8-26-2022

Automatic Generation of Near-Body Structured Grids

Yuyan Hao
Syracuse University, yhao08@syr.edu

Abstract

Numerical grid generation has been a bottleneck in the computational fluid dynamics process for a long time when using the structured overset grids. Many current structured overset grid generation schemes like the hyperbolic grid generation method require significant user interaction to generate good computational grids robustly. Other grid generation schemes like the elliptic grid generation method take a significant amount of time for grid calculation, which is not desirable for computational fluid dynamics.

Herein a new grid generation method is presented that combines the hyperbolic grid generation scheme with the elliptic grid generation scheme that uses Poisson's equation. The new scheme builds upon the strengths of the different techniques by first applying hyperbolic grid generation, which is very fast but sometimes fails in strong concavities, and then using elliptic grid generation to locally fix the problems where hyperbolic grid generation results are not acceptable for computational fluid dynamics calculation. The new technique is demonstrated in various examples that are known to cause problems for either hyperbolic or elliptic grid generation when applied alone. The computational speed of the combined scheme grid generation is also exanimated by comparing the results with hyperbolic and elliptic grid generation.

The combined grid generation scheme is further implemented in Engineering Sketch Pad to get useful near-body structure grids based on the geometry of the model. Attributes in Engineering Sketch Pad are used to define the places where the surface and volume grids should be generated, while the tessellations are used to locate and project grid generation results and therefore boost grid generation speed. Three cases are tested

to illustrate the implementation of the combined grid generation scheme in Engineering Sketch Pad.

Automatic Generation of Near-Body Structured Grids

by

Yuyan Hao

B.S., Shandong University, 2014

M.S., Syracuse University, 2016

Dissertation

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical and Aerospace Engineering.

Syracuse University

August 2022

Content

List of Figures

1. Introduction

Numerical grid generation has been an essential tool in computational fluid dynamics (CFD), and it is one of the main technical challenges identified in the CFD Vision 2030 Study [1]. Generally, the grids can be divided into two main types: structured grids and unstructured grids. A structured grid means that the grid cells are well ordered, in which the cells are quadrilaterals for two dimensions or cuboids for three dimensions. Single structured grids are usually used when the geometry is simple, like a turbine blade or a wing on a fuselage in the flow field. An unstructured grid means that the grid cells are irregularly connected, such as triangles for two dimensions or tetrahedra for three dimensions. Unstructured grids have the advantage that they can be made to conform to nearly any desired geometry, and therefore they are used to deal with situations with complex geometry [2]. However, composite structured grid schemes, including overset grids scheme [3], can also be applied to resolve complex geometry or flow features, and there are several advantages to using overset grids:

- Overset grids consist of several structured grids, which means that the grids can be generated fast, and the grid points have good connectivity. Unstructured grids need more information to be stored and retrieved than structured grids, like the neighbor connectivity list of grid points and grid cells. Changing element types and sizes may also increase numerical approximation errors.
- Structured overset grids are usually orthogonal grids (grid lines intersect at a right angle [4]). An orthogonal grid offers significant advantages when solving partial differential equations and the simulations of computational fluid dynamics. Unstructured grids cannot be orthogonal grids.

- Overset grids are generated separately based on different parts of the geometry, which means that if any parameters of the geometry change or any parts of the geometry are moved or deleted, only relevant grids need to be recalculated, and other grids can be kept the same. For the unstructured grids, all grid points should be recalculated if such changes have been made.

This thesis will focus on generating near-body grids of the overset grids. Hyperbolic and elliptic grid generation schemes are the two methods that are widely used to generate near-body grids. A detailed literature review of hyperbolic is discussed in Chapter 2 and elliptic methods in Chapter 3.

1.1 Motivation

Overset grid generation is one method to simulate complex computational fluid flow problems. In an overset grid system, complex geometry is decomposed into several simple overlapping grids, including near-body grids and grids in the far-field. Information about flow variables is exchanged between these grids via interpolation, and some grid points may not be used in the solution. The boundary or fringe points of each block are in the interior of a neighboring block (or blocks). Values at these points can be acquired from the containing block(s). Pre-processors like PEGASUS 5 [5] or SUGGAR++ [6] are typically used for domain connectivity.

Here an example figure about overset grids is shown in Figure 1, where near-body grids are embedded around space shuttle, and the far-field grids are the Cartesian grids filled in the remaining areas. The near-body grids should be as orthogonal to the body as possible to decrease computational error when solving flow equations. Near-body grids should also be generated fast

in order to deal with the cases where the geometry of the model is changing. Current near-body grids techniques require significant user interactions, which requires users to have a strong background about grid generation to achieve good grids. Besides, there is no open-source code for generating overset grids. Current overset grid generation codes usually have restrictions and are private. Chimera Grid Tools (CGT) [7], which is provided by NASA, is a software package containing a variety of tools for solving complex configuration problems using the Chimera overset grid approach, including OVERGRID [8] and SURGRD [9] for surface grid generation and HYPGEN [10] for near-body volume grid generation. This software package can only be requested and used by U.S. citizens and permanent residents belonging to a U.S. organization. Other commercial computational fluid dynamics software also have their own grid generation code, but they are not published. Thus, an open-source code for overset grid generation is highly in demand for research purpose.

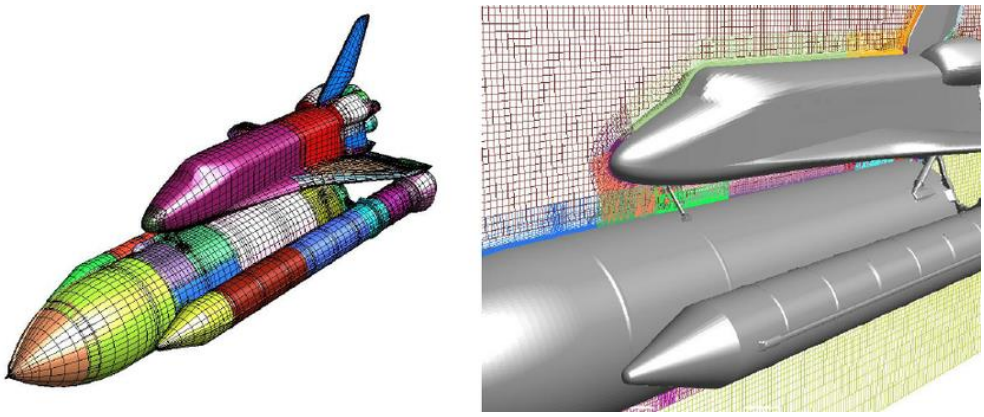


Figure 1: Overset grids on a space shuttle vehicle [11]

1.2 Objective

Based upon the brief introduction to the advantages and disadvantages of the structured overset grids, we can find that there are some critical factors for a good grid:

- Whether the grid lines are smooth, i.e., gridlines must not be folded or degenerate at any points or lines
- Whether the orthogonality can be controlled near boundaries where data collections are generally required to reduce calculation error
- Whether spacing can be controlled, especially for near-body or viscous region
- Whether the scheme is economical or time-consuming
- Whether the scheme needs manual work or not.

For the smoothness of grid lines, generating grid lines by solving partial differential equations is generally used. The hyperbolic scheme has the advantage that it is speedy, while the elliptic scheme has the advantage of no overlapping between grid lines. However, both schemes have some user-defined parameters, like outer boundaries of overset grids in the elliptic grid generation or smoothing terms in the hyperbolic grid generation.

The present method aims to find an intelligent way to combine the hyperbolic scheme with the elliptic scheme, which exploits their advantages. Furthermore, the present scheme automatically generates a whole near-body structure grid with the default setting of different kinds of parameters at the beginning, which does not require users to have a strong background of grid generation but can achieve a usable grid within a considerable time.

1.3 Roadmap

In this thesis, detailed descriptions of the hyperbolic grid generation and the elliptic grid generation are presented in Chapter 2 and Chapter 3. Both chapters start with the literature review and math manipulation of planar, surface, and volume grid generation. Several example

cases are tested and compared to show the strengths and weaknesses of these two grid generation techniques. Then the new combined grid generation scheme is introduced in Chapter 4. Several essential steps, including bad points detection, outer boundary correction, and connecting curve correction, are discussed in this chapter. Sample cases and comparisons of three grid generation schemes are also presented. Then the application of the combined grid generation scheme in Engineering Sketch Pad [12] is shown in Chapter 5. Several features in Engineering Sketch Pad are utilized to help automatic near-body grid generation, and three example cases are presented in this chapter. Conclusion and future work are discussed in Chapter 6.

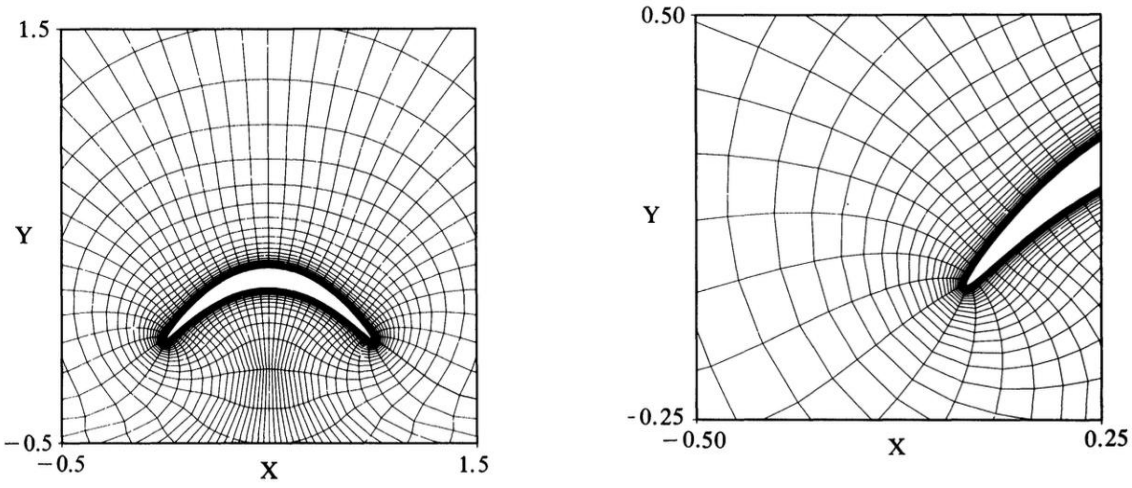
2. Hyperbolic Grid Generation

In hyperbolic grid generation, new mesh points are generated by propagating from a given level of points. The governing equations of hyperbolic grid generation are typically derived from orthogonality of the grid and grid cell size constraints. Local linearization of these differential equations allows a mesh to be generated by marching from a known state to the next. The total number of marching layers and the grid sizes at each layer can be prescribed based on the requirements of the specific application [13].

2.1 Introduction

The founders of the hyperbolic grid generation are Joseph L. Steger and Denny S. Chaussee [14]. In 1980, they published the earliest edition of the hyperbolic grid generation scheme. At that time, they started to use hyperbolic partial differential equations to generate a primary grid. They demonstrated that normal directions could be constructed from the initial distribution of points on the initial surface to an adjacent control plane or level line. Then these normal directions can be used to find the adjacent level line [15]. The grids could be generated by repeating these steps. The grid spacing could also be specified within this process. They also showed how hyperbolic partial differential equations were solved numerically by a linearization near a known state. It can be seen from Figure 2 the local orthogonality and spacing can be easily controlled using hyperbolic partial differential equations, compared with elliptic schemes mentioned above. Moreover, it satisfies the problems that the outer boundary is not known. They also came up with a method to calculate the area or volume of the grids accurately. However,

they ignored the most critical defects in the hyperbolic grid generation scheme — grid lines may overlap at a concave body curve.



(a) Grid detail near the body

(b) Grid detail near the leading edge

Figure 2: Viscous grid generated about highly cambered airfoil or turbine blade. [14]

After a few years, M.J. Berger and J. Oliger [16] came up with a method to generate adaptive mesh refinement for the hyperbolic scheme. It is the earliest version for adaptive grid generated by the hyperbolic scheme. They said the results of solving hyperbolic equations were often smooth and easily approximated over large portions of domains. Nevertheless, when the algorithm was applied to locally isolated internal regions with steep gradients, shocks, or body discontinuities, it is difficult to approximate those solutions numerically, which is very similar to what is discussed in the elliptic scheme. They adaptively used finer grids instead of the coarse ones generated by a standard hyperbolic scheme without any assumptions about the number or type of irregular regions. The method they used was called rectangles of arbitrary orientation. It has the advantages of allowing to approximately align coordinates with singular surfaces such as discontinuities and reducing the size of refined regions and the number of added mesh points. The values on bodies of finer grids are calculated by interpolation between coarser grids where

the refinement is embedded. Their method is significant to the further development by T. Tang and H. Tang [17] of adaptive grid generated by hyperbolic partial differential equations. However, some problems still need to be studied, including the best solution strategy for steady-state computations and data structures for component grids in different coordinate systems.

In 1985, J. L. Steger [18] developed the scheme of three-dimensional grids by solving hyperbolic partial differential equations. Steger extended his scheme for a three-dimensional grid generation problem. He derived the equations from the original hyperbolic partial differential equations and came up with the method to solve these equations numerically. It was the first time that approximate factorization was added into the hyperbolic scheme, which improved the computational speed greatly. After approximate factorization was introduced, the matrix can be solved by sequences of one-dimensional-like block tridiagonal systems. Moreover, Steger added a combination of fourth and second differences explicitly and implicitly into the basic algorithm to make it more stable. He also mentioned he had tried many dissipation terms, which is the most difficult problem in the hyperbolic scheme in the future. The algorithm opened a window in the grid generation scheme, but this algorithm was not suitable for all situations. When the body surface is discontinuous or when the user specified surface grid distribution is too irregular, the hyperbolic grid generator can fail, although it did well in simple continuous body shapes. Furthermore, user interaction cannot be avoided.

Four years later, in 1989, Steger [19] himself answered how to use a grid generation scheme by solving hyperbolic partial differential equations in complex body shapes. As mentioned before, when facing a problem with complicated body shapes, it could only be solved by using composite or unstructured grids. He came up with an extension of the procedure to use the hyperbolic scheme to generate semi-unstructured grids, which is the chimera overset grid

method. This method is one technique to generate grids over very complex geometry. It constructs a grid system made up of blocks of overlapping structured grids. The complex geometry can be decomposed into a geometrically simple overlapping grids system. The boundary conditions are exchanged between grid points by interpolating flow variables, and many grid points may not be used in the solution (hole points). In the Chimera overset grids scheme, each body component of a complex configuration generates mesh independently, and the composite grid is made up of the superposition of the original individual grids, which can be seen in Figure 3. These are the overset grids of a space shuttle vehicle made up of a shuttle orbiter, external tank, and solid rocket boosters. The individual grids are first connected by cutting out points that fall within another body and setting up interpolation links between cut-out-hole and outer boundaries. Then we can use an efficient structured grid flow solver to solve the flow equations on each grid and transport values on grid points across different grids. In this way, the advantage of grid generation by solving hyperbolic partial differential schemes has significantly been exploited.

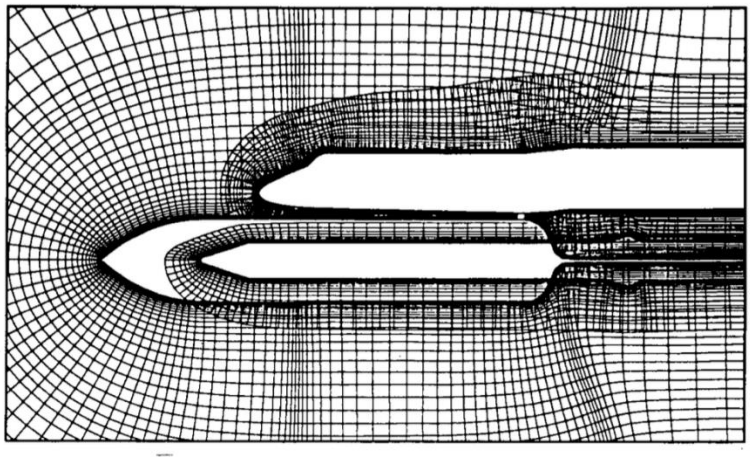


Figure 3: Overset grids on a space shuttle vehicle. [19]

In 1992, E.N. Ferry and C.J. Nietubicz [20] developed a grid generator that was interactive for projectile CFD. They gave the visual analysis, which is very important in using the hyperbolic grid generation scheme. In the previous hyperbolic solver program developed by Steger written in FORTRAN, an input data file was read, and a data file containing the final grid was produced. This file was like a notebook to record any user's initial setting on the grid. However, if any of those parameters required adjustment, the user had to change the input file and re-run the code. Furthermore, at that time, another program called DISSPLA [21] was used to view the grid to determine if it was acceptable to be used by the appropriate solver. But if someone used this program to generate grids, it was very time-consuming because the user could only view one part of the grid at a time. Ferry and Nietubicz developed a hyperbolic solver which was interactive and worked very efficiently. They combined the hyperbolic grid generator with Iris Silicon Graphics workstations. Adding user interactions through the graphics made the hyperbolic grid generation task much easier and quicker. The reason is that users could discover the parts needed to make modifications and see the results correspondingly so that the result could reflect whether those modifications were what the users needed.

