 1e-4,1e-5	176030	122605	79660	142	0

Table 6 shows the extrusion edge distribution based on the deviation angle defined as the angle between the ideal extruding direction of one extrusion edge and its real extruding direction. Table 7 shows the tetrahedra distribution based on the max dihedral angle. The pure tetrahedra

mesh is obtained based on the mesh topology adjustment method 1 described in Section 5.2.3. There is no angle larger than 160 degrees in Table 7. Since the ideal extruding direction of each extrusion edge is a weighted average normal of its surround facets, Table 5 and 6 reflect the good mesh quality of the mixed-elements mesh because these features of the pure tetrahedral meshes in Table 5 and 6 are the same as the ones of the matched mixed-elements meshes. However, based on Table 7, the strategy to obtain tetrahedra could be further developed to generate tetrahedra with better quality, such as using the method mentioned in [71].

Table 7 Tetrahedra distribution in terms of max dihedral angle alpha

Grid case name	Total number of tetrahedra	max $\alpha \in [60^{\circ}, 90^{\circ}]$	max $\alpha \in (90^{\circ}, 130^{\circ}]$	max $\alpha \in (130^{\circ}, 150^{\circ}]$	max $\alpha \in (150^{\circ}, 160^{\circ}]$
Case 1: 1e-3,1e-3	421455	31511	389020	921	3
Case 2: 1e-4,1e-4	693644	37586	654989	1062	7
Case 3: 1e-4,1e-5	1001808	35314	963622	2866	6

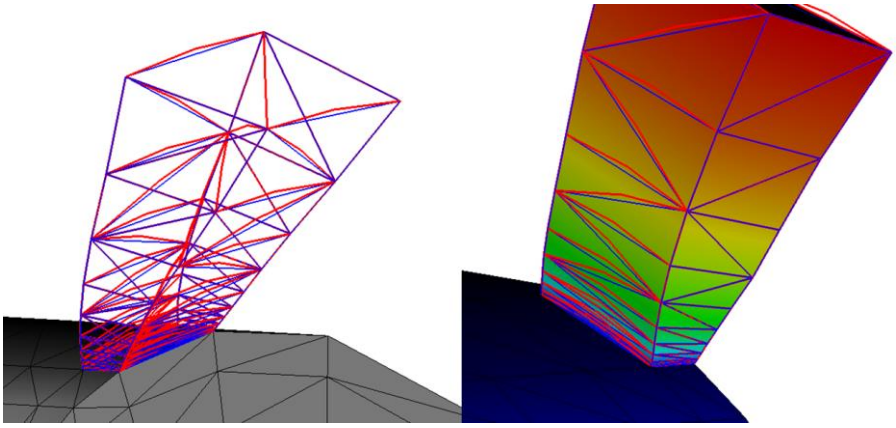


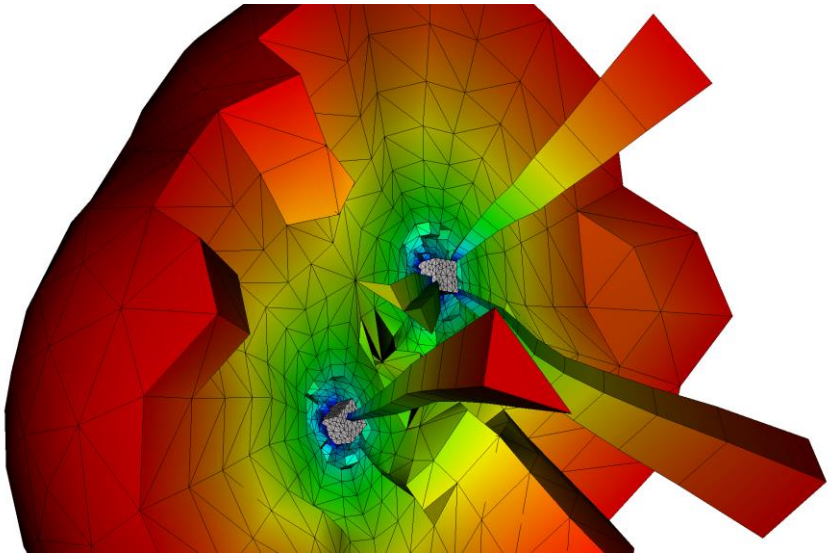
Figure 73 The curved quadratic mesh pipe structure

The curved quadratic mesh pipe structure is shown in Fig. 73. The curved mesh is generated by deforming the linear mesh layer by layer along this pipe structure. Since the mesh shape is similar to the linear one, the curved 3D mesh results are not emphasized in this mesh results section.