In the same year, another critical researcher in the hyperbolic grid generation scheme, Chan [22], had made massive progress in developing the hyperbolic grid generation scheme. He made a couple of enhancements to Steger's original three-dimensional hyperbolic grid generation scheme [18]. Smoothing terms were critical to the hyperbolic grid generation scheme to avoid grid collapse when facing concave corners. Moreover, it was an essential task in the study of HYPGEN to generate a useful grid by using the hyperbolic scheme in a concave corner. Chan developed a way to represent all parameters that could affect the grid points in a concave corner. He introduced five parameters into the smoothing terms. Those parameters represented

the normal distance from the body surface, the distances between neighboring grid points, the angles between neighboring grid points, respectively, which will be discussed later in this thesis. In this way, the grids could be smoothed when facing different kinds of body shapes. However, users still need to adjust those parameters to make better grids when facing complex body shapes with certain concave corners. But at least there is a guideline about how to make the adjustments. Chan also developed a local treatment of severe convex corners. It provided extra robustness at convex corners by switching from solving the hyperbolic grid generation equations to some other equations at the convex corner point, like the implicit averaging scheme or prediction of the exact location of grid points in advance. Furthermore, Chan came up with a method to smooth the grids near a concave corner. He used a metric correction procedure to provide a satisfactory treatment of corner discontinuities in all but highly convex cases. However, the scheme would fail in some extreme cases, which is mentioned in the following sections.

A year later, Chan [10] wrote a user manual for the HYPGEN hyperbolic grid generator, and he developed an HGUI graphical user interface. In this user manual, it can be seen that there are several user-defined parameters in his scheme. Three of them are introduced below. The parameters TIMJ helps to spread out grid lines that might converge as the grid is matched out. The parameters ITSVOL controls the averaging of volumes between grid cells. Increasing ITSVOL value will increase the averaging and tend to spread out covering grid lines in tight concave corners but may have the undesirable effect of over-distorting the cell volumes at other places. SMU2 is the scale factor for the smoothing coefficient. If a user is not very familiar with the process of numerical grid generation or does not know much about numerical knowledge, it is tough to obtain a useful combination of parameters ultimately. An example of the hyperbolic grids on an orbiter is shown in Figure 4. Figure 4 (a) gives the original result of hyperbolic grid

generation, which has significant grid collapse due to the concave corner. Figure 4 (b) through (d) gives the grid generation results after tuning the parameters TIMJ, ITSVOL, and SMU2, respectively, and the results are much better after changing these parameters. However, the values of these parameters are not derived from the geometry of the body, and they may change for different cases. Users can only get the values by experience or testing for different values until a suitable grid for computational fluid dynamics is generated.

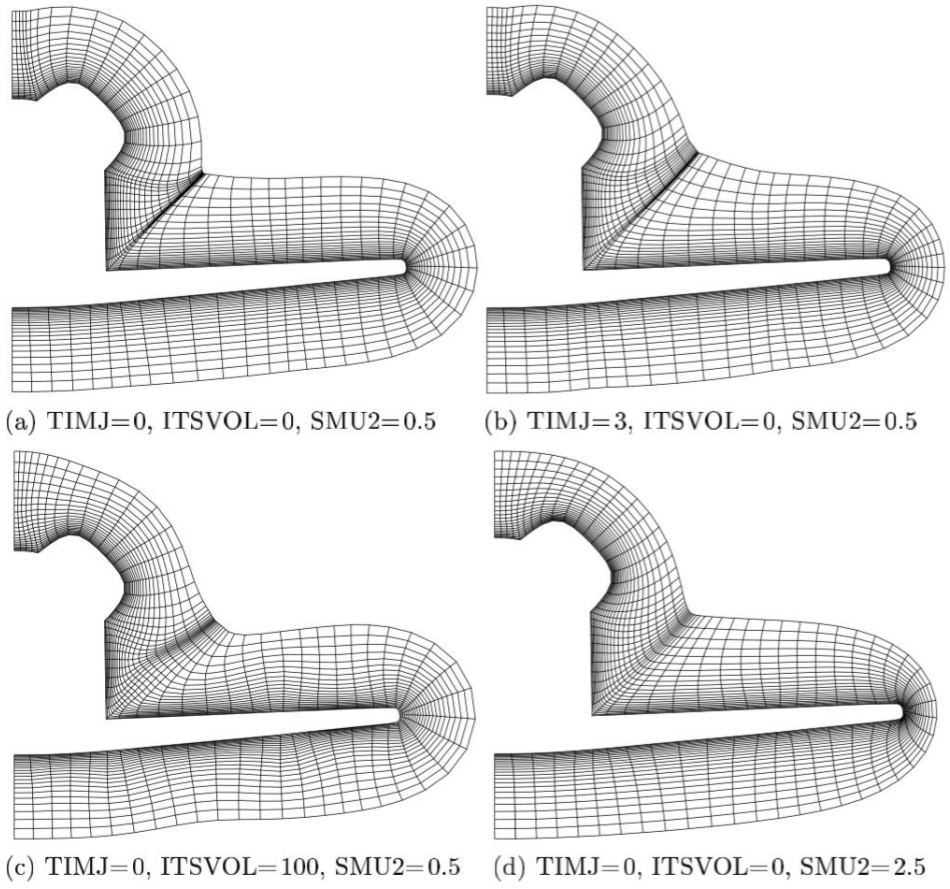
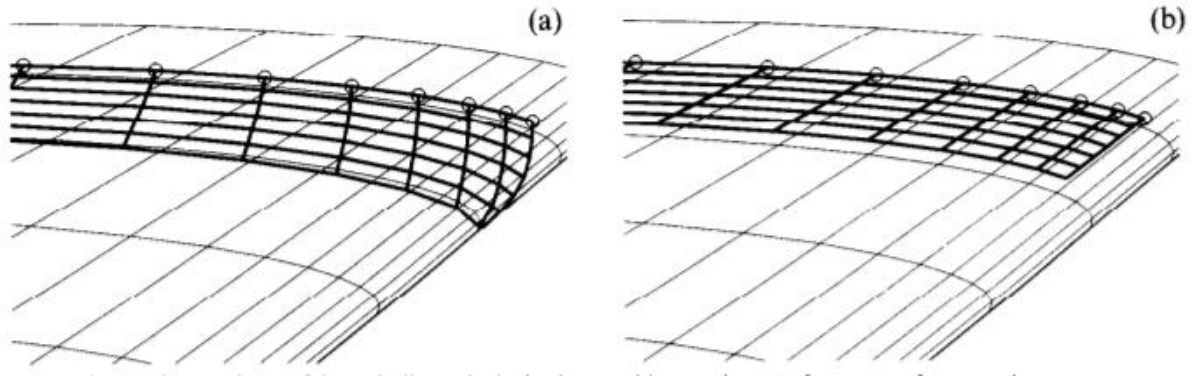


Figure 4: Hyperbolic grid generated on an orbiter. [10]

In 1995, Chan [9] developed a method to enhance the generality and robustness of Steger's scheme, and he also developed a code called SURGRD, which was the embryonic form of the software with the same name. Chan introduced a search and projection algorithm to

enhance the robustness of the grids. In Steger's scheme, the newly generated grid points are projected onto the reference surface after each grid marching step. Those projected points and the local surface normal directions of those points are used to perform the next grid marching step. But Chan assumed the reference surface to be a bilinear surface defined by a collection of panel networks. Each network has a sequent set of quadrilaterals. Sometimes it is necessary to march away from the body to a direction that is not orthogonal to the body surface, as shown in Figure 5. Under this condition, the original hyperbolic scheme cannot solve the problem, and the algebraic marching scheme can be used here. These techniques reduce the time for generating overset grids, and SURGRD code combines hyperbolic, algebraic, and elliptic methods on an overset surface grid generator, which can offer users a choice of an appropriate scheme when facing with different situations.



(a) Hyperbolic marching results in a grid that is orthogonal to the initial curve

(b) Algebraic marching results in a grid that is parallel to a family of curves defined by the surface panels.

Figure 5: Comparison of hyperbolic and algebraic marching options (reference surface panels are represented by thin lines, points on the initial curve are indicated by open circles, surface grids are represented by thick lines). [9]

Unlike other researchers who utilize finite-difference for hyperbolic grid generation calculation, H.A. Dwyer [23] came up with an alternative and extended presentation of the concepts used in hyperbolic grid generation using vector analysis. He presented a clear interpretation of the basic hyperbolic partial differential equations, shown in Figure 6.

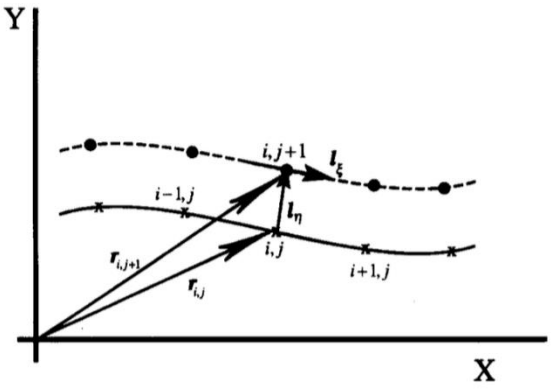


Figure 6: Interpretation of the hyperbolic partial differential equations with mesh points and vectors. [23]

In this figure, $\vec{I}_\xi$ and $\vec{I}_\eta$ are the vectors defined along and normal to the curve. The orthogonality condition is defined by $\vec{I}_\xi \cdot \vec{I}_\eta = 0$ and spacing controlling is defined by $\vec{I}_\xi \times \vec{I}_\eta = A(i,j)$, where $A(i,j)$ is the specified area. Then the equations are linearized and solved by Newton's method. His contribution is to introduce a complete description of hyperbolic grid generation, and it can be extended to many other grid generation schemes. The problem with his method is that the vector he introduced cannot be solved linearly, which needs Newton's method involving iterations. It avoids the principle that the hyperbolic scheme does not need iteration to get a fast-computing speed.

In the algorithm of hyperbolic grid generation, the most important question to each researcher is how to solve the problem that grid lines overlap in different situations. In 1995, C.H.

Tai, S.L. Yin, and C.Y. Soong [24] introduced an inherent adaptive dissipation into hyperbolic grid generation (HGAD), improving the oscillation and overlapping of grid lines. The dissipation is a field property and alters automatically, reducing the tendency to oscillate and overlapping the grid lines. The critical point of the scheme is that they discretized the original hyperbolic partial differential equations by the upwind scheme instead of normal central differencing. After the discretization, there is an inherent second-order dissipation term in the equations. Then Tai, Yin, and Soong introduced an artificial dissipation similar to the smoothing terms of Chan's scheme. This dissipation term can be second-order or forth-order. This term looks better than the smoothing terms in Chan's scheme because it adds forward and backward operators on a known term without any user defined parameters, which can avoid user interaction. However, they only compared their results and iteration numbers with the elliptic scheme and explicit hyperbolic grid generation (without smoothing). And it is not extended into three-dimension grid generation.

After that, in 1998, Kenichi Matsuno [25] developed a high-order upwind method for hyperbolic grid generation. The previous method employed a three-point backward difference scheme to discretize a derivative in a marching direction, while Matsuno used a high order accurate upwind scheme. He also demonstrated that in the hyperbolic grid generation, the first-order accurate upwind scheme is too dissipative to generate an orthogonal grid, and he could use a high-order upwind scheme to get a good grid on complex geometry with sharp edges or deep concave corners. The problem is that the method introduced an iteration procedure to solve nonlinear equations at each marching step to obtain robustness and orthogonality realized by the high-order upwind scheme. In 2D, Newton's method was used at each marching step before going to the next line. For three dimensions, a new pseudo-time iteration approach was

developed. But it is still time-consuming compared with the general hyperbolic grid generation scheme.

In recent years, Chan has kept studying overset grids. In 2004, Chan [26] found the problem setup is difficult and not standardized, so he developed a solver-independent standard protocol and made it easy to use. He also developed a Chimera Grid Tools (CGT) library, which contains file manipulation, grid information, grid editing, grid distribution, grid generation, math functions, program execution, error checking, and so on. In 2009, Chan [27] summarized what had been done till then about overset grid technology. Overset grids play an essential role in simulating high-fidelity compressible viscous flow on very complex aerospace configurations. Until 2009, surface grid generation, near-body volume grid generation, off-body volume grid generation had been used as primary grid generation schemes to generate overset grids. Also, multiple software, like visualization software, pre-processing and post-processing software, overset grid flow solvers have been developed based on the basic idea of overset grids technique and make it more convenient for users to solve flows. However, Chan [28] showed that there are still some bottlenecks in overset grids scheme. For post-processing, it is difficult to determine solution convergence if the number of grids and geometric components is large ($10^3 - 10^4$), and it is difficult to compute surface loads accurately. For volume grids domain connectivity, there is no such software that is robust, automatic, fast and has low memory usage. Moreover, for surface grid generation, making decisions in surface domain decomposition and grid point distribution to get a good grid is also a challenge, which should be studied in the future.

2.2 Hyperbolic Planar Grid Generation

We will show the math manipulation of hyperbolic grid generation by starting from the two-dimensional scheme in the coming sections. The basic idea of hyperbolic planar grid generation is to generate grids with an initial body curve and then march this curve into a far field. In Figure 7, the bold red curve is the initial body curve, and all dashed lines are unknown grid lines. At the circle point, if we want to assure its orthogonality, it needs to satisfy,

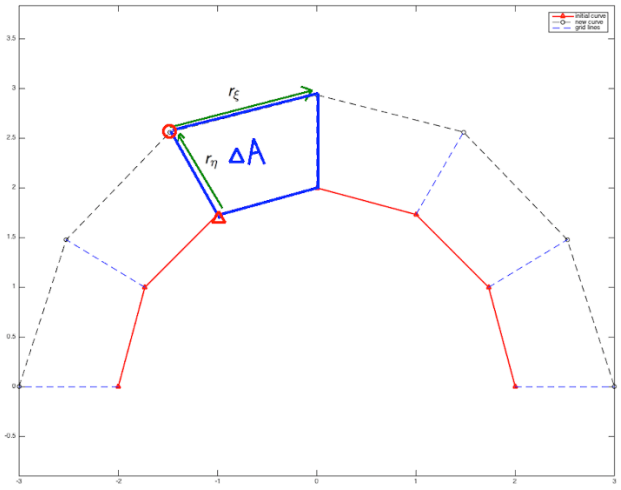


Figure 7: Illustration about grid generation based on a curve.

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \tag{2.1}$$

where $\vec{r}$ refers to the position vector of each grid point, i.e., $\vec{r} = [x, y]^T$, ξ and η are the computational coordinates along and normal to the curve. $\vec{r}_\xi$, $\vec{r}_\eta$ represents the derivatives of $\vec{r}$ in ξ and η direction, respectively. And if we want to control the spacing between the initial body curve and the next grid line, it needs to satisfy

$$\left|\vec{r}_\xi \times \vec{r}_\eta\right| = \Delta A \tag{2.2}$$

where ΔA is the local cell area, which is typically defined by the local arc length along ξ direction, i.e., $\left|\vec{r}_{\xi}\right|$, times a user-defined spacing Δs along η direction. That is,

$$\Delta A = \left|\vec{r}_{\xi}\right| * \Delta s \tag{2.3}$$

This user-defined spacing can be set very small near the initial curve and stretched out in the far-field. In this thesis, a constant stretching ratio, which is also defined by users, is applied for simplicity. This cell size specification provides reasonable grid clustering control near the body, which would be very important in the viscous calculation. An improved way of cell size specification provided by Chan [22] is discussed in Chapter 2.5.