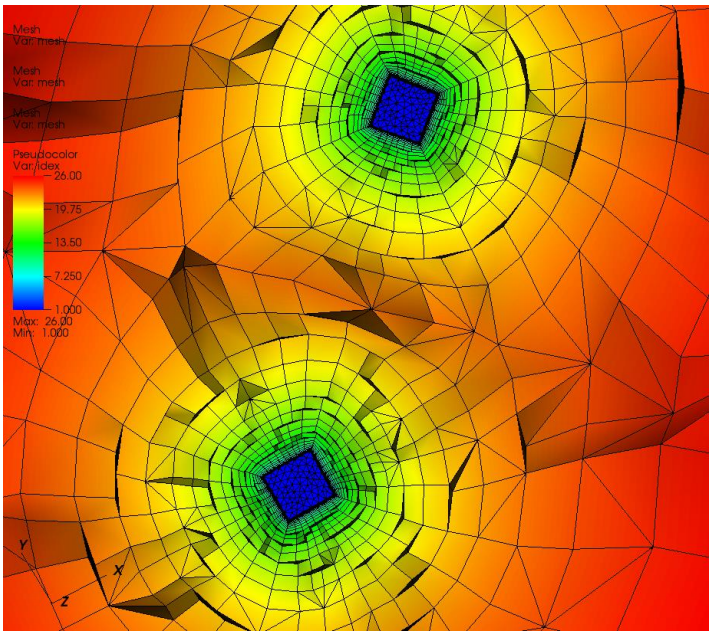
5.5 Mesh Results with Front Closing

With the 3D front-closing part added into the 3D mesh generation, now the method can handle some 3D mesh cases with multiple objects. Fig. 74 shows the meshes in the two-cubes case and Fig. 75 shows the 30P30N airfoil case. For the 30P30N airfoil case, the growth ratio is 1.3. The ideal distance of the first extrusion from boundaries is 0.000045. The total number of extruded mesh layers is 40. These figures illustrate that this mesh generation method can provide mixed-elements and pure tetrahedral meshes for the solver. In Fig. 75, the boundary layer mesh shows a strong perpendicular and anisotropic character, which benefits the numerical solver. In the 30P30N airfoil case, the max dihedral angle in the pure tetrahedral mesh is 165 degrees. In terms of

tetrahedral mesh quality, although it is acceptable for the solver, the method has plenty of room for improvement.

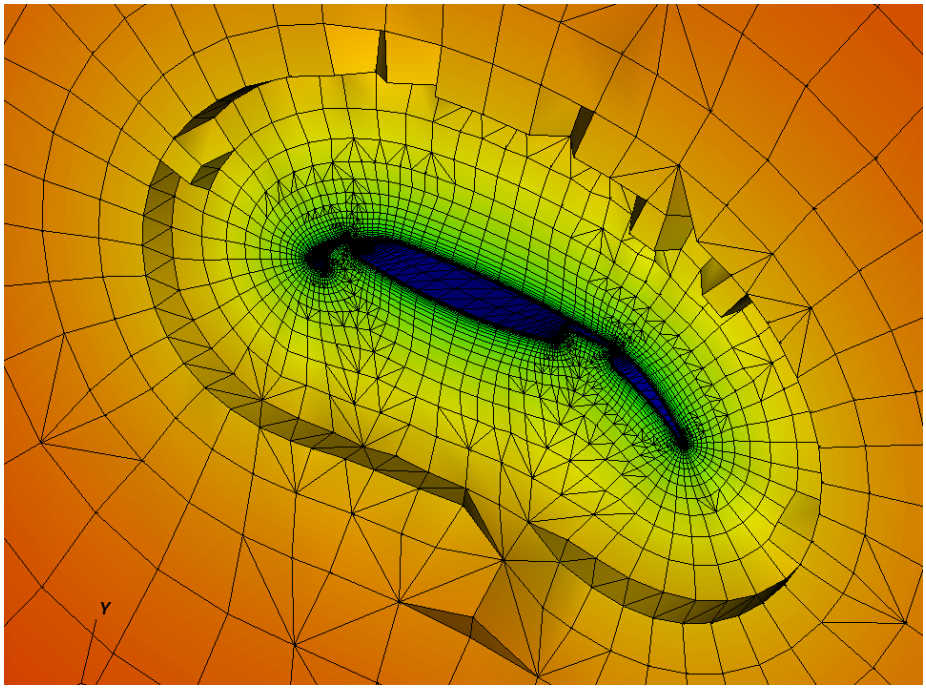


a)

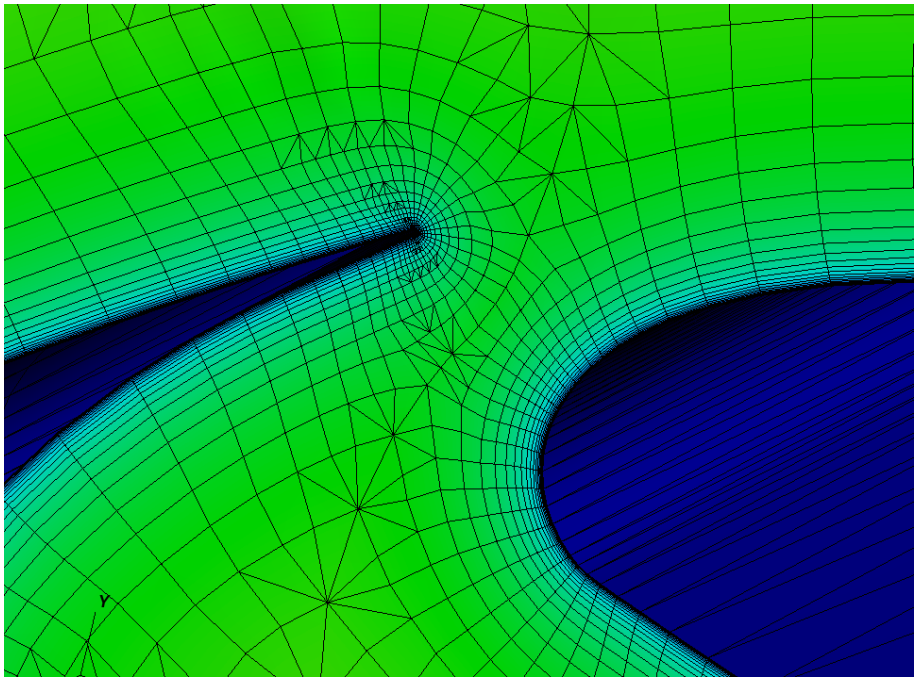


b)

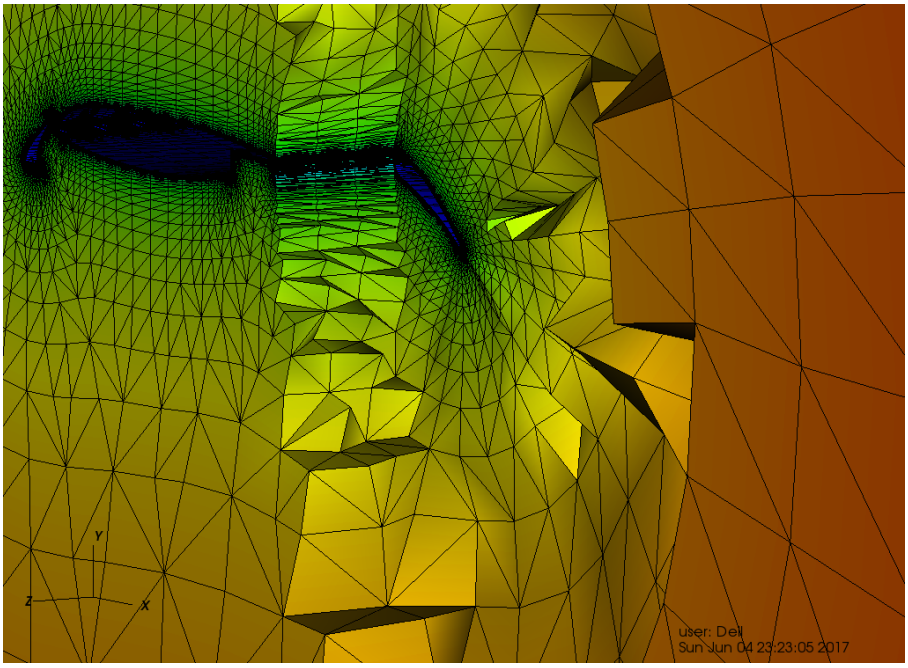
Figure 74 Two-cubes case. a) The tetrahedral mesh with small total number of boundary triangles; b) The mixed-element mesh with larger total number of boundary triangles



a)



b)



c)

Figure 75 30P30N airfoil case. a) a close view 1 of the mixed-elements mesh; b) a close view 2 of the mixed-elements mesh; c) a close view of the pure tetrahedral mesh

Table 8 shows the total number of different types of mesh cells in the mixed-elements 30P30N airfoil case and the pure tetrahedral case. The total number of mesh cells for the mixed-elements mesh is around 250000 and the total number of mesh cells for the pure tetrahedral mesh is around 660000. The mixed-element mesh will benefit the numerical solver by providing fewer mesh cells while covering the same space. It also illustrates that the mesh topology adjustment method described in Section 5.2.3 works. But this mesh topology adjustment method and the method used for 3D front closing are not yet robust. A change of the input coefficients such as growth ratio may lead to a failure of the mesh generation in the front-closing area. Hence in the future a better method of 3D front closing could be provided along with a more robust method to

transfer the non-standard type of mesh cells to standard types. For the current 3D front-closing and mesh topology adjustment methods used in this section, there are too many special situations to be taken into consideration. Hence the improvement of robustness is very difficult if following the current method of 3D front closing and the method of mesh topology adjustment.

Table 8 The number of mesh cells in different type

Grid case	Number of tetrahedra	Number of pyramid	Number of prism	Number of tetrahedra in the pure tetrahedral mesh
30P30N airfoil	45875	6911	199603	658506