The differential form of the control equations 2.1 and 2.2 can be written as

$$x_{\xi}x_{\eta} + y_{\xi}y_{\eta} = 0 \tag{2.4a}$$

$$x_{\xi}y_{\eta} - x_{\eta}y_{\xi} = \Delta\mathrm{A} \tag{2.4b}$$

Since all derivatives belong to points on the new grid lines, Equations 2.4 are non-linear equations that are very difficult to solve. Therefore, a local linearization of these equations is implemented at a given state 0, and we can have

$$\begin{aligned} x_{\xi}x_{\eta} &= (x_0 + \tilde{x})_{\xi}(x_0 + \tilde{x})_{\eta} \\ &= x_{\xi 0}x_{\eta 0} + x_{\xi 0}(x - x_0)_{\eta} + x_{\eta 0}(x - x_0)_{\xi} + h.o.t. \\ &\approx x_{\xi 0}x_{\eta} + x_{\eta 0}x_{\xi} - x_{\xi 0}x_{\eta 0} \end{aligned} \tag{2.5a}$$

$$\begin{aligned} x_{\xi}y_{\eta} &= (x_0 + \tilde{x})_{\xi}(y_0 + \tilde{y})_{\eta} \\ &= x_{\xi 0}y_{\eta 0} + x_{\xi 0}(y - y_0)_{\eta} + y_{\eta 0}(x - x_0)_{\xi} + h.o.t. \\ &\approx x_{\xi 0}y_{\eta} + y_{\eta 0}x_{\xi} - x_{\xi 0}y_{\eta 0} \end{aligned} \tag{2.5b}$$

$$y_\xi x_\eta = (y_0 + \tilde{y})_\xi (x_0 + \tilde{x})_\eta$$

$$= y_{\xi 0} x_{\eta 0} + y_{\xi 0}(x - x_0)_\eta + x_{\eta 0}(y - y_0)_\xi + \textit{h.o.t.}$$

$$\approx y_{\xi 0} x_\eta + x_{\eta 0} y_\xi - y_{\xi 0} x_{\eta 0} \qquad (2.5c)$$

$$y_\xi y_\eta = (y_0 + \tilde{y})_\xi (y_0 + \tilde{y})_\eta$$

$$= y_{\xi 0} y_{\eta 0} + y_{\xi 0}(y - y_0)_\eta + y_{\eta 0}(y - y_0)_\xi + \textit{h.o.t.}$$

$$\approx y_{\xi 0} y_\eta + y_{\eta 0} y_\xi - y_{\xi 0} y_{\eta 0} \qquad (2.5d)$$

with $\tilde{x}$ and $\tilde{y}$ refer to a small perturbation in x and y direction, $\textit{h.o.t.}$ refers to higher-order terms which can be neglected. Substitute Equations 2.5 into original hyperbolic partial differential Equations 2.4

$$x_\xi x_\eta + y_\xi y_\eta = x_{\xi 0} x_\eta + x_{\eta 0} x_\xi - x_{\xi 0} x_{\eta 0} + y_{\xi 0} y_\eta + y_{\eta 0} y_\xi - y_{\xi 0} y_{\eta 0}$$

$$= x_{\eta 0} x_\xi + y_{\eta 0} y_\xi + x_{\xi 0} x_\eta + y_{\xi 0} y_\eta$$

$$= 0$$

$$(2.6a)$$

$$x_\xi y_\eta - x_\eta x_\xi = x_{\xi 0} y_\eta + y_{\eta 0} x_\xi - x_{\xi 0} y_{\eta 0} - y_{\xi 0} x_\eta - x_{\eta 0} y_\xi + y_{\xi 0} x_{\eta 0}$$

$$= y_{\eta 0} x_\xi - x_{\eta 0} y_\xi - y_{\xi 0} x_\eta + x_{\xi 0} y_\eta - (x_{\xi 0} y_{\eta 0} - y_{\xi 0} x_{\eta 0})$$

$$= y_{\eta 0} x_\xi - x_{\eta 0} y_\xi - y_{\xi 0} x_\eta + x_{\xi 0} y_\eta - \Delta A_0$$

$$= \Delta A$$

$$(2.6b)$$

Equations 2.6 can be arranged into a matrix form

$$A_0 \overrightarrow{r_\xi} + B_0 \overrightarrow{r_\eta} = \vec{f} \qquad (2.7)$$

where

$$A_0 = \begin{bmatrix} x_{\eta 0} & y_{\eta 0} \\ y_{\eta 0} & -x_{\eta 0} \end{bmatrix} \; B_0 = \begin{bmatrix} x_{\xi 0} & y_{\xi 0} \\ -y_{\xi 0} & x_{\xi 0} \end{bmatrix} \; \vec{f} = \begin{bmatrix} 0 \\ \Delta A \end{bmatrix} \tag{2.8}$$

Since $|B_0| = {x_{\xi 0}}^2 + {y_{\xi 0}}^2$, we can left-multiply Equation 2.8 by ${B_0}^{-1}$, which gives

$${B_0}^{-1} A_0 \overrightarrow{r_\xi} + \overrightarrow{r_\eta} = {B_0}^{-1} \vec{f} \tag{2.9}$$

Here, a non-iterative implicit finite difference scheme, which is centrally differenced in ξ and differenced in η, is applied. And the equation 2.9 becomes to

$${B_k}^{-1} A_k \delta_\xi (\vec{r}_{k+1} - \vec{r}_k) + \vec{r}_{k+1} - \vec{r}_k = {B_k}^{-1} \vec{f}_k \tag{2.10}$$

with

$$\delta_\xi \vec{r} = \frac{\vec{r}_{j+1} - \vec{r}_{j-1}}{2} \tag{2.11a}$$

$$\vec{f}_k = (0, \Delta A_k)^T \tag{2.11b}$$

where subscript j refers to ξ direction while subscript k refers to η direction.

In Equation 2.10, the coefficient matrix B_k is directly computed using central differencing along ξ direction while the coefficient matrix A_k contains derivatives along η direction. These derivatives are calculated using Equation 2.4 as

$$\begin{pmatrix} x_\eta \\ y_\eta \end{pmatrix} = \frac{\Delta A}{x_\xi^2 + y_\xi^2} \begin{pmatrix} -y_\xi \\ x_\xi \end{pmatrix} = B^{-1} \vec{f} \tag{2.12}$$

In regions with the grid spacing changing rapidly in the ξ direction, a more robust method for calculating η derivatives will be described later in Chapter 2.6.

Equation 2.10 can also be written as,

$$\left[I + B_k^{-1} A_k \delta_\xi\right](\vec{r}_{k+1} - \vec{r}_k) = B_k^{-1} \vec{f}_k \tag{2.13}$$

where I is the 2-by-2 identity matrix. This equation means that the unknown values $\vec{r}_{j-1,k+1}$, $\vec{r}_{j,k+1}$,and $\vec{r}_{j+1,k+1}$ can be solved simultaneously with points on the known state $\vec{r}_{j-1,k}$, $\vec{r}_{j,k}$, and $\vec{r}_{j+1,k}$. With all points given on the known state, we can solve all the new points with a proper boundary condition. The boundary conditions are dictated by the topology of the initial curve or by the desired boundary behavior. For a non-periodic initial curve, users can directly define the side boundary grids starting from the two ending points on the curve. The boundaries can also be allowed for free-floating and splay conditions using the equation provided by Chan [22], which gives at $j = 1$ boundary

$$(\Delta\vec{r})_{j=1} = (\Delta\vec{r})_{j=2} + \varepsilon_x\left((\Delta\vec{r})_{j=2} - (\Delta\vec{r})_{j=3}\right) \tag{2.14}$$

where $0 \leq \varepsilon_x \leq 1$ is the extrapolation factor. This equation is a mixed zeroth- and first-order extrapolation of the boundary points. A free-floating condition is achieved by letting $\varepsilon_x = 0$, and the boundaries can splay out from the grid interior by increasing the extrapolation factor. A default value of 0.1 is used in the thesis, and a similar equation can be obtained on the other side of the boundary.

A block tridiagonal matrix can be generated and solved in these two cases by the Thomas algorithm. The block tridiagonal matrix can be written as

$$\begin{bmatrix} I & M_1 & & & & \\ -M_2 & I & M_2 & & & \\ & -M_3 & I & M_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -M_{n-1} & I & M_{n-1} \\ & & & & -M_n & I \end{bmatrix} \begin{pmatrix} \Delta\vec{r}_1 \\ \Delta\vec{r}_2 \\ \Delta\vec{r}_3 \\ \vdots \\ \Delta\vec{r}_{n-1} \\ \Delta\vec{r}_n \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_{n-1} \\ F_n \end{pmatrix} \tag{2.15}$$

with

$$M_j = \begin{cases} \frac{1}{2} {B_{j,k}}^{-1} A_{j,k}, & 2 \le j < n-1 \\ 0, & j = 1, n \end{cases} \tag{2.16a}$$

$$F_j = \begin{cases} B_{j,k}^{-1} \vec{f}_{j,k}, & 2 \le j < n-1 \\ \Delta \vec{r}_j, & j = 1, n \end{cases} \tag{2.16b}$$

for defined boundary points case and

$$M_j = \begin{cases} \frac{1}{2} {B_{j,k}}^{-1} A_{j,k}, & 2 \le j < n-1 \\ I + \frac{\varepsilon_x}{1+\varepsilon_x} M_2^{-1}, & j = 1 \\ -\left(I + \frac{\varepsilon_x}{1+\varepsilon_x} M_{n-1}^{-1} \right), & j = n \end{cases} \tag{2.17a}$$

$$F_j = \begin{cases} B_{j,k}^{-1} \vec{f}_{j,k}, & 2 \le j < n-1 \\ \frac{\varepsilon_x}{1+\varepsilon_x} M_2^{-1} F_2, & j = 1 \\ -\frac{\varepsilon_x}{1+\varepsilon_x} M_{n-1}^{-1} F_{n-1}, & j = n \end{cases} \tag{2.17b}$$

for extrapolation case. The subscripts refer to j^{th} points on the known state ranging from 1 to n. Since M_j is not a diagonal matrix, Equation 2.15 cannot be decoupled into two equations with respect to x and y coordinates separately.

In the Thomas algorithm, a forward sweep is first used to eliminate the values below the diagonal. For Equation 2.15, the second row can be modified by adding the first row times M_2 and dividing $(I + M_1 M_2)$which gives

$$\begin{bmatrix} I & M_1 & & & & \\ & I & \frac{M_2}{I+M_1M_2} & & & \\ & -M_3 & I & M_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -M_{n-1} & I & M_{n-1} \\ & & & & -M_n & I \end{bmatrix} \begin{pmatrix} \Delta\vec{r}_1 \\ \Delta\vec{r}_2 \\ \Delta\vec{r}_3 \\ \vdots \\ \Delta\vec{r}_{n-1} \\ \Delta\vec{r}_n \end{pmatrix} = \begin{pmatrix} F_1 \\ \frac{F_2+F_1M_2}{I+M_1M_2} \\ F_3 \\ \vdots \\ F_{n-1} \\ F_n \end{pmatrix} \tag{2.18}$$

Then we can take a similar step on the third row to eliminate $-M_3$ using the equation on the second row. This procedure is repeated until the nth row, and all the values below the diagonal are removed as

$$\begin{bmatrix} I & M_1 & & & & \\ & I & M'_2 & & & \\ & & I & M'_3 & & \\ & & & \ddots & \ddots & \\ & & & & I & M'_{n-1} \\ & & & & & I \end{bmatrix} \begin{pmatrix} \Delta\vec{r}_1 \\ \Delta\vec{r}_2 \\ \Delta\vec{r}_3 \\ \vdots \\ \Delta\vec{r}_{n-1} \\ \Delta\vec{r}_n \end{pmatrix} = \begin{pmatrix} F_1 \\ F'_2 \\ F'_3 \\ \vdots \\ F'_{n-1} \\ F'_n \end{pmatrix} \tag{2.19}$$

where

$$M'_j = \begin{cases} \frac{M_j}{I+M_jM'_{j-1}}, & 2 \le j \le n-1 \\ M_1, & j = 1 \end{cases} \tag{2.20a}$$

$$F'_j = \begin{cases} \frac{F_j+M_jF'_j}{I+M_jM'_{j-1}}, & 2 \le j \le n-1 \\ F_1, & j = 1 \end{cases} \tag{2.20b}$$

Then a backward substitution about this equation can be applied to produce the solution, which is

$$\Delta\vec{r}_j = \begin{cases} F'_j - M'_j\Delta\vec{r}_{j-1}, & 1 \le j \le n-1 \\ F'_j, & j = n \end{cases} \tag{2.21}$$

For a periodic boundary condition, however, additional terms are added on the top-right and bottom-left of the matrix, which becomes

$$\begin{bmatrix} I & M_1 & & & & -M_1 \\ -M_2 & I & M_2 & & & \\ & -M_3 & I & M_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -M_{n-1} & I & M_{n-1} \\ M_n & & & & -M_n & I \end{bmatrix} \begin{pmatrix} \Delta\vec{r}_1 \\ \Delta\vec{r}_2 \\ \Delta\vec{r}_3 \\ \vdots \\ \Delta\vec{r}_{n-1} \\ \Delta\vec{r}_n \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_{n-1} \\ F_n \end{pmatrix} \tag{2.22}$$

In this case, we can make use of the Sherman–Morrison formula [44] to modify our results, which gives

$$(M + uv^T)^{-1} = M^{-1} - \frac{M^{-1}uv^TM^{-1}}{1 + v^TM^{-1}u} \tag{2.23}$$

This equation can be proved by

$$\begin{aligned} (M + uv^T)\left(M^{-1} - \frac{M^{-1}uv^TM^{-1}}{1 + v^TM^{-1}u}\right) &= MM^{-1} + uv^TM^{-1} - \frac{MM^{-1}uv^TM^{-1}}{1 + v^TM^{-1}u} - \frac{uv^TM^{-1}uv^TM^{-1}}{1 + v^TM^{-1}u} \\ &= I + uv^TM^{-1} - \frac{u(1 + v^TA^{-1}u)v^TA^{-1}}{1 + v^TA^{-1}u} \\ &= I + uv^TM^{-1} - uv^TM^{-1} \end{aligned} \tag{2.24}$$

This formula provides a numerically cheap way to compute the inverse of $(M + uv^T)$ if the inverse of M can be easily found. In this case, u and v can be defined as

$$u = \begin{pmatrix} -b_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ M_n \end{pmatrix}, \qquad v = \begin{pmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \\ M_1/b_1 \end{pmatrix} \tag{2.25a}$$

where b_1 is the first element on the diagonal and is I in this case. Then the matrix M becomes

$$M = \begin{bmatrix} 2I & M_1 & & & & \\ -M_2 & I & M_2 & & & \\ & -M_3 & I & M_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -M_{n-1} & I & M_{n-1} \\ & & & & -M_n & I - M_1 M_n \end{bmatrix} \tag{2.25b}$$

and the solution is

$$\begin{aligned} \Delta\vec{r} &= (M + uv^T)^{-1} F \\ &= M^{-1}F - \frac{(M^{-1}u)v^T}{I + v^T(M^{-1}u)} M^{-1}F \\ &= M^{-1}F - (M^{-1}u)[I + v^T(M^{-1}u)]^{-1} v^T M^{-1}F \end{aligned} \tag{2.25c}$$

In practice, we first solve for $M^{-1}F$ and $M^{-1}u$ using the Thomas algorithm, and then take vector calculation to get the results. Forward sweep can be computed once as for $M^{-1}F$ and $M^{-1}u$ share the same matrix. Vector calculation can also be calculated fast since there are only two non-zero blocks in the matrix v. This solution can be obtained in $O(n)$ operations and is tested by generating grids around a circle, which is shown in Table 1. In this table, all grids are generated around a circle with $r = 1$, the initial spacing along η direction is 0.001, and the stretching ratio is 1.1.