5.6 CFD Results

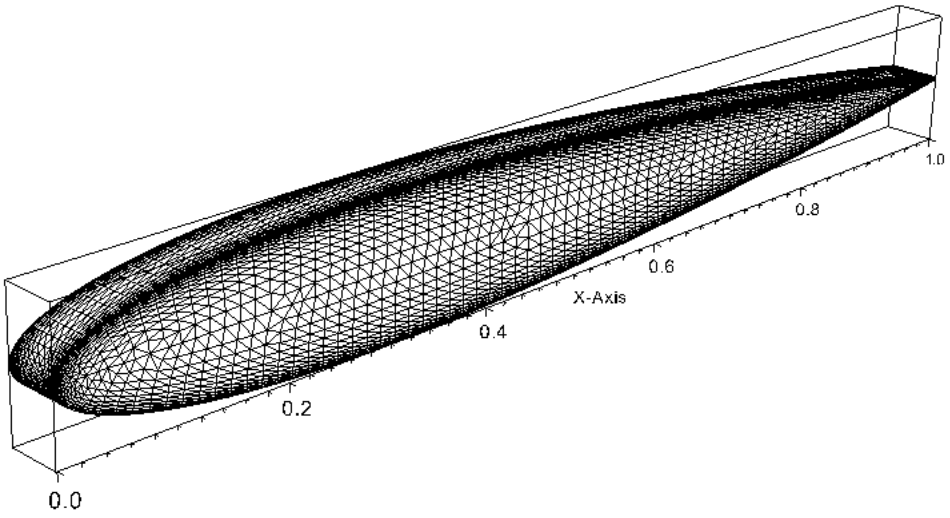


Figure 76 The 3D NACA0012 airfoil boundary surface mesh used for the solver

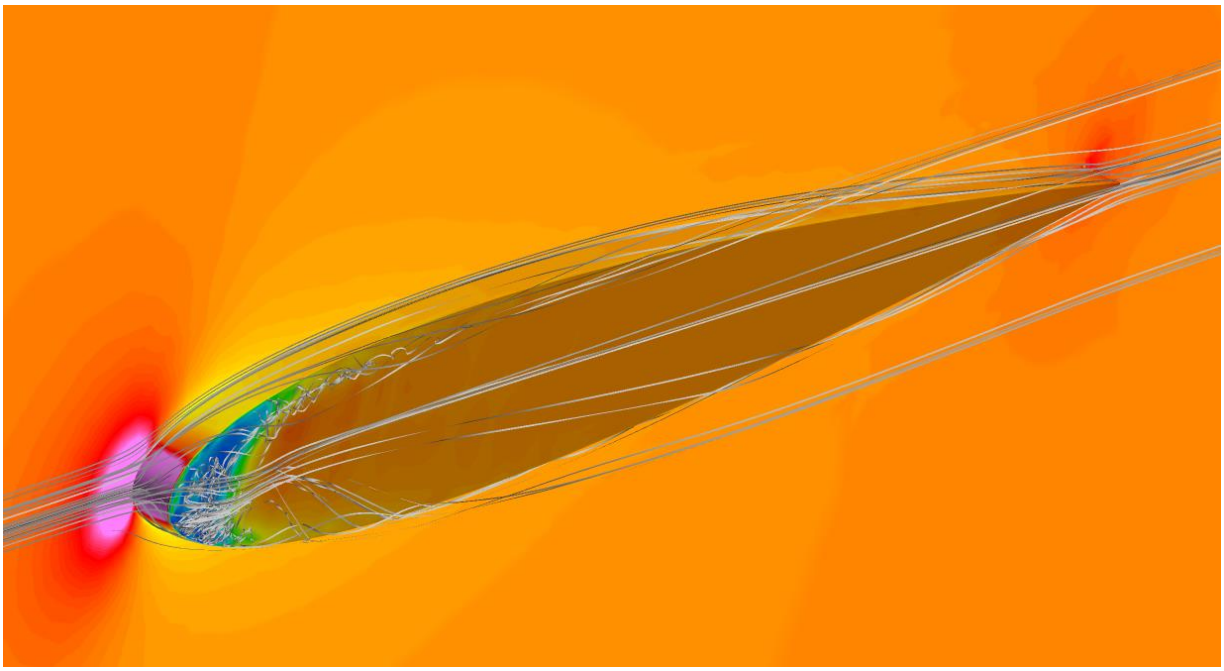
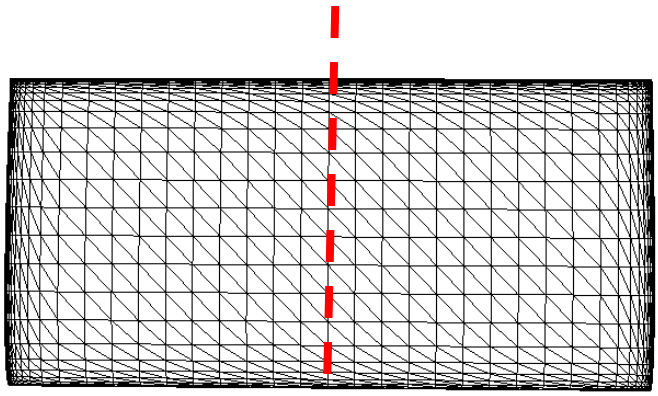


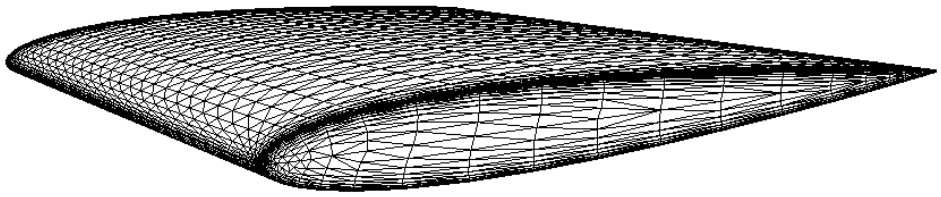
Figure 77 Pressure distribution and streamlines around the airfoil