Table 1: Computational time of generating hyperbolic grids around a circle

Number of points in the ξ direction n	Number of layers in the η direction L	Computational time t (s)	$\frac{t}{L}$ (s)	$\frac{t}{Ln}$ (s)
1001	30	0.066	$2.19 * 10^{-3}$	$2.19 * 10^{-6}$
1001	45	0.094	$2.08 * 10^{-3}$	$2.08 * 10^{-6}$
1001	60	0.125	$2.08 * 10^{-3}$	$2.08 * 10^{-6}$
1001	90	0.177	$1.96 * 10^{-3}$	$1.96 * 10^{-6}$
2001	30	0.130	$4.33 * 10^{-3}$	$2.17 * 10^{-6}$
2001	45	0.195	$4.34 * 10^{-3}$	$2.17 * 10^{-6}$
2001	60	0.250	$4.16 * 10^{-3}$	$2.08 * 10^{-6}$
2001	90	0.374	$4.15 * 10^{-3}$	$2.08 * 10^{-6}$
4001	30	0.281	$9.38 * 10^{-3}$	$2.34 * 10^{-6}$
4001	45	0.416	$9.25 * 10^{-3}$	$2.31 * 10^{-6}$
4001	60	0.547	$9.11 * 10^{-3}$	$2.28 * 10^{-6}$
4001	90	0.782	$8.69 * 10^{-3}$	$2.17 * 10^{-6}$

Note: Computational time may be different on different software and operating system. Cases are compiled and run in Clion software from JetBrains on a laptop with Windows 10 operating system, Intel(R) Core i7-7700HQ CPU 2.80 GHz processor, and 16GB RAM. Same computer sets are used for all tables in the thesis.

From this table, we can find that the computational time of hyperbolic grid generation is linearly related to the number of layers in the η direction. $O(n)$ operations are required to

generate each layer of the grids based on the number of points in the ξ direction n. This result can be further proved by taking a logarithmic fit between the computational time/number of layers and the number of points, shown in Figure 8. A slope of 1.06 is calculated from the figure, which is reasonable and within the calculation error.

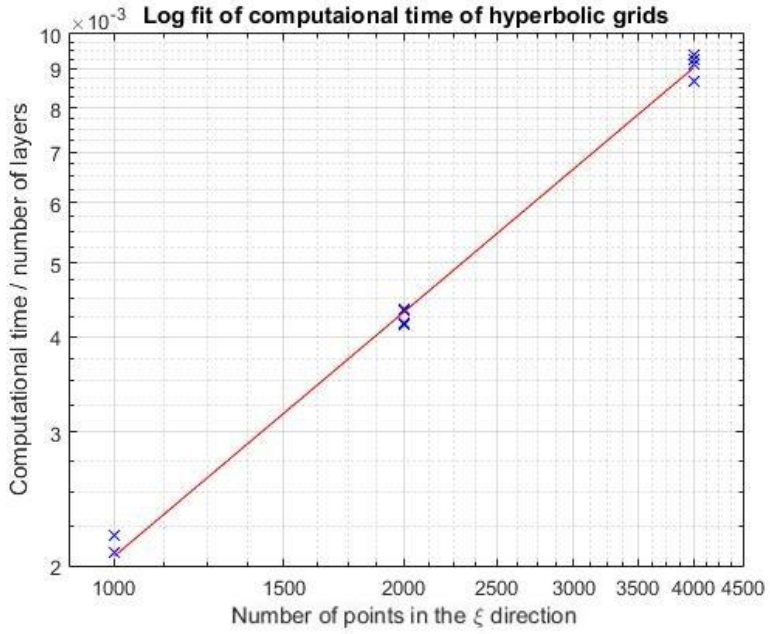
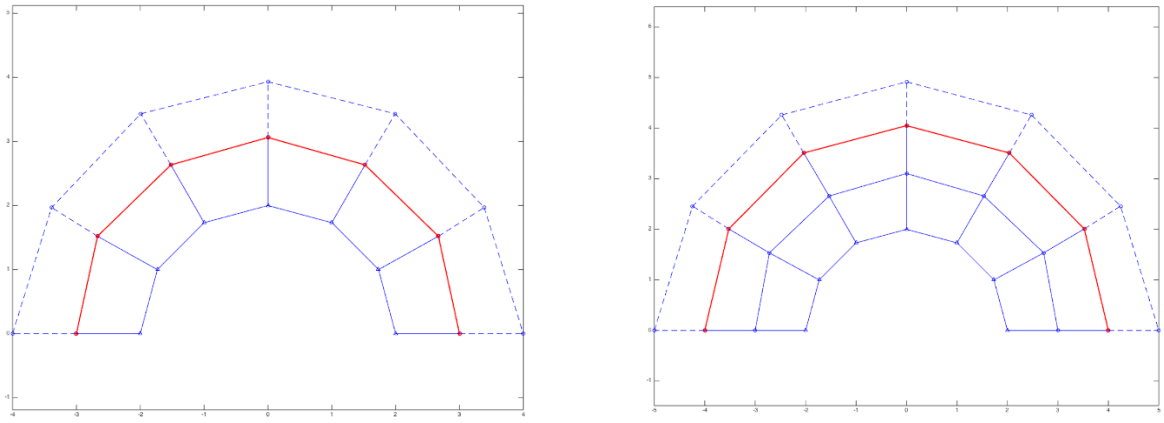


Figure 8: Logarithmic fit of computational time of hyperbolic grids

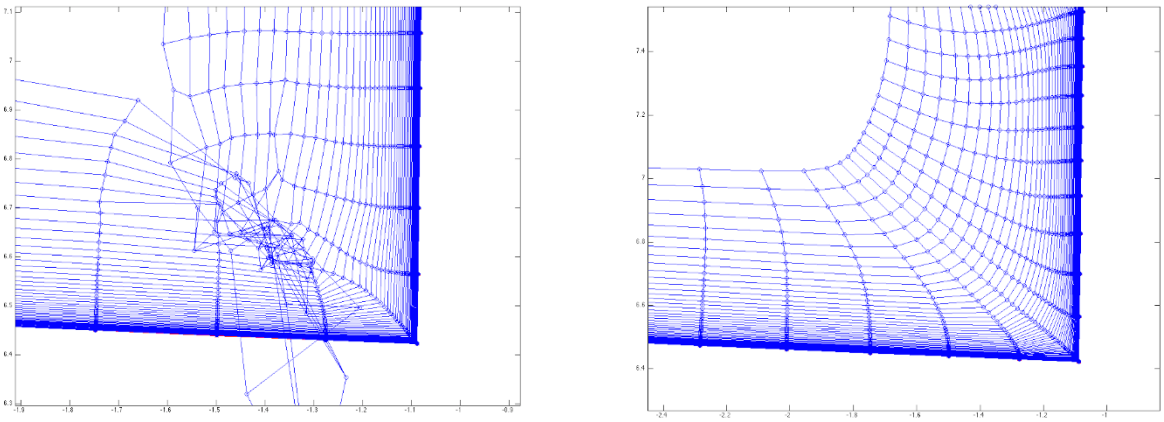
After finding the solutions of $(\vec{r}_{k+1} - \vec{r}_k)$, we can get the points on the new grid lines. Then we can use this new grid line as an initial curve to generate the next grid line as Figure 9. In Figure 9 (a), the dashed line is calculated based on the red line, and that dashed line becomes the red line in Figure 9 (b), which can be used to generate the next level of grid lines.



(a) Generate the second grid line (b) Generate the third grid line

Figure 9: Further illustration for grid generation.

This equation works well for the convex body curve. However, problems occur when the hyperbolic grid generation scheme is applied to a concave corner. When the normal vectors of two points intersect with each other, it can lead to grid lines overlapping, as Figure 10 (a)



(a) Grids without smoothing (b) Grids after adding smoothing

Figure 10: Grids after adding smoothing terms

In order to improve the grid quality at concave corners, the implicitness factor θ [29] and explicit and implicit smoothing terms [22] are added into Equation 2.13, and the equations become to

$$\left[I + \left(1 + \theta_\xi\right)B_k^{-1}A_k\delta_\xi - \varepsilon_{i\xi}(\Delta\nabla)_\xi\right](\vec{r}_{k+1} - \vec{r}_k) = B_k^{-1}\vec{f}_{k+1} - \varepsilon_{e\xi}(\Delta\nabla)_\xi\vec{r}_k \qquad (2.26)$$

where $(\Delta\nabla)_\xi \vec{r}_k = \vec{r}_{j+1} - 2\vec{r}_j + \vec{r}_{j-1}$. The implicitness factor θ ranges between 0 and 4. A larger value of the implicitness factor is used to deal with large concave geometry. However, it would also lead to instability in the far-field of hyperbolic grids. In this thesis, a default value of 2 is applied to reduce grid overlapping. $\varepsilon_{i\xi}$, $\varepsilon_{e\xi}$ are implicit and explicit smoothing terms, $\varepsilon_{i\xi} \approx 2\varepsilon_{e\xi}$. The detail about smoothing terms will be talked about later. This equation can be solved by using the Thomas algorithm and the Sherman–Morrison formula discussed above with some modification of the matrix. After adding the smoothing term, the grid lines in the concave corner do not overlap, as shown in Figure 10 (b).

2.3 Hyperbolic Surface Grid Generation

The governing equation of hyperbolic grid generation on a curved surface is similar to that in two dimensions. Since there is one more unknown variable, one more equation is needed to solve the problem. Except for orthogonality and cell size constraints, another equation is added that the marching direction $\overrightarrow{r_\eta}$ is perpendicular to the surface normal direction, as shown in Figure 11. The governing equation can be written as

$$x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta = 0 \tag{2.27a}$$

$$\hat{n}_1\left(y_\xi z_\eta - z_\xi y_\eta\right) + \hat{n}_2\left(z_\xi x_\eta - x_\xi z_\eta\right) + \hat{n}_3\left(x_\xi y_\eta - y_\xi x_\eta\right) = \Delta S \tag{2.27b}$$

$$\hat{n}_1 x_\eta + \hat{n}_2 y_\eta + \hat{n}_3 z_\eta = 0 \tag{2.27c}$$

where $\vec{n} = (\hat{n}_1, \hat{n}_2, \hat{n}_3)$ is the local unit surface normal. ΔS is the local cell size on the surface, which is the same as the cell size in two dimensions with z coordinates added.

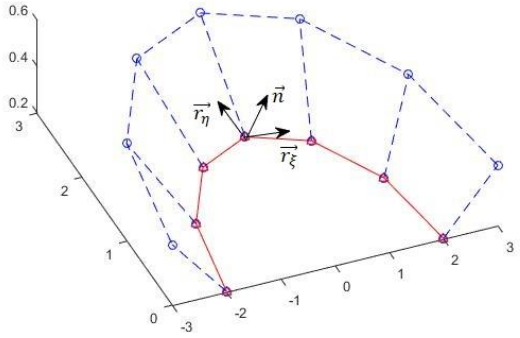


Figure 11: Illustration of grids generated on a curved surface.

Local linearization of these equations will result in the same grid generation equation except for changing the size of matrix or vector, which gives

$$A_0\overrightarrow{r_\xi} + B_0\overrightarrow{r_\eta} = \vec{f} \tag{2.28}$$

where

$$\mathrm{A} = \begin{bmatrix} x_\eta & y_\eta & z_\eta \\ \hat{n}_3 y_\eta - \hat{n}_2 z_\eta & \hat{n}_1 z_\eta - \hat{n}_3 x_\eta & \hat{n}_2 x_\eta - \hat{n}_1 y_\eta \\ 0 & 0 & 0 \end{bmatrix} \tag{2.29a}$$

$$\mathrm{B} = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ -\hat{n}_3 y_\xi - \hat{n}_2 z_\xi & -\hat{n}_1 z_\xi - \hat{n}_3 x_\xi & -\hat{n}_2 x_\xi - \hat{n}_1 y_\xi \\ \hat{n}_1 & \hat{n}_2 & \hat{n}_3 \end{bmatrix} \tag{2.29b}$$

$$\vec{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.29c}$$

$$\vec{f} = \begin{bmatrix} 0 \\ \Delta S \\ 0 \end{bmatrix} \tag{2.29d}$$

The matrix B_0^{-1} exists unless the arc length in ξ direction is zero. Besides, $B_0^{-1}A_0$ is symmetric, and the system of equations is hyperbolic for marching in η direction.

Similar to the scheme employed for hyperbolic planar grid generation, Equation 2.28 is solved numerically by central differencing with explicit and implicit smoothing in ξ direction and implicit differencing in η direction. The governing equation can be written as

$$\left[I + \left(1 + \theta_\xi\right)B_k^{-1}A_k\delta_\xi - \varepsilon_{i\xi}(\Delta\nabla)_\xi\right](\vec{r}_{k+1} - \vec{r}_k) = B_k^{-1}\vec{f}_{k+1} - \varepsilon_{e\xi}(\Delta\nabla)_\xi\vec{r}_k \tag{2.30}$$

which is the same as Equation 2.26.

The elements of matrix A contains η derivatives, which can be computed using Equation 2.27 as

$$\begin{pmatrix} x_\eta \\ y_\eta \\ z_\eta \end{pmatrix} = \frac{1}{\beta}\begin{bmatrix} x_\xi\hat{n}_1 w & \hat{n}_2 z_\xi - \hat{n}_3 y_\xi & \hat{n}_1 s_\xi^2 - x_\xi w \\ y_\xi\hat{n}_2 w & \hat{n}_3 x_\xi - \hat{n}_1 z_\xi & \hat{n}_2 s_\xi^2 - y_\xi w \\ z_\xi\hat{n}_3 w & \hat{n}_1 y_\xi - \hat{n}_2 x_\xi & \hat{n}_3 s_\xi^2 - z_\xi w \end{bmatrix}\vec{g} = B^{-1}\vec{g} \tag{2.31}$$

where

$$w = \hat{n}\cdot\vec{r}_\xi = \hat{n}_1 x_\xi + \hat{n}_2 y_\xi + \hat{n}_3 z_\xi \tag{2.32a}$$

$$s_\xi^2 = \vec{r}_\xi\cdot\vec{r}_\xi = x_\xi^2 + y_\xi^2 + z_\xi^2 \tag{2.32b}$$

$$\beta = Det(B) = s_\xi^2 - w^2 \tag{2.32c}$$

When calculating for a new layer of grid points, the local unit surface normal of each point is computed based on the current known old points. Therefore, these new points calculated by this method are not on the surface and should be projected onto the reference surface after each marching step. In Figure 12, *P'* is the point computed on $\overrightarrow{r_\eta}$ direction. This point should be

projected onto the surface, which gives the point *P*. Generally, for a given parameterized reference surface and each grid point, the normal equation can be applied to project the calculated point onto the surface, which can give the results with the least squared residuals. By using the Jacobian matrix at reference points, the difference between the calculated point and the reference point in Cartesian coordinates $(dx \quad dy \quad dz)^T$ can be converted into a difference in parameter base $(du \quad dv)^T$.

$$J\begin{bmatrix} du \\ dv \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.33}$$

where J is the Jacobian matrix at given points

$$J = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} \tag{2.34}$$

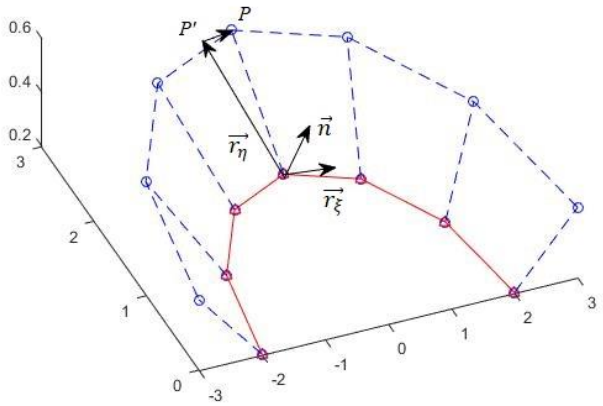


Figure 12: Project calculated results onto the surface.

Once the differences in parameter base du and dv are calculated, the new grid point can be acquired by adding this difference onto the reference point and evaluating the new parameters. This new point is guaranteed on the reference surface and is closest to the calculated point if the surface can be parameterized. For the cases with multiple surfaces, further steps to locate and project the new grid points will be discussed in Chapter 5.2.