The mesh generation technique described in this paper is examined by a finite element solver. The flow solver is based on a stabilized finite element method (Streamline Upwind Petrov Galerkin, or SUPG), which solves for steady and unsteady flow problems using the compressible Navier-Stokes equations and the Spalart-Allmaras turbulence model [67]. Fig. 76 shows the triangular mesh as the input boundary mesh for generating the pure tetrahedral mesh used in the simulation. The first extrusion length of the 2D airfoil and the first extrusion length of 3D boundary surface are both 0.001. The growth ratios are 1.3. The total number of nodes is 86791 for P1 and 681804 for P2. The total number of tetrahedra is 502083. The total number of mesh layers is 40 in the 3D extruding direction. Fig. 77 shows the pressure distribution of the flow field and the flow streamlines near the NACA0012 airfoil. The calculation is done with linear elements, and the flow condition is at a free stream Mach number 0.15, an angle of attack of zero degree, and a Reynolds number of six million based on the airfoil chord. It is shown that a three-dimensional flow pattern

is established, with strong tip vortices near the leading edge of the airfoil. Though the Jacobians are positive, the P2 solution did not converge. The reason is that at the side of the leading-edge zone of the airfoil, the unsteady flow character is too strong to obtain a steady solution. If the solver is set to unsteady, the P2 case converges. Since it is a higher-order simulation, the total number of the mesh elements could be less than the number of the mesh elements in the linear simulation. The boundary mesh is shown in Fig. 78. Based on that boundary mesh, the 3D domain mesh is generated with a growth ratio of 1.5. The *Cp* distribution is shown in Fig. 79. The points for calculating the *Cp* distribution are picked along the red dash line as shown in Fig. 78a). The Mach number is set to 0.15. The angle of attack is 0 degree. Reynolds number is set to 6 million. The Wall distance is set to 1e-5. Fig. 79 shows that the present numerical solutions agree well with the experimental data. It illustrates this 3D mesh generation method is promising for a higher-order scheme.



a)



b)

Figure 78 The boundary mesh used in the high-order unsteady simulation. a) Top view of the NACA0012 boundary mesh; b) a whole view of the NACA0012 boundary mesh

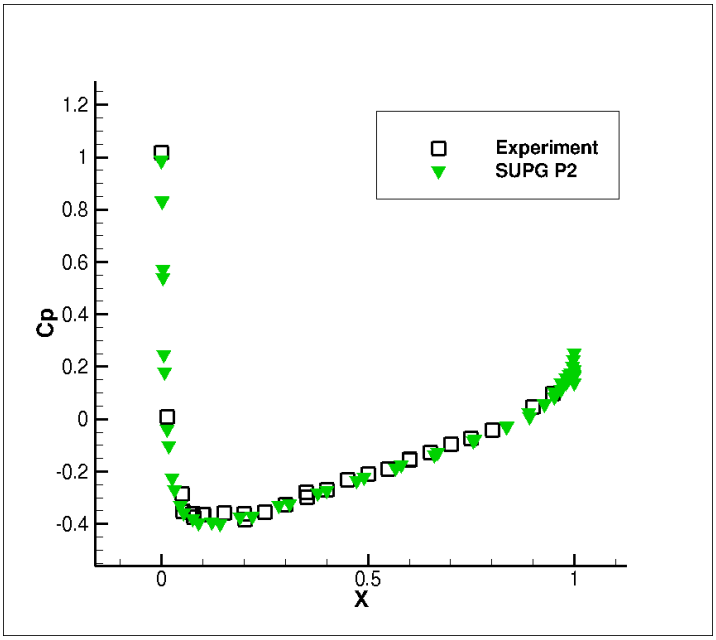


Figure 79 The *Cp* distribution of the 3D NACA0012 airfoil

CHAPTER 6

CONCLUSION

6.1 SUMMARY

In this dissertation, A mesh generation approach that allows for automatic generation of unstructured high-order meshes with the option to produce mixed types of elements (triangles and quadrilaterals) is presented. For the 2D cases, three main procedures, consisting of the boundary grid generation, the linear viscous grid generation, and the curved deformation, are performed in sequence. A new force model referred to as Spring-Field is presented and employed in the viscous domain-mesh generation process. This proposed mesh generation approach is capable of handling complex domain boundaries while maintaining good mesh quality in the transition areas between the boundary layer viscous mesh and the domain interior mesh. In addition, a deformation control process is supplemented to provide an automatic switch from non-curved to curved grids, depending on necessity. It is also noted that the cost of generating a mesh with an order higher than quadratic is at the same level as lower-order cases. Numerical analysis shows that the use of a cubic mesh in conjunction with a high-order discontinuous Galerkin scheme demonstrates promising results for complex geometry configurations in resolving the flow field as well as revealing turbulent boundary layer features. This mesh generation method has been extended to 3D cases. The results show that the layered “pipe-like” mesh structure, the force model Spring-Field and the Vector-Adding method is extendable to 3D and can produce good mesh quality.

This mesh generation method can also be used for the mesh movement purposes. With the adjustments of the method, the approach can not only handle linear mixed-element mesh generation with complex geometry but also has the capability to manage mesh movement or mesh morphing. Furthermore, with help from the Vector-Adding method for these pipe-like structure linear meshes, now both high-order viscous mesh generations and movements can be handled by the method. The test cases of large mesh motion and morphing show promising results.

6.2 Recommendations for Future Work

In the future research, a more efficient design of the spring model could be provided to further improve the mesh quality in the 2D mesh movement cases. 3D mesh movement method could be developed based on the 3D mesh generation method and the 2D mesh movement strategy.

This mesh generation method has been tested for the optimization of an airfoil in the framework of a high order turbulent flow solver. However, due to the high sensitivity of the whole domain mesh to boundary perturbations, the sensitivity derivatives cannot be calculated precisely enough for optimization in high-order cases. Future work could be also concentrated on this obstacle.

For the 3D cases, the improvement of robustness of this mesh generation tool is needed. Several parts need to be improved. First, a more robust front-closing strategy could be developed to handle not only the front-nodes merging but also the change of the mesh topology to merge front-edges. Second, a more robust strategy to obtain a standard mixed-elements mesh from the nonstandard mixed-elements mesh could be developed. Finally, Spring-Field could be further

developed to handle the mixed-elements boundary surface mesh (triangles and quadrilaterals) as the input to provide 3D volume mesh.

REFERENCES

[1] J. A. George, "Computer implementation of the finite element method," DTIC Document1971.

[2] S. Lo, "A new mesh generation scheme for arbitrary planar domains," *International Journal for Numerical Methods in Engineering,* vol. 21, pp. 1403-1426, 1985.

[3] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, "Adaptive remeshing for compressible flow computations," *Journal of computational physics,* vol. 72, pp. 449-466, 1987.

[4] S. Pirzadeh, "Unstructured viscous grid generation by the advancing-layers method," *AIAA journal,* vol. 32, pp. 1735-1737, 1994.

[5] R. V. Garimella and M. S. Shephard, "Boundary layer mesh generation for viscous flow simulations," *International Journal for Numerical Methods in Engineering,* vol. 49, pp. 193-218, 2000.

[6] F. Alauzet and D. Marcum, "A closed advancing-layer method with changing topology mesh movement for viscous mesh generation," in *Proceedings of the 22nd International Meshing Roundtable*, 2014, pp. 241-261.

[7] K. Shimada, A. Yamada, and T. Itoh, "Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles," in *Proceedings of the 6th International Meshing Roundtable*, 1996, pp. 375-390.

[8] V. Parthasarathy and Y. Kallinderis, "Directional viscous multigrid using adaptive prismatic meshes," *AIAA Journal,* vol. 33, pp. 69-78, 1995.

[9] Y. Kallinderis, A. Khawaja, and H. McMorris, "Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries," *AIAA Journal,* vol. 34, pp. 291-298, 1996/02/01 1996.

[10] Y. Kallinderis, A. Khawaja, H. McMorris, S. Irmisch, and D. Walker, "Hybrid Prismatic/Tetrahedral Grids for Turbomachinery Applications," in *Proceedings of 6th International Meshing Roundtable*, 1997, pp. 21-31.

[11] Y. Ito and K. Nakahashi, "Improvements in the reliability and quality of unstructured hybrid mesh generation," *International Journal for Numerical Methods in Fluids*, vol. 45, pp. 79-108, 2004.

[12] S. Fotia and Y. Kallinderis, "Quality index and improvement of the interfaces of general hybrid grids," *Procedia Engineering*, vol. 82, pp. 416-427, 2014.

[13] C. T. Druyor, "An adaptive hybrid mesh generation method for complex geometries," M.S. Thesis, Computational Engineering Dept., University of Tennessee at Chattanooga, Chattanooga, 2011.

[14] S. L. Karman, "SPLITFLOW - A 3D unstructured Cartesian/prismatic grid CFD code for complex geometries," presented at the 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, 1995.

[15] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, *et al.*, "Adaptive mesh generation for curved domains," *Applied Numerical Mathematics*, vol. 52, pp. 251-271, 2005.

[16] S. Sherwin and J. Peiró, "Mesh generation in curvilinear domains using high ‐ order elements," *International Journal for Numerical Methods in Engineering*, vol. 53, pp. 207-223, 2002.

[17] P.-O. Persson and J. Peraire, "Curved mesh generation and mesh refinement using lagrangian solid mechanics," in *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, 2009.

[18] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, *et al.*, "Adaptive mesh generation for curved domains," *Applied Numerical Mathematics*, vol. 52, pp. 251-271, 2005.

[19] J.-F. Remacle, T. Toulorge, and J. Lambrechts, "Robust untangling of curvilinear meshes," *Proceedings of the 21st International meshing roundtable*, pp. 71-83, 2013.

[20] L. Wang, W. K. Anderson, J. T. Erwin, and S. Kapadia, "Discontinuous Galerkin and Petrov Galerkin methods for compressible viscous flows," *Computers & Fluids*, vol. 100, pp. 13-29, 2014.

[21] Y. Ito and K. Nakahashi, "Unstructured Mesh Generation for Viscous Flow Computations," in *IMR*, 2002, pp. 367-377.