2.4 Hyperbolic Volume Grid Generation

In three-dimensional hyperbolic grid generation, the general coordinates become to $\xi(x,y,z)$, $\eta(x,y,z)$, and $\zeta(x,y,z)$ corresponding to grid indices j, k, and l. The requirements of 3D hyperbolic volume grid generation equations can be written as

$$\overrightarrow{r_\xi} \cdot \overrightarrow{r_\zeta} = x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta = 0 \tag{2.35a}$$

$$\overrightarrow{r_\eta} \cdot \overrightarrow{r_\zeta} = x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta = 0 \tag{2.35b}$$

$$\overrightarrow{r_\zeta} \cdot \left(\overrightarrow{r_\xi} \times \overrightarrow{r_\eta}\right) = \begin{vmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ x_\zeta & y_\zeta & z_\zeta \end{vmatrix} = \Delta V \tag{2.35c}$$

ΔV is derived by the surface area ΔA times a user-defined spacing off the surface Δs, which is

$$\Delta V_{j,k,l} = \Delta A_{j,k,l} \Delta s \tag{2.36}$$

where the surface area ΔA is computed using the cross product of the derivatives at the grid points, which is

$$\Delta A_{j,k,l} = \overrightarrow{r_\xi} \times \overrightarrow{r_\eta} \tag{2.37}$$

Local linearization of Equation 2.23 gives

$$A_0\overrightarrow{r_\xi} + B_0\overrightarrow{r_\eta} + C_0\overrightarrow{r_\zeta} = \vec{e} \tag{2.38}$$

where

$$A = \begin{bmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ (y_\eta z_\zeta - y_\zeta z_\eta) & (x_\zeta z_\eta - x_\eta z_\zeta) & (x_\eta y_\zeta - x_\zeta y_\eta) \end{bmatrix} \tag{2.39a}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z_\zeta \\ (y_\zeta z_\xi - y_\xi z_\zeta) & (x_\xi z_\zeta - x_\zeta z_\xi) & (x_\zeta y_\xi - x_\xi y_\zeta) \end{bmatrix} \tag{2.39b}$$

$$C = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ (y_\xi z_\eta - y_\eta z_\xi) & (x_\eta z_\xi - x_\xi z_\eta) & (x_\xi y_\eta - x_\eta y_\xi) \end{bmatrix} \tag{2.39c}$$

$$\vec{e} = \begin{bmatrix} 0 \\ 0 \\ \Delta V \end{bmatrix} \tag{2.39d}$$

The coefficient matrix A, B, and C in this equation contains derivatives in ξ, η, and ζ directions. The ξ and η derivatives can be obtained directly by central differencing, while ζ derivatives are calculated based on Equation 2.35 as a linear combination of ξ and η derivatives which is shown as:

$$\begin{pmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{pmatrix} = \frac{\Delta V}{Det(C)} \begin{pmatrix} y_\xi z_\eta - y_\eta z_\xi \\ x_\eta z_\xi - x_\xi z_\eta \\ x_\xi y_\eta - x_\eta y_\xi \end{pmatrix} = C^{-1}\vec{g} \tag{2.40}$$

with $Det(C) = \left(y_\xi z_\eta - y_\eta z_\xi\right)^2 + \left(x_\eta z_\xi - x_\xi z_\eta\right)^2 + \left(x_\xi y_\eta - x_\eta y_\xi\right)^2$. This equation can also be improved by the metric correction, which is talked about later.

Using finite differencing for Equation 2.38 and left multiplying C_l^{-1} we can get

$$C_l^{-1}A_l\delta_\xi(\vec{r}_{l+1} - \vec{r}_l) + C_l^{-1}B_l\delta_\eta(\vec{r}_{l+1} - \vec{r}_l) + (\vec{r}_{l+1} - \vec{r}_l) = C_l^{-1}\vec{g}_l \tag{2.41}$$

with

$$\delta_\xi \vec{r} = \frac{\vec{r}_{j+1} - \vec{r}_{j-1}}{2} \tag{2.42a}$$

$$\delta_\eta \vec{r} = \frac{\vec{r}_{k+1} - \vec{r}_{k-1}}{2} \tag{2.42b}$$

$$\vec{g}_l = (0, 0, \Delta V_l)^T \tag{2.42c}$$

This equation provides a relation between the five points (one point and its four neighbor points along ξ and η directions) on the known surface and the five points on the new surface, which can be written into a sparse matrix that can be tough to solve.

A better way to solve Equation 2.41 is provided by approximate factoring, which gives

$$\left[I + C_l^{-1} B_l \delta_\eta\right] \left[I + C_l^{-1} A_l \delta_\xi\right] (\vec{r}_{l+1} - \vec{r}_l) = C_l^{-1} \vec{g}_l \tag{2.43}$$

where I is the 3-by-3 identity matrix. In this equation, an additional term $[C_l^{-1} B_l C_l^{-1} A_l \delta_\eta \delta_\xi](\vec{r}_{l+1} - \vec{r}_l)$ is added for simplicity with $\delta_\eta \delta_\xi \vec{r}$ is actually the differential form of second-order cross partial derivative $\vec{r}_{\eta\xi}$, which would be very small and negligible in most of the cases. After approximate factoring, the problem is now reduced to solve a sequence of block tridiagonal systems along ξ and η directions. Periodic and non-periodic boundary conditions can be applied on ξ and η directions based on the geometry of the initial surface. The Thomas algorithm and the Sherman–Morrison formula discussed before can then be applied to solve the matrix along ξ and η directions separately.

Like in hyperbolic planar grid generation, smoothing is also required to get a better performance in concave surfaces, which gives

$$\left[I + \left(1+\theta_\eta\right)C_l^{-1}B_l\delta_\eta - \varepsilon_{i\eta}(\Delta\nabla)_\eta\right]\left[I + \left(1+\theta_\xi\right)C_l^{-1}A_l\delta_\xi - \varepsilon_{i\xi}(\Delta\nabla)_\xi\right](\vec{r}_{l+1} - \vec{r}_l) = C_l^{-1}\vec{g}_{l+1} - \left[\varepsilon_{e\xi}(\Delta\nabla)_\xi + \varepsilon_{e\eta}(\Delta\nabla)_\eta\right]\overrightarrow{r_l} \quad (2.44)$$

2.5 Grid Quality Improvement Mechanism

2.5.1 Smoothing

Added dissipation is an essential part of controlling the smoothness of the grid. The form and magnitude of dissipation may change the shape of the grid significantly, and therefore the dissipation must be applied selectively for different body shapes. A low amount of smoothing is desired near the body and in low curvature regions of the geometry, where grid orthogonality should dominate. In contrast, a large value of smoothing is needed to prevent grid lines from crossing in concave regions. A spatially variable dissipation coefficient based on the above attributes was in Chan and Steger's paper [22]. It should be noticed that the smoothing terms should be relatively small compared with the other terms in the hyperbolic grid generation equations. Otherwise, the characteristics of the governing equations would be changed.

The explicit dissipation coefficient $\varepsilon_{e\xi}$ in planar and surface grid generation depend on five quantities as follows:

$$\varepsilon_{e\xi} = \varepsilon_c N_\xi S_k \bar{d}_{j,k}^{\xi} a_{j,k}^{\xi} \quad (2.45)$$

In this equation, ε_c is a user-specified constant of $O(1)$. The value of ε_c varies for different shapes of bodies. Small values of ε_c would be applied for cases where less dissipation is required while large values of ε_c would be taken to get more dissipation. A default value of 0.5 is used in the thesis.

N_ξ is the scaling with the local mesh spacing, which can be approximated by the matrix norm $\|B^{-1}A\|$, given by

$$N_\xi = \sqrt{\frac{x_\eta^2 + y_\eta^2 + z_\eta^2}{x_\xi^2 + y_\xi^2 + z_\xi^2}} \tag{2.46}$$

The scaling function S_k controls the dissipation based on the distance from the body such that small dissipation is applied near-body where orthogonality is desired. The form of the function is given by

$$S_k = \begin{cases} \sqrt{\dfrac{(k-1)}{(k_{max}-1)}}, & 1 \le k \le k_{trans} \\ \sqrt{\dfrac{(k_{trans}-1)}{(k_{max}-1)}}, & k_{trans} \le k \le k_{max} \end{cases} \tag{2.47}$$

where k_{max} is the number of points in k direction and k_{trans} is restricted to the range $\left[\frac{3}{4}k_{max}, k_{max}\right]$ and once the following is true:

$$\max_j d_{j,k}^{\xi} - \max_j d_{j,k-1}^{\xi} < 0 \tag{2.48}$$

This term is used to guarantee slight smoothing near the body curve. As the number of layers increases, larger smoothing is required to generate grids in the concave corners. This term is also increased to match up the smoothing. And since we also have the grid point distribution sensor function and the grid angle function to deal with the concave corners, smoothing provided by this term will be enough and will no longer need to increase when the convergence of the local grid lines is slowing down (Equation 2.48).

The gridpoint distribution sensor function $\bar{d}_{j,k}^{\xi}$ is given by

$$\bar{d}_{j,k}^{\xi} = max\left[\left(d_{j,k}^{\xi}\right)^{\frac{2}{S_k}}, 0.1\right] \tag{2.49}$$

where

$$d_{j,k}^{\xi} = \begin{cases} \dfrac{\left|\vec{r}_{j+1,k-1} - \vec{r}_{j,k-1}\right| + \left|\vec{r}_{j-1,k-1} - \vec{r}_{j,k-1}\right|}{\left|\vec{r}_{j+1,k} - \vec{r}_{j,k}\right| + \left|\vec{r}_{j-1,k} - \vec{r}_{j,k}\right|}, & k \geq 2 \\ 1, & k = 1 \end{cases} \tag{2.50}$$

This term depends on the ratio of the average distances between grid points at layer (k-1) to that at layer k, which means that it can locally increase the smoothing where grid line convergence is detected.

The grid angle function $a_{j,k}^{\xi}$ is used to locally increase the smoothing at severe concave corner points. It can be more conventionally defined in the following unit vectors. Let vectors points in the plus and minus ξ directions at grid point (j,k) be represented by $\vec{r}_j^{\,+}$ and $\vec{r}_j^{\,-}$ respectively, where

$$\vec{r}_j^{\,+} = \vec{r}_{j+1,k} - \vec{r}_{j,k}, \quad \vec{r}_j^{\,-} = \vec{r}_{j-1,k} - \vec{r}_{j,k} \tag{2.51}$$

And let $\hat{r}_j^{+}$ and $\hat{r}_j^{-}$ represent the unit vectors of $\vec{r}_j^{\,+}$ and $\vec{r}_j^{\,-}$. The local unit normal $\hat{n}_{j,k}$ can be calculated as

$$\hat{n}_{j,k} = \frac{\vec{n}_{sur} \times \left(\hat{r}_j^{+} - \hat{r}_j^{-}\right)}{|\vec{n}_{sur}|\left|\hat{r}_j^{+} - \hat{r}_j^{-}\right|} \tag{2.52}$$

where $\vec{n}_{sur}$ is the surface normal and is $(0,0,1)^T$ for two-dimensional grids.

The cosine of the local half-angle $\alpha_{j,k}$ is given by

$$\cos\alpha_{j,k} = \hat{n}_{j,k} \cdot \hat{r}_j^{+} = \hat{n}_{j,k} \cdot \hat{r}_j^{-} \tag{2.53}$$

The grid angle function $a_{j,k}^{\xi}$ is then defined as

$$a_{j,k}^{\xi} = \begin{cases} \dfrac{1}{\left(1 - cos^2\alpha_{j,k}\right)} & \text{if } 0 \le \alpha_{j,k} \le \dfrac{\pi}{2} \\ 1 & \text{if } \dfrac{\pi}{2} \le \alpha_{j,k} \le \pi \end{cases} \tag{2.54}$$

which means that the grid angle function is designed to have the value of one except at a severe concave corner point. The effect of the smoothing terms has been discussed previously and shown in Figure 10.

For hyperbolic volume grid generation, similar equations can be obtained for $\varepsilon_{e\xi}$ and $\varepsilon_{e\eta}$ by calculating relevant coefficient along ξ and η direction separately, which is given below as

$$\varepsilon_{e\xi} = \varepsilon_c N_\xi S_l \bar{d}_{j,k,l}^{\xi} a_{j,k,l}^{\xi} \tag{2.55a}$$

$$\varepsilon_{e\eta} = \varepsilon_c N_\eta S_l \bar{d}_{j,k,l}^{\eta} a_{j,k,l}^{\eta} \tag{2.55b}$$

$$N_\xi = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\xi^2 + y_\xi^2 + z_\xi^2}} \tag{2.56a}$$

$$N_\eta = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\eta^2 + y_\eta^2 + z_\eta^2}} \tag{2.56b}$$

$$S_l = \begin{cases} \sqrt{\dfrac{(l-1)}{(l_{max}-1)}}, & 1 \le l \le l_{trans} \\ \sqrt{\dfrac{(l_{trans}-1)}{(l_{max}-1)}}, & l_{trans} \le l \le l_{max} \end{cases} \tag{2.57}$$

l_{trans} is set in the range $\left[\frac{3}{4}k_{max}, k_{max}\right]$ and one of the Equations 2.58 (a) and (b) is true

$$\max_{j,k} d^{\xi}_{j,k,l} - \max_{j,k} d^{\xi}_{j,k,l-1} < 0 \tag{2.58a}$$

$$\max_{j,k} d^{\eta}_{j,k,l} - \max_{j,k} d^{\eta}_{j,k,l-1} < 0 \tag{2.58b}$$

$$\bar{d}^{\xi}_{j,k,l} = max\left[\left(d^{\xi}_{j,k,l}\right)^{\frac{2}{S_l}}, 0.1\right] \tag{2.59a}$$

$$\bar{d}^{\eta}_{j,k,l} = max\left[\left(d^{\eta}_{j,k,l}\right)^{\frac{2}{S_l}}, 0.1\right] \tag{2.59b}$$

$$d^{\xi}_{j,k,l} = \begin{cases} \dfrac{\left|\vec{r}_{j+1,k,l-1} - \vec{r}_{j,k,l-1}\right| + \left|\vec{r}_{j-1,k,l-1} - \vec{r}_{j,k,l-1}\right|}{\left|\vec{r}_{j+1,k,l} - \vec{r}_{j,k,l}\right| + \left|\vec{r}_{j-1,k,l} - \vec{r}_{j,k,l}\right|}, & k \geq 2 \\ 1, & k = 1 \end{cases} \tag{2.60a}$$

$$d^{\eta}_{j,k,l} = \begin{cases} \dfrac{\left|\vec{r}_{j,k+1,l-1} - \vec{r}_{j,k,l-1}\right| + \left|\vec{r}_{j,k-1,l-1} - \vec{r}_{j,k,l-1}\right|}{\left|\vec{r}_{j,k+1,l} - \vec{r}_{j,k,l}\right| + \left|\vec{r}_{j,k-1,l} - \vec{r}_{j,k,l}\right|}, & k \geq 2 \\ 1, & k = 1 \end{cases} \tag{2.60b}$$

$$a^{\xi}_{j,k,l} = \begin{cases} \dfrac{1}{\left(1 - cos^2\alpha_{j,k,l}\right)} & \text{if } 0 \leq \alpha_{j,k,l} \leq \dfrac{\pi}{2} \\ 1 & \text{if } \dfrac{\pi}{2} \leq \alpha_{j,k,l} \leq \pi \end{cases} \tag{261a}$$

$$a^{\eta}_{j,k,l} = \begin{cases} \dfrac{1}{\left(1 - cos^2\beta_{j,k,l}\right)} & \text{if } 0 \leq \beta_{j,k,l} \leq \dfrac{\pi}{2} \\ 1 & \text{if } \dfrac{\pi}{2} \leq \beta_{j,k,l} \leq \pi \end{cases} \tag{2.61b}$$

$$\cos\alpha_{j,k,l} = \hat{n}_{j,k,l}\hat{r}^{+}_{j} = \hat{n}_{j,k,l}\hat{r}^{-}_{j} \tag{2.62a}$$

$$\cos\beta_{j,k,l} = \hat{n}_{j,k,l}\hat{r}^{+}_{k} = \hat{n}_{j,k,l}\hat{r}^{-}_{k} \tag{2.62b}$$

$$\hat{n}_{j,k,l} = \frac{\left(\hat{r}_j^+ - \hat{r}_j^-\right) \times \left(\hat{r}_k^+ - \hat{r}_k^-\right)}{\left|\left(\hat{r}_j^+ - \hat{r}_j^-\right) \times \left(\hat{r}_k^+ - \hat{r}_k^-\right)\right|} \tag{2.63}$$

$$\vec{r}_j^{\,+} = \vec{r}_{j+1,k,l} - \vec{r}_{j,k,l}, \quad \vec{r}_j^{\,-} = \vec{r}_{j-1,k,l} - \vec{r}_{j,k,l} \tag{2.64a}$$

$$\vec{r}_k^{\,+} = \vec{r}_{j,k+1,l} - \vec{r}_{j,k,l}, \quad \vec{r}_k^{\,-} = \vec{r}_{j,k-1,l} - \vec{r}_{j,k,l} \tag{2.64b}$$