[22] R. Jain and T. J. Tautges, "PostBL: Post-mesh boundary layer mesh generation tool," in *Proceedings of the 22nd International Meshing Roundtable*, ed: Springer, 2014, pp. 331-348.

[23] S. L. Karman, "Unstructured viscous layer insertion using linear-elastic smoothing," *AIAA journal*, vol. 45, p. 168, 2007.

[24] K. Shimada and D. C. Gossard, "Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing," in *Proceedings of the third ACM symposium on Solid modeling and applications*, 1995, pp. 409-419.

[25] K. Shimada, A. Yamada, and T. Itoh, "Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles," in *6th International Meshing Roundtable*, 1997, pp. 375-390.

[26] F. J. Bossen and P. S. Heckbert, "A pliant method for anisotropic mesh generation," in *5th Intl. Meshing Roundtable*, 1996, pp. 63-74.

[27] A. K. Michler, "Aircraft control surface deflection using RBF-based mesh deformation," *International Journal for Numerical Methods in Engineering*, vol. 88, pp. 986-1007, Dec 9 2011.

[28] A. L. Gaitonde, "A Dual-Time for Solution of the Unsteady Euler Equation," *Aeronautical Journal*, vol. 98, pp. 283-291, Oct 1994.

[29] A. Goswami and I. Parpia, "Grid restructuring for moving boundaries," in *10th AIAA Computational Fluid Dynamics Conference*, Honolulu, HI, 1991, pp. 696-704.

[30] E. Luke, E. Collins, and E. Blades, "A fast mesh deformation method using explicit interpolation," *Journal of Computational Physics*, vol. 231, pp. 586-601, 2012.

[31] J. T. Batina, "Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex-Aircraft Aerodynamic Analysis," *AIAA Journal*, vol. 29, pp. 327-333, Mar 1991.

[32] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne, "Torsional springs for two-dimensional dynamic unstructured fluid meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 163, pp. 231-245, Sep 21 1998.

[33] C. Degand and C. Farhat, "A three-dimensional torsional spring analogy method for unstructured dynamic meshes," *Computers & Structures*, vol. 80, pp. 305-316, Feb 2002.

[34] T. E. Tezduyar, M. Behr, S. Mittal, and A. Johnson, "Computation of unsteady incompressible flows with the stabilized finite element methods: Space-time formulations, iterative strategies and massively parallel implementations," presented at the ASME Pressure Vessels and Piping Division PVP, New York, 1992.

[35] R. Lohner and C. Yang, "Improved ALE mesh velocities for moving bodies," *Communications in Numerical Methods in Engineering,* vol. 12, pp. 599-608, Oct 1996.

[36] S. M. Shontz and S. A. Vavasis, "A robust solution procedure for hyperelastic solids with large boundary deformation," *Engineering with Computers,* vol. 28, pp. 135-147, Apr 2012.

[37] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," Royal Signals and Radar Establishment, Malvern, UK, Memorandum 4148, 1988.

[38] J. A. Witteveen and H. Bijl, "Explicit mesh deformation using inverse distance weighting interpolation," presented at the 19th AIAA Computational Fluid Dynamics Conference, San Antonio, Texas, 2009.

[39] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM National Conference*, 1968, pp. 517-524.

[40] C. S. Peskin, "Flow patterns around heart valves: a numerical method," *Journal of Computational Physics,* vol. 10, pp. 252-271, 1972.

[41] L. D. Zhu and C. S. Peskin, "Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method," *Journal of Computational Physics,* vol. 179, pp. 452-468, Jul 1 2002.

[42] S. Xu and Z. J. Wang, "An immersed interface method for simulating the interaction of a fluid with moving boundaries," *Journal of Computational Physics,* vol. 216, pp. 454-493, Aug 10 2006.

[43] M. Berger and M. Aftosmis, "Aspects (and aspect ratios) of Cartesian mesh methods," in *16th International Conference on Numerical Methods in Fluid Dynamics*, Arcachon, 1998, pp. 1-12.

[44] D. K. Clarke, H. A. Hassan, and M. D. Salas, "Euler calculations for multielement airfoils using Cartesian grids," *AIAA journal,* vol. 24, pp. 353-358, 1986.

[45] D. Dezeeuw and K. G. Powell, "An adaptively refined Cartesian mesh solver for the Euler equations," *Journal of Computational Physics,* vol. 104, pp. 56-68, Jan 1993.

[46] H. S. Udaykumar, W. Shyy, and M. M. Rao, "ELAFINT: A mixed Eulerian-Lagrangian method for fluid flows with complex and moving boundaries," *International Journal for Numerical Methods in Fluids,* vol. 22, pp. 691-712, Apr 30 1996.

[47] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, "An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries," *Journal of Computational Physics,* vol. 156, pp. 209-240, Dec 10 1999.

[48] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annual Review of Fluid Mechanics,* vol. 37, pp. 239-261, 2005.

[49] J. L. Steger, F. C. Dougherty, and J. A. Benek, "A chimera grid scheme," presented at the Proceedings of the Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Houston, 1983.

[50] J. A. Benek, P. G. Buning, and J. L. Steger, "A 3-D chimera grid embedding technique," presented at the 7th Computational Physics Conference, Cincinnati,OH, 1985.