2.5.2 Cell size specification

Except for adding dissipation terms, there is another way to enhance grid smoothness by performing smoothing steps on the prescribed cell sizes, which is also introduced by Chan [22]. The improved way of cell size specification is given by averaging the cell size of its neighbor cells, which is

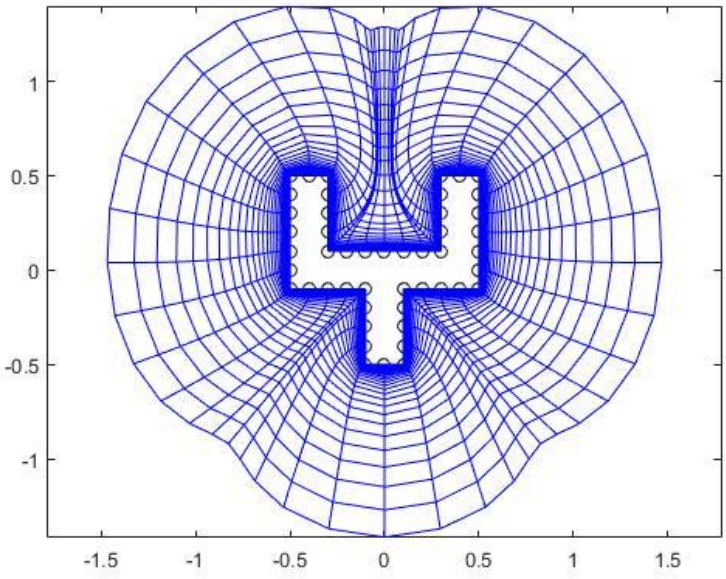
$$\Delta A_{j,k} = (1 - v_a)\Delta\bar{A}_{j,k} + \frac{v_a}{2}\left(\Delta\bar{A}_{j-1,k} + \Delta\bar{A}_{j+1,k}\right) \tag{2.65}$$

for two-dimensional grids and

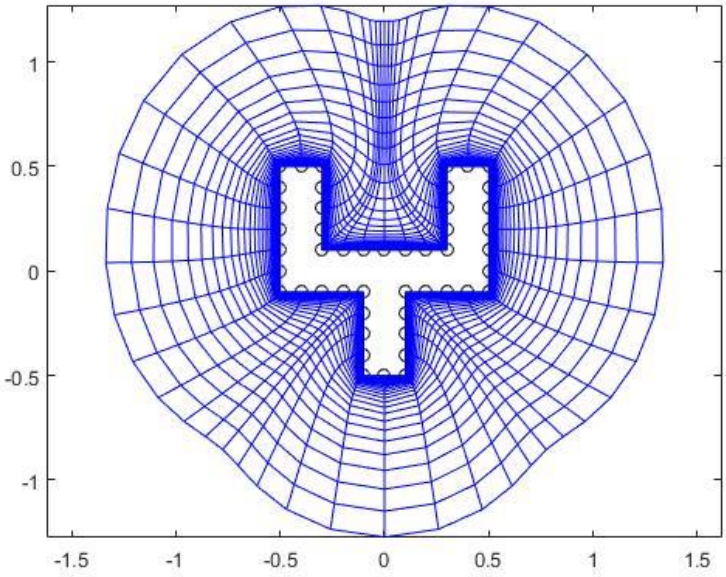
$$\Delta V_{j,k,l} = (1 - v_a)\Delta\bar{V}_{j,k,l} + \frac{v_a}{4}\left(\Delta\bar{V}_{j-1,k,l} + \Delta\bar{V}_{j+1,k,l} + \Delta\bar{V}_{j,k-1,l} + \Delta\bar{V}_{j,k+1,l}\right) \tag{2.66}$$

for three-dimensional grids, where $\Delta\bar{A}$ and $\Delta\bar{V}$ are the original area and volume of the grid cells in Equation 2.3 and 2.24, respectively. v_a is the volume average factor that can help generate more uniformly spaced grids. A default value of $v_a = 0.5$ is used in this thesis. Varying the volume average factor has a similar effect as the dissipation terms, i.e., a larger value can better avoid grid clustering but get less orthogonality. A smaller value has the opposite effect. An example with different values of v_a is shown in Figure 13. It can be seen that grids are highly

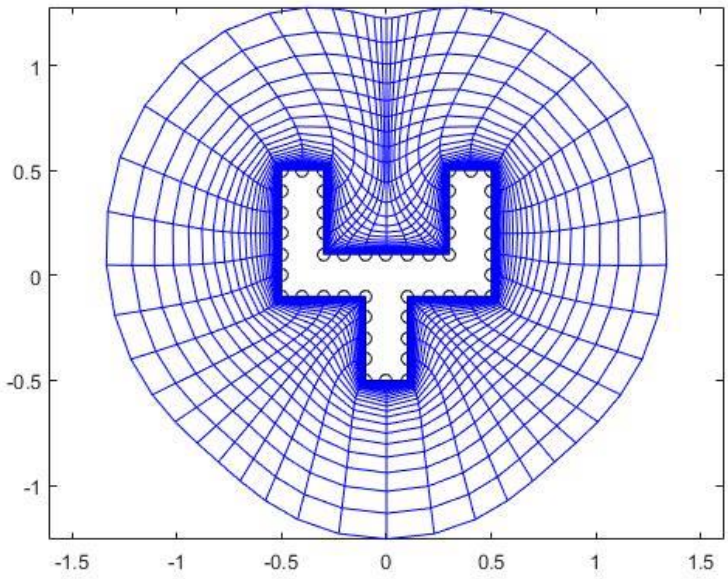
clustered in the concave regions with $v_a = 0$ in Figure 13 (a), and this problem is relieved with $v_a = 0.5$ in Figure 13 (b).



(a) Hyperbolic grids with $v_a = 0$



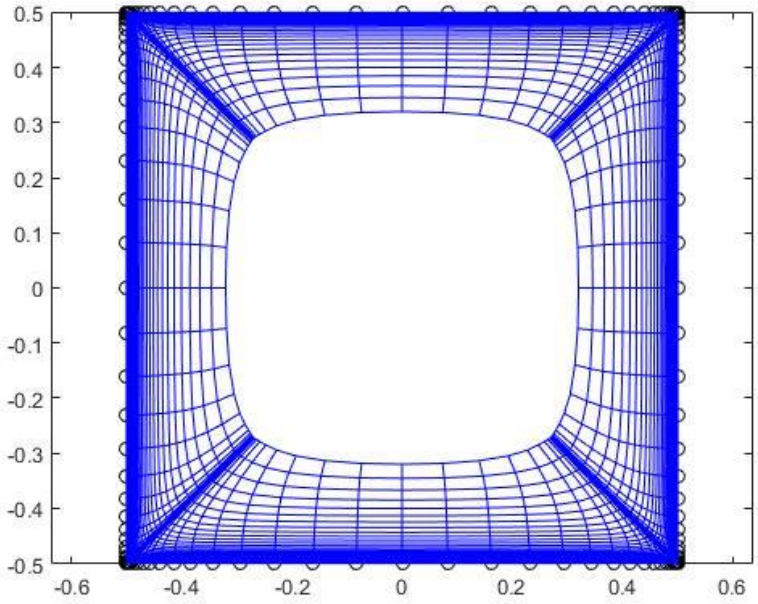
(b) Hyperbolic grids with $v_a = 0.5$



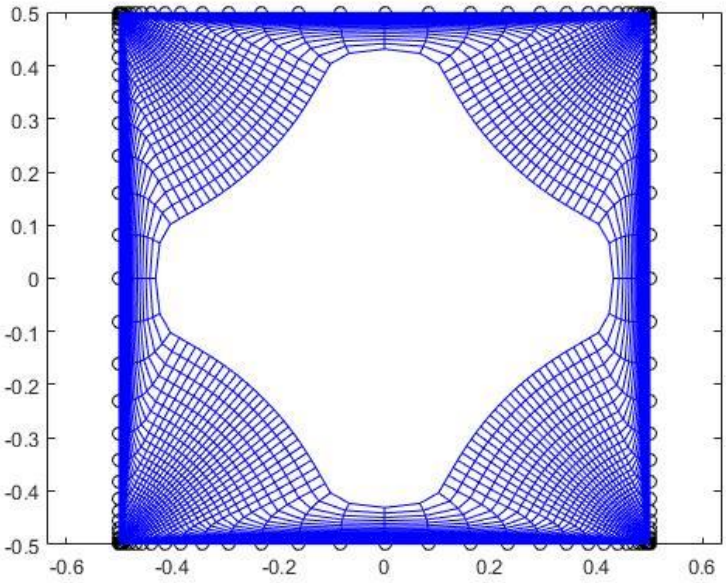
(c) Hyperbolic grids with $v_a = 0.5$, taking the average step 5 times

Figure 13: Hyperbolic grids with different values of volume average factor

In practice, this averaging step can be applied one or more times when calculating each layer of hyperbolic grids to enlarge the small cells and lessen the large cells slightly. Taking this step iteratively can further avoid grid clustering or twisting and thus smooth the grids. Figure 13 (c) shows the grid result of the same case after taking the averaging step five times (itsvol = 5). However, the number of iteration times is another user-defined parameter. A small number of iteration times may remain the problem of grid clustering or twisting. On the other hand, a large number of iteration times may lead to an over-smoothed grid. An example is shown in Figure 14, where the grids are generated inside a square, and the points are clustered in the corners. The grids are generated by taking averaging step 50 times. In this thesis, a default value of itsvol = 5 is applied.



(a) Hyperbolic grids with $itsvol = 5$



(b) Hyperbolic grids with $itsvol = 50$

Figure 14: Hyperbolic grid generated inside a square

2.6 Metric Correction

The smoothing terms mentioned in Chapter 2.5 are typically sufficient to generate a smooth grid at either convex or concave corners if the grid spacing to each side of the corner is approximately equal. Nevertheless, if the grid spacing is much different at a corner, smoothing terms alone is usually not enough to get a smooth grid. Therefore, an additional remedy, called the metric correction, is applied in such cases to solve the problem.

With η derivatives shown in Equation 2.12, the direction in which the grid will develop is such that it is perpendicular to the line joining the two neighbor points of the corner, which will not be a good feature for a corner with unevenly spaced points. An example is shown in Figure 15 (a). In order to guide the grids out in a direction that can bisect the angles at a point subtended by its neighbor points, the derivatives are modified as follows:

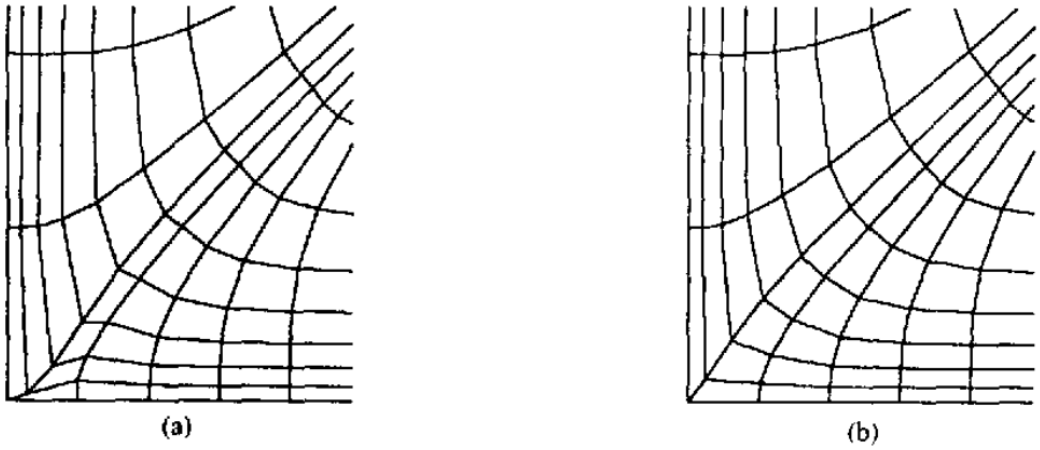


Figure 15: Comparison treatment of concave corner with unequal grid spacings. (a) Without metric correction. (b) with metric correction. [22]

$$\begin{pmatrix} x'_\eta \\ y'_\eta \end{pmatrix} = \frac{\Delta A}{Det(B')} \begin{pmatrix} -y'_\xi \\ x'_\xi \end{pmatrix} \tag{2.67}$$

where

$$Det(B') = {x'_\xi}^2 + {y'_\xi}^2 \tag{2.68}$$

$$\begin{pmatrix} x'_\xi & y'_\xi \end{pmatrix}^T = \frac{1}{4}\left(|\vec{r}_J^{\,+}| + |\vec{r}_J^{\,-}|\right)\left(\frac{\vec{r}_J^{\,+}}{|\vec{r}_J^{\,+}|} - \frac{\vec{r}_J^{\,-}}{|\vec{r}_J^{\,-}|}\right) \tag{2.69}$$

While Equation 2.67 is preferred to calculate η derivatives near the body, the original method of computing these quantities (Equation 2.12) should still be restored away from the body, which can be achieved by

$$\begin{pmatrix} x_\eta & y_\eta \end{pmatrix}^T = (1 - \nu_k)\begin{pmatrix} x_\eta^o & y_\eta^o \end{pmatrix}^T + \nu_k \begin{pmatrix} x'_\eta & y'_\eta \end{pmatrix}^T \tag{2.70}$$

where $\nu_k = 2^{1-k}$ and x_η^o, y_η^o are values calculated in Equation 2.12. These modified derivatives can be computed as if their neighbor points of the corner are equally spaced on the two sides of the corner. The result of applying this procedure to a concave corner is shown in Figure 15 (b). Similar equations can also be obtained for surface grids and volume grids.

2.7 Sample Results

Several computational examples are presented in this section to demonstrate the utility of hyperbolic grid generation in two dimensions, including on planar and curved surfaces and hyperbolic grid generation in three dimensions.

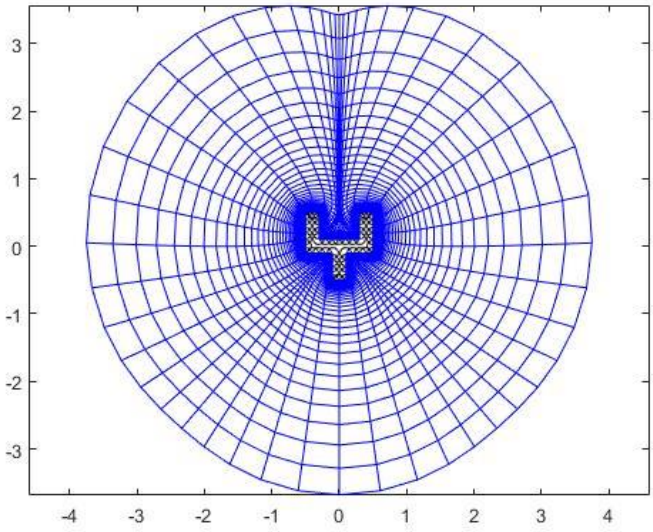


Figure 16: Case 1. Hyperbolic grid around Y-shaped body.