[51] L. P. Zhang and Z. J. Wang, "A block LU-SGS implicit dual time-stepping algorithm for hybrid dynamic meshes," *Computers & Fluids,* vol. 33, pp. 891-916, Aug 2004.

[52] S. M. Mirsajedi, S. M. H. Karimian, and M. Mani, "A Multizone Moving Mesh Algorithm for Simulation of Flow Around a Rigid Body With Arbitrary Motion," *ASME Journal of Fluids Engineering,* vol. 128, pp. 297-304, 2005.

[53] T. Liu, L. Wang, S. L. Karman, and B. Hilbert, "Automatic 2D high-order viscous mesh generation by Spring-Field and vector-adding," presented at the 54th AIAA Aerospace Sciences Meeting, San Diego, CA, 2016.

[54] T. Liu, L. Wang, S. L. Karman, and C. B. Hilbert, "Automatic Two-Dimensional High-Order Viscous Mesh Generation by Spring Field and Vector Adding," *AIAA Journal*, vol. 55, pp. 3072-3084, 2017/09/01 2017.

[55] T. Liu and C. B. Hilbert, "High-order mesh movement with Spring-Field and vector-adding," in *Proceedings of 25th International Meshing Roundtable*, Washington, D.C., 2016.

[56] T. Liu, W. Lin, and C. B. Hilbert, "Automatic high-order mesh movement with Spring-Field and vector-adding: 2D moving mesh with viscous layer," in *23rd AIAA Computational Fluid Dynamics Conference*, ed: American Institute of Aeronautics and Astronautics, 2017.

[57] T. Liu and C. B. Hilbert, "Automatic high-order mesh generation with Spring-Field and vector-adding: 3D domain mesh generation," in *55th AIAA Aerospace Sciences Meeting*, 2017, p. 0584.

[58] P. W. Fackler, "Physics-based point placement by particle dynamics simulation," 2013.

[59] J. BATINA, "Unsteady Euler airfoil solutions using unstructured dynamic meshes," *AIAA journal*, vol. 28, pp. 1381-1388, 1990.

[60] J. Teran, N. Molino, R. Fedkiw, and R. Bridson, "Adaptive physics based tetrahedral mesh generation using level sets," *Engineering with computers*, vol. 21, pp. 2-18, 2005.

[61] A. Smirnov, "Tool-assisted mesh generation based on a tissue-growth model," *Medical and Biological Engineering and Computing*, vol. 41, pp. 494-497, 2003.

[62] D. McLaurin and S. M. Shontz, "Automated edge grid generation based on arc-length optimization," in *Proceedings of the 22nd international meshing roundtable*, ed: Springer, 2014, pp. 385-403.

[63] M. S. Ebeida, R. L. Davis, and R. W. Freund, "A new fast hybrid adaptive grid generation technique for arbitrary two ‐ dimensional domains," *International Journal for Numerical Methods in Engineering*, vol. 84, pp. 305-329, 2010.

[64] A. S. Jonathon, "Hybird Grids," in *Handbook of Grid Generation*, J. Thompson, Soni, B. K., and Weatherill, N. P., Ed., ed Boca Raton: CRC Press, 1998.

[65] C. Liu, J. C. N. III, W. K. Anderson, and B. R. Ahrabi, "Three-Dimensional Dynamic Overset Method for Stabilized Finite Elements," in *22nd AIAA Computational Fluid Dynamics Conference*, 2015, p. 3423.

[66] R. M. Hicks and P. A. Henne, "Wing design by numerical optimization," *Journal of Aircraft*, vol. 15, pp. 407-412, 1978.

[67] J. T. Erwin, W. K. Anderson, S. Kapadia, and L. Wang, "Three-dimensional stabilized finite elements for compressible Navier–Stokes," *AIAA journal*, vol. 51, pp. 1404-1419, 2013.

[68] W. K. Anderson and D. L. Bonhaus, "Airfoil design on unstructured grids for turbulent flows," *AIAA journal*, vol. 37, pp. 185-191, 1999.

[69] W. Lin, W. K. Anderson, J. Newman, and X. Zhang, "Shape Optimization for Two-Dimensional Acoustic Metamaterials and Phononic Crystals with a Time-Dependent Adjoint Formulation," in *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, San Diego, California, 2016.

[70] M. Kennel. (2004). https://arxiv.org/abs/physics/0408067. Available: https://github.com/jmhodges/kdtree2/blob/master/src-f90/kdtree2.f90

[71] C. B. Hilbert, "Tetrahedral mesh optimization and generation via topological transformations and gradient based node perturbation," Ph.D. Dissertation, Computational Engineering Dept., University of Tennessee at Chattanooga, Chattanooga, 2015.

VITA

Tuo Liu completed the Bachelor of Science degree and the Master of Science degree in Aerospace Engineering at the Beihang University (previously known as Beijing University of Aeronautics and Astronautics) in 2008 and 2011 receptively. After graduating, he worked for one year at the Sinovel Wind Group Company as an aerodynamic engineer. In 2012, he decided to pursue his Ph.D. in Computational Engineering at the SimCenter, the University of Tennessee at Chattanooga and earned his degree in 2017.