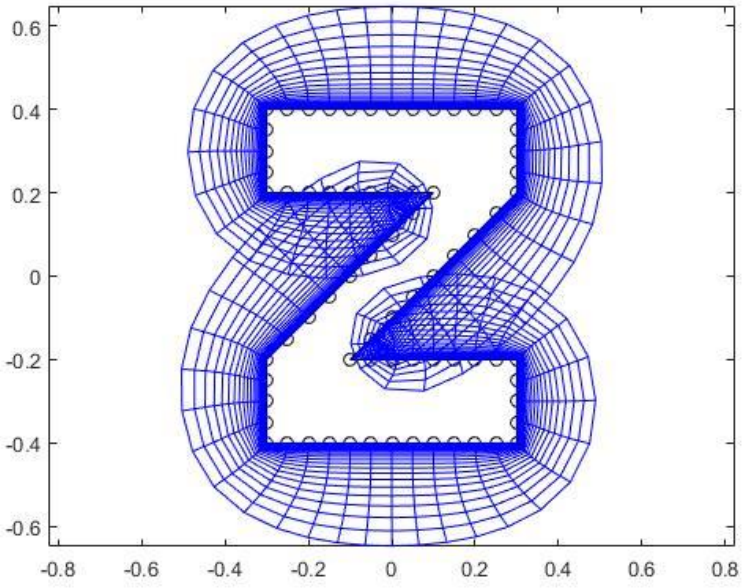


Figure 17: Case 2. Hyperbolic grid around Z-shaped body

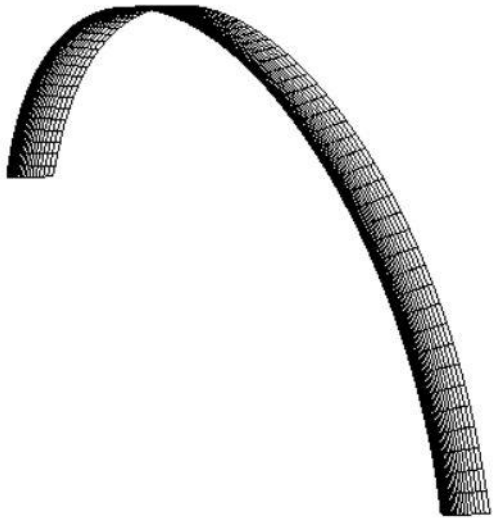


Figure 18: Case 3. Hyperbolic surface grid starting from a curve.

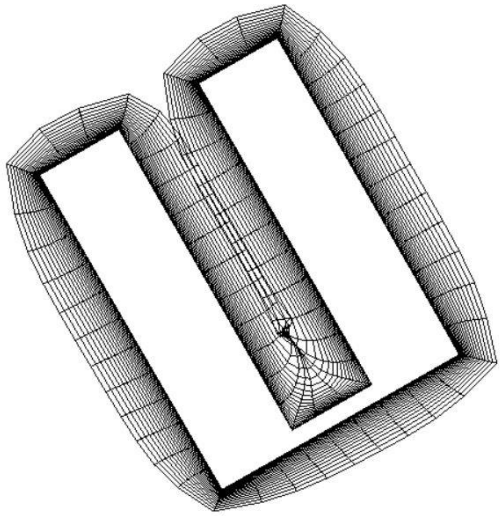
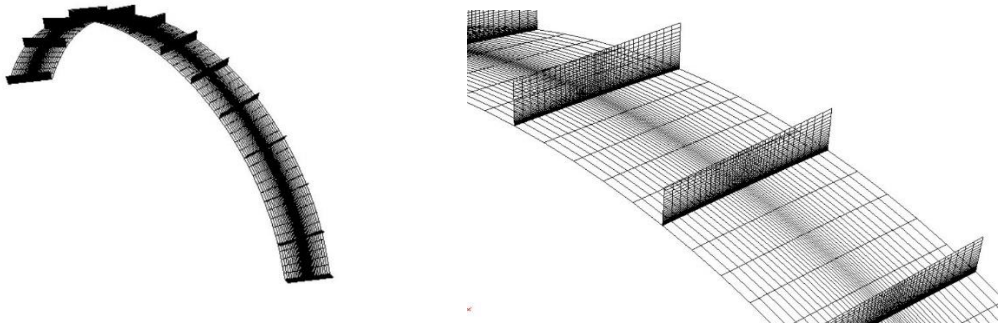
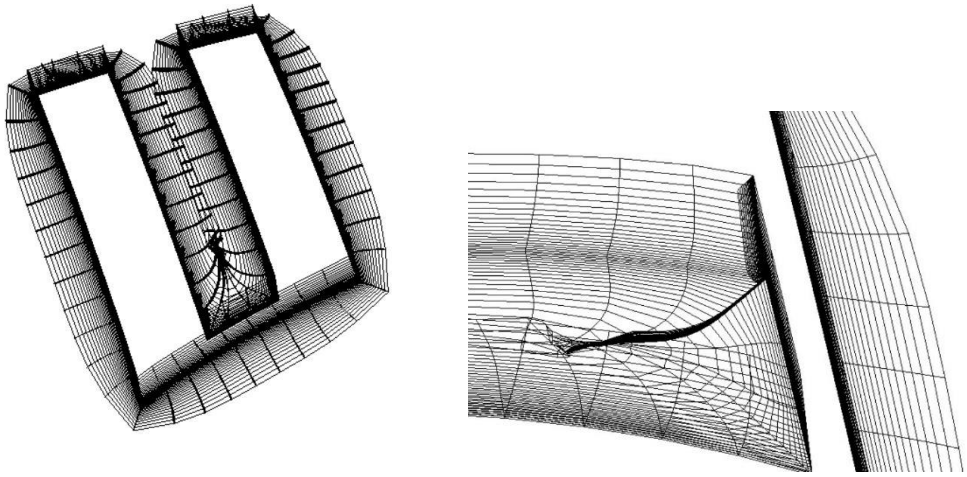


Figure 19: Case 4. Hyperbolic surface grid around a U-shaped body.



(a) Far field view (b) Close-up view

Figure 20: Case 5. Slice of hyperbolic volume grid around a U-shaped body.



(a) Far-field view (b) Close-up view at concave corner

Figure 21: Case 6. Slices of hyperbolic volume grid starting from a curve.

From the cases listed above, it can be seen that hyperbolic grid generation can work well in some cases (Case 1, 3, and 5). However, there are also some cases where grid lines overlap (Case 2, 4, and 6). These cases are not desirable and may lead to failure in computational fluid dynamics. User-defined parameters are selected as the default values mentioned above in these cases. Grid results will be different if these parameters are changed. But it may take some effort

to find one good set of parameters for a particular case. The computational time for these cases is listed in Table 3. As mentioned above, the computational speed is significantly fast to generate the hyperbolic grids.

Table 3: Computational time of grid generation using the hyperbolic scheme

Case number	Name	Grid type	Grid size	Computational time (s)
1	Y-shaped body	Surface grid	45×48	0.007
2	Z-shaped body*	Planar grid	31×36	0.003
3	Curve	Surface grid	21×62	0.007
4	U-shaped body*	Surface grid	21×66	0.031
5	Curve	Volume grid	41×62×21	0.243
6	U-shaped body*	Volume grid	41×66×21	0.264

Note: Computational time may be different on different software and operating system. The first three cases are compiled and run in Clion software from JetBrains, and the last four cases are compiled are run in x64 Native Tools Command Prompt for VS 2019 from Visual Studio on a laptop with Windows 10 operating system, Intel(R) Core i7-7700HQ CPU 2.80 GHz processor, and 16GB RAM.

It should be mentioned that the results marked with an asterisk (hyperbolic grid generation result for cases 2, 4, and 6) are not desirable because the grids will overlap in some places.

2.8 Comments for hyperbolic grid generation scheme

Based on the discussion above, we can draw the conclusion that using the hyperbolic technique to generate grids is very fast. However, the hyperbolic grids are not guaranteed to work every time because they may cluster or overlap in the far-field when dealing with concave corners. Otherwise, the user needs to change the parameter such as smoothing coefficients or implicit smoothing factors for different cases to get good grids, which requires a significant amount of knowledge and experience of the hyperbolic grid generation.

3. Elliptic Grid Generation

Unlike the hyperbolic technique, the mesh generated by elliptic grid generation is solved simultaneously by given point information on the boundary. The essence of elliptic grid generation is a mapping between the physical space to the computational domain, as shown in Figure 22. The primary motivation to use this method is that grids generated by elliptic methods tend to be smooth and the elliptic method works very well for complex geometry.

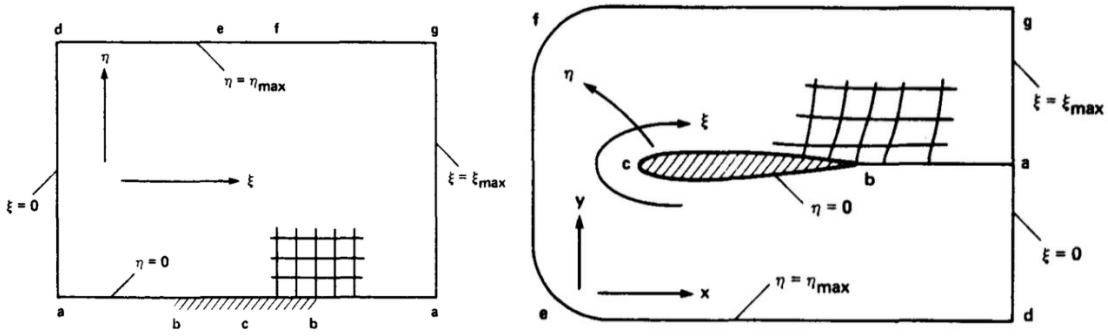
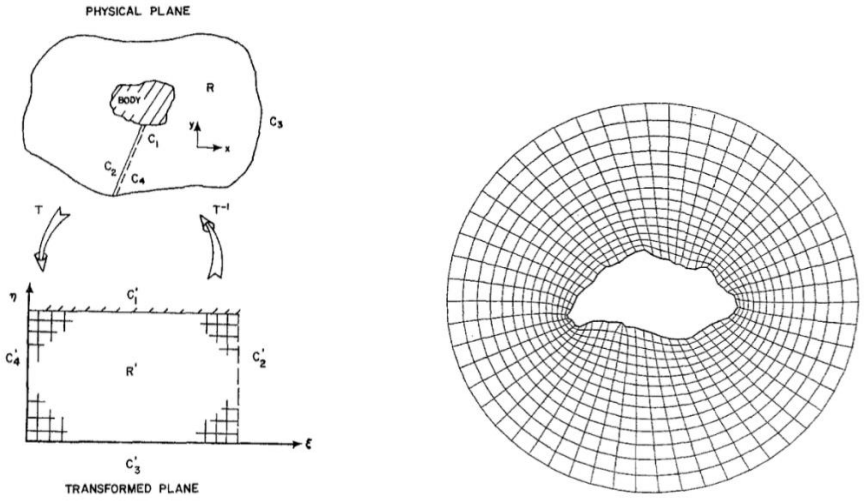


Figure 22: Comparison between computational space (left) and physical space (right). [30]

3.1 Introduction

The embryo of the elliptic grid generation scheme came from Joe G. Thompson [31], who talks about the numerical generation of body-fitted curvilinear coordinate systems in 1974. He was the first person who systematically introduced that a numerical generation of a general curvilinear coordinate system can be done automatically, regardless of the shape of boundaries. He used the Laplace equation to transform any shape in the physical plane into a rectangle transformed plane, as shown in Figure 23 (a). His scheme could avoid the interpolations between grid points, which is very important for boundaries with strong curvature or slope discontinuities

because interpolation between grid points not coincident with the boundaries is inaccurate. However, his scheme cannot control space or orthogonality near the inner boundary. From Figure 23 (b), it can be seen that the cells are not orthogonal at all.



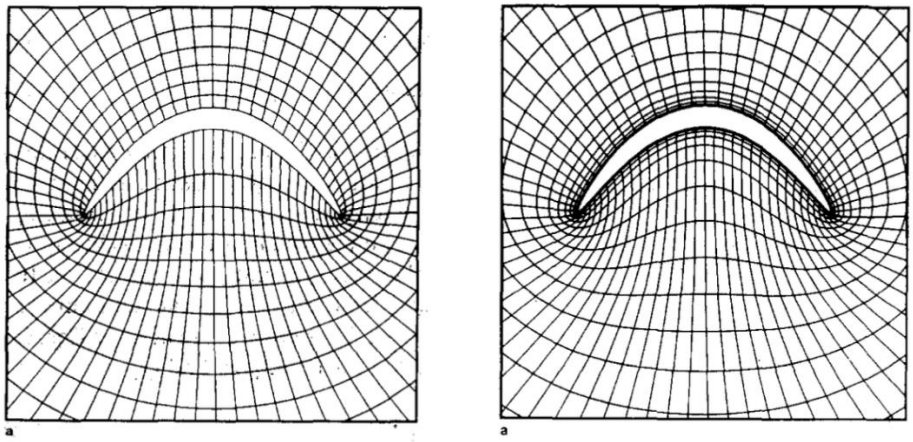
(a) Field transformation-single body (b) Grids near a rock shape

Figure 23: Grid Generation of body-fitted curvilinear. [31]

In 1977, the author of the GRAPE scheme, Reese L. Sorenson and Joseph L. Steger [32], developed an algorithm that can simplify clustering mesh points near the inner boundary. They added a source term to the right-hand side of Poisson equations. They derived the equations by setting source values to make points clustering near the inner boundary at first, then did reversed derivation on formulas. They use the Newton-Raphson method to iterate the value of source terms by manually setting the spacing between grid lines along the η direction, which is relatively slow.

Two years later, J.L. Steger and R.L. Sorenson [33] improved their scheme in 1979. They made significant progress this time. In earlier schemes, the values of source terms were set manually. They computed all values of each layer to get what they needed to cluster points near

the inner boundary, which can help get the source term. In the scheme of 1979, the derivatives along η direction on the inner boundary can be achieved by solving the general spacing equation, and the values of source terms P and Q on the inner boundary can be calculated. Therefore, all values of source terms in a whole field can be obtained by an exponential decay. The result of the scheme is shown in Figure 24 (b). The problem with this algorithm is the same as the previous one. Although they could compute source terms P and Q automatically, orthogonality on the inner boundary can still not be guaranteed.



(a) Grid without spacing control (b) Grid with spacing control

Figure 24: Grid Generation of body-fitted curvilinear. [33]

One year later, R.L. Sorenson [30] came up with the early version of the GRAPE scheme, which is short for "Generate Two-Dimensional Grids About Airfoils and Other Shapes by the Use of Poisson's Equation." By using this scheme, the orthogonality on the inner boundary can also be guaranteed besides spacing controlling from previous schemes. Sorenson added angle terms $cos\theta$ and $sin\theta$ into previous $x_\eta$and y_η terms, where θ is specified by users. Generally, it is set to 90 degrees to achieve orthogonality on the inner boundary.

In 1988, Reese L. Sorenson [34] finally developed a scheme in three dimensions for GRAPE. He rewrote the equations so that they were compatible in 3D. The critical idea in 3D does not have many differences compared to 2D. The sources of the inner boundary are calculated then exponentially decayed into the field. However, the algorithm became more complicated with the increment of the number of surfaces. The algorithm can solve the problems in 3D by using Poisson's equation by splitting the whole space into different zones then the grid could be drawn. But the effort for dividing zones and computing elliptic equations in each zone is tough. Sorenson [35] also wrote a handbook to illustrate using the 3D GRAPE generator coded by Fortran.

After Sorensen set up the theory of GRAPE, many other people studied it. Sungcho Kim [36] developed new source terms in 2000. Unlike Steger and Sorenson, Kim evaluated the control functions by solving the governing equations simultaneously, and the grid spacing is controlled in the nearest body surface region. His idea is that he assumed the application of the method is used on the planes perpendicular to the main flow direction coordinate. This type of main flow direction co-ordinate can be assumed to be the function of only one coordinate in the computational domain. This means for the vector $\vec{r}$, x coordinate can be assumed as $x(\xi)$. By doing this, the original equation can easily be transformed into a quasi-three-dimensional form, and the three source terms can easily be acquired. Also, Kim mentioned that the grid orthogonality was improved by 40 percent compared with the initial grid. This method works well in certain flows, but it does not fit for general cases as it needs the assumption.

In the original paper, there is one group of parameters called decay factors in the GRAPE scheme. They are set to be constant in the original scheme, which may fail in some instances or take a long time manually setting to get the ultimate value. In 2003, Upender K. Kaul [37]

developed a scheme that set a new boundary constraint for elliptic partial differential equations. These constraints are derived using Green's Theorem based on the conservation of thermal energy near boundaries. For the previous scheme, constant values for the decay factor work well in zones near a small extent of clustering of spacing control. When going to large gradients of clustering near boundaries, it may fail or not be very economical to get the appropriate values. In Kaul's method, these decay factors are set to be a function of coordinates. Then he uses Neumann type boundary constraints to fix them. By this, the values of those decay factors are obtained automatically without any sources. He uses the knowledge of thermodynamics, which has a deep meaning in elliptic grid generation. The method is further applied in 3D [38]. Also, the scheme is applied to certain physical cases.

The most important term in the elliptic grid generation scheme is the control function — source terms. It determines the quality of the grids directly. In 2004, S.H. Lee and B.K. Soni [39] developed a new way to calculate the second derivative terms of source terms. In the previous scheme, the second derivative terms $\vec{r}_{\eta\eta}$ are related to the location of grid points in the first, second and third layer and the first order derivatives $\vec{r}_{\eta}$. Lee and Soni derived the equations in their tensor form and rewrote the source functions. Therefore, the second derivative terms $\vec{r}_{\eta\eta}$ are only based on the first-order derivative terms $\vec{r}_{\eta}$. When trying to discretize the second-order derivative terms normal to the boundary involved in terms of control functions, it was found that the quality of a grid depends on the first order derivatives $\vec{r}_{\eta}$ from the second derivative equations because it is the only term that can be iterative through converge process. Based on this, a 7-point finite analytic scheme was derived. This new method can help make a smoothing connection between the so-called boundaries of blocks in algebraic systems.

In 2009, Vianey Villamizar and Sebastian Acosta [40] discovered some exciting aspects of source terms and developed two systems. One is to generate grids with near-uniform cell areas. We have known that Φ and Ψ (or written in P and Q) are the important source terms in elliptic grid generation. Villamizar and Acosta discovered that an increment of the Φ -value at a given grid point offers a local displacement of the corresponding η-curves in the direction of increasing η. And it is the reversed process that the grid line will move inwards in decreasing η direction if the value of Φ goes down. This kind of movement is also related to the magnitude of value Φ changes. The same thing happens to source Ψ and the movement of grid lines in ξ direction, shown in Figure 25. Figure 25 a represents grid lines with given values of Ψ and Φ. Figure 25 b represents that the value of Φ has been changed. The value of Ψ increasing leads to a movement outwards of ξ direction in Figure 24 c.

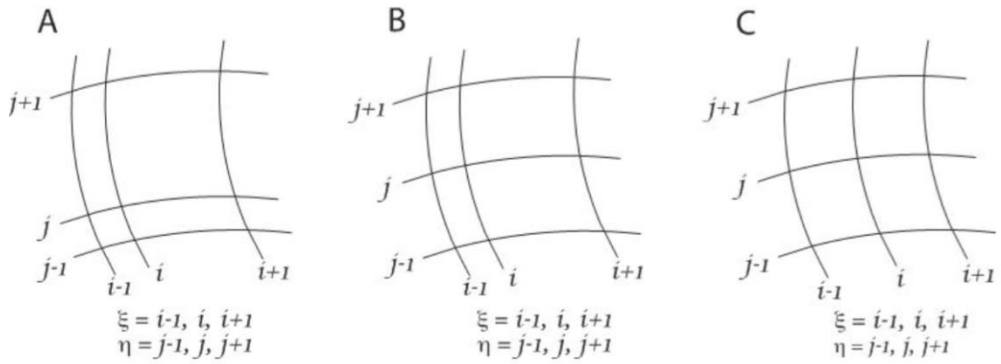


Figure 25: Making grids with uniform area cells. [40]

Villamizar and Sebastian also developed a way to get a nearly uniform cell area. They found that the area of a cell region R is approximately equal to the determinant of Jacobian. Based on this theory, Villamizar and Sebastian used some approximation for the values in the governing equations and used a numerical iterative method to solve them. With the check of cell size, smoothness and orthogonality have significantly improved compared with initial algebraic

mesh. The method is fascinating and has a meaningful aspect of studying the source term. The problem is that it can only achieve a uniform grid, which does not have the spacing clustering near the inner boundary as we need.

Many researchers are still focusing on studying source terms in the GRAPE scheme in recent years. W. Wenli, Z. Pei, Y. l. Liu [41] has developed a new method to determine the source terms (P and Q) of Poisson equations. The basic idea of determining the functions P and Q is based on two conditions:

- The boundary points are selected as desired.
- The constraint that the transverse coordinate curves be orthogonal to the boundary is imposed.

After applying the new source terms, it is much easier to deal with irregular solid boundary conditions. The generated grid can adapt to the change of physical quantities, which leads to reasonable accuracy in computational fluid dynamics. This method comes from Thompson's method and only slightly changes the original equation. The degree of discrepancy between the two methods is the problem of this method.

During the gap of GRAPE development, Dale A. Anderson [42] developed a new scheme that combines equidistribution schemes, Poisson generators, and adaptive grids. In the physical domain, the generated grid needs to be adjusted to position points where need to be refined to get better resolution in regions or to provide some decrement in global error. Therefore, an equidistribution scheme is a common way to generate an adaptive mesh. Anderson found that a transformed form of GRAPE equations was very similar to equidistribution expressions in differential form. After substituting source terms P, Q into the equidistribution scheme, a revised

elliptic generator can be used to get an adaptive mesh. The scheme is very economical. Just a trivial change can transform grid generators into solution-adaptive schemes.

Based on Anderson's scheme, Y.N. Jeng and Y.C. Lou [43] developed a desired grid stretching over the smooth region during initial grid system generation and before grid adaptation is performed. They rewrote the weighting form in Anderson's method and used a term M_{ij} to represent the measurement of a physical variable's variation, and they combined the scheme that could achieve boundary grid control on all the boundaries so that the requirement of stretching grid points over regions of the smooth solution is imposed during initial grid generation.

3.2 Governing Equations of Elliptic Planar Grid Generation (GRAPE)

In two-dimensional elliptic grid generation, the mapping function, following Thompson [32], are required to satisfy the Poisson equations

$$\xi_{xx} + \xi_{yy} = P \tag{3.1a}$$

$$\eta_{xx} + \eta_{yy} = Q \tag{3.1b}$$

And the transformed equations between computational space and physical space is given in Equation 3.2

$$\xi_x = \frac{y_\eta}{J} \tag{3.2a}$$

$$\xi_y = -\frac{x_\eta}{J} \tag{3.2b}$$

$$\eta_x = -\frac{y_\xi}{J} \tag{3.2c}$$

$$\eta_y = \frac{x_\xi}{J} \tag{3.2d}$$

$$J = x_\xi y_\eta - x_\eta y_\xi \tag{3.2e}$$

Transforming the above equations in computational space yields a set of elliptical PDEs

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = -J^2\left(P x_\xi + Q x_\eta\right) \tag{3.3a}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -J^2\left(P y_\xi + Q y_\eta\right) \tag{3.3b}$$

where

$$\alpha = {x_\eta}^2 + {y_\eta}^2 \tag{3.4a}$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta \tag{3.4b}$$

$$\gamma = {x_\xi}^2 + {y_\xi}^2 \tag{3.4c}$$

The grid can be generated by solving the equations above with a particular choice of source terms P and Q and a particular set of boundary conditions. If $P = Q = 0$, Equations 3.3 becomes Laplace equations, which are only able to generate basic grids without any orthogonality or spacing control, which is shown in Figure 26.

(a) Far field view (b) Close-up view

Figure 26: Grids around a square shape using Laplace equations.

Therefore, picking source terms P and Q to assure orthogonality and spacing control becomes a problem. From Reese L. Sorenson [30], the source terms can be set as

$$P = p(\xi)e^{-a\eta} + r(\xi)e^{-c(\eta_{max}-\eta)} \tag{3.4a}$$

$$Q = q(\xi)e^{-b\eta} + s(\xi)e^{-d(\eta_{max}-\eta)} \tag{3.4b}$$

where p, q, r, and s are the values of source terms defined on body curves and outer boundary, which can be obtained by the boundary conditions. Details about getting these values are discussed in Chapter 3.4. Factors a, b, c, and d are the four positive constants that determine the rate of the exponential decay from the boundary and the interior grid lines. Small values (e.g., 0.2) lead to the slow decay of the source terms, which means that the geometry of the initial grid lines and outer boundary will affect deeper into the grid field. However, small values will also cause the problem that the numerical convergence will be more difficult at the same time. Large values (e.g., 0.7) have the opposite effects. If the values are too large, the effect of the source terms will be negligible, and the governing equation becomes the Laplace equation inside the grid region. The default value of the exponential decay factors is 0.4.

3.3 Governing Equations of Elliptic Surface Grid Generation

The governing equation for elliptic grid generation on a curved surface is an extension based on that in two-dimension, which is given as

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = -J^2\left(Px_{\xi} + Qx_{\eta}\right) \tag{3.5a}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -J^2\left(Py_{\xi} + Qy_{\eta}\right) \tag{3.5b}$$

$$\alpha z_{\xi\xi} - 2\beta z_{\xi\eta} + \gamma z_{\eta\eta} = -J^2\left(Pz_{\xi} + Qz_{\eta}\right) \tag{3.5c}$$

and the Jacobian matrix becomes to

$$J = \begin{vmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ \hat{n}_1 & \hat{n}_2 & \hat{n}_3 \end{vmatrix} \tag{3.6}$$

where $\hat{n} = (\hat{n}_1, \hat{n}_2, \hat{n}_3)^T$ is the local unit surface normal. For a parameterized surface $\begin{pmatrix} x \\ y \\ z \end{pmatrix} =$

$f(u \quad v)$, the surface normal can be computed as

$$(\hat{n}_1, \hat{n}_2, \hat{n}_3)^T = \frac{\vec{n}}{|\vec{n}|} = \frac{1}{|\vec{n}|}\begin{pmatrix} y_u z_v - z_u y_v \\ z_u x_v - x_u z_v \\ x_u y_v - y_u x_v \end{pmatrix} \tag{3.7a}$$

$$|\vec{n}| = (y_u z_v - z_u y_v)^2 + (z_u x_v - x_u z_v)^2 + (x_u y_v - y_u x_v)^2 \tag{3.7b}$$

The choice of inhomogeneous terms P and Q for elliptic surface grid generation will be the same as for elliptic planar grid generation.

3.4 Governing Equations of Elliptic Volume Grid Generation

For a three-dimensional elliptic grid, it is required that the mapping between ξ, η, ζ, and x, y, z satisfy the Poisson equation (Sorenson [33])

$$\xi_{xx} + \xi_{yy} + \xi_{zz} = P \tag{3.8a}$$

$$\eta_{xx} + \eta_{yy} + \eta_{zz} = Q \tag{3.8b}$$

$$\zeta_{xx} + \zeta_{yy} + \zeta_{zz} = R \tag{3.8c}$$

Equations 3.8 are solved using transform equation, which becomes

$$\alpha_{11}\vec{r}_{\xi\xi} + \alpha_{22}\vec{r}_{\eta\eta} + \alpha_{33}\vec{r}_{\zeta\zeta} + 2\left(\alpha_{12}\vec{r}_{\xi\eta} + \alpha_{13}\vec{r}_{\xi\zeta} + \alpha_{23}\vec{r}_{\eta\zeta}\right) = -J^2\left(P\vec{r}_\xi + Q\vec{r}_\eta + R\vec{r}_\zeta\right) \tag{3.9a}$$

where

$$\vec{r} = [x, y, z]^T \tag{3.9b}$$

$$\alpha_{ij} = \sum_{m=1}^{3} \gamma_{mi}\gamma_{mj} \tag{3.9c}$$

γ_{ij} is ij[th] cofactor of the matrix M

$$\mathrm{M} = \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix} \tag{3.9d}$$

and J is the determinant of M, i.e.

$$\alpha_{11} = \left(y_\eta z_\zeta - z_\eta y_\zeta\right)^2 + \left(z_\eta x_\zeta - x_\eta z_\zeta\right)^2 + \left(x_\eta y_\zeta - y_\eta x_\zeta\right)^2 \tag{3.9e}$$

$$\alpha_{22} = \left(y_\xi z_\zeta - z_\xi y_\zeta\right)^2 + \left(z_\xi x_\zeta - x_\xi z_\zeta\right)^2 + \left(x_\xi y_\zeta - y_\xi x_\zeta\right)^2 \tag{3.9f}$$

$$\alpha_{33} = \left(y_\xi z_\eta - z_\xi y_\eta\right)^2 + \left(z_\xi x_\eta - x_\xi z_\eta\right)^2 + \left(x_\xi y_\eta - y_\xi x_\eta\right)^2 \tag{3.9g}$$

$$\begin{aligned} \alpha_{12} = &-\left(y_\eta z_\zeta - z_\eta y_\zeta\right)\left(y_\xi z_\zeta - z_\xi y_\zeta\right) - \left(z_\eta x_\zeta - x_\eta z_\zeta\right)\left(z_\xi x_\zeta - x_\xi z_\zeta\right) \\ &-\left(x_\eta y_\zeta - y_\eta x_\zeta\right)\left(x_\xi y_\zeta - y_\xi x_\zeta\right) \end{aligned} \tag{3.9h}$$

$$\begin{aligned} \alpha_{13} = &\left(y_\eta z_\zeta - z_\eta y_\zeta\right)\left(y_\xi z_\eta - z_\xi y_\eta\right) + \left(z_\eta x_\zeta - x_\eta z_\zeta\right)\left(z_\xi x_\eta - x_\xi z_\eta\right) \\ &+\left(x_\eta y_\zeta - y_\eta x_\zeta\right)\left(x_\xi y_\eta - y_\xi x_\eta\right) \end{aligned} \tag{3.9i}$$

$$\begin{aligned} \alpha_{23} = &-\left(y_\xi z_\zeta - z_\xi y_\zeta\right)\left(y_\xi z_\eta - z_\xi y_\eta\right) - \left(z_\xi x_\zeta - x_\xi z_\zeta\right)\left(z_\xi x_\eta - x_\xi z_\eta\right) \\ &-\left(x_\xi y_\zeta - y_\xi x_\zeta\right)\left(x_\xi y_\eta - y_\xi x_\eta\right) \end{aligned} \tag{3.9j}$$

In this elliptic grid generation approach, the choice of inhomogeneous terms will affect the grid results. From Sorenson's paper [33], the term P is chosen as

$$P(\xi,\eta,\zeta) = \sum_{n=1}^{6} P_n(\xi,\eta,\zeta) \tag{3.10}$$

where n refers to the face number of the computational cube, $1 \leq n \leq 6$. Face 1, for example, is the face where ξ is fixed at zero. And term P_1 is given as

$$P_1(\xi,\eta,\zeta) = p_1(\eta,\zeta)e^{-a\xi} \tag{3.11}$$

where a is a positive constant. It can be seen that on face 1, the exponential factor becomes one, and terms P_2 through P_6 are almost of no effect. Therefore, on face 1, inhomogeneous term P will reduce to just p_1. Term P_2 through P_6 are found similarly in order to get the source terms on the other computational cubic faces. Terms Q and R are identical in form.

3.5 Boundary Conditions for Elliptic Grid Generation

3.5.1 Boundary Conditions for Elliptic Planar Grid Generation

The system of partial differential equations talked in Chapter 3.2 can be solved for a specific set of boundary conditions. First, if we want to control the spacing neat the body curve, we can specify the distance as

$$s_\eta|_{k=1} = \left(x_\eta{}^2 + y_\eta{}^2\right)_{k=1}^{1/2} \tag{3.12}$$

The second geometric requirement is the angle of the intersection between the body and the ξ = constant line. This angle is also able to be defined by users. Generally, CFD researchers want grid lines to be orthogonal to the body curve that the angle should be set as $90°$. From the definition of the dot product, the angle θ is defined by

$$(\nabla\xi \cdot \nabla\eta)_{k=1} = (|\nabla\xi||\nabla\eta|cos\theta)_{k=1} \tag{3.13}$$

Expanding, we have

$$\left(\xi_x\eta_x + \xi_y\eta_y\right)_{k=1} = \left[\left(\xi_x^2 + \xi_y^2\right)^{\frac{1}{2}}\left(\eta_x^2 + \eta_y^2\right)^{\frac{1}{2}}cos\theta\right]_{k=1} \tag{3.14}$$

Applying the transformation Equations 3.2 to Equation 3.14 yields

$$\left(-y_\eta y_\xi - x_\eta x_\xi\right)_{k=1} = \left[\left(y_\eta^2 + x_\eta^2\right)^{\frac{1}{2}}\left(y_\xi^2 + x_\xi^2\right)^{\frac{1}{2}}cos\theta\right]_{k=1} \tag{3.15}$$

Combining those two Equations 3.12 and 3.15, we can find the derivatives x_η, y_η at the inner boundary as.

$$x_\eta|_{k=1} = \left[\frac{s_\eta\left(x_\xi cos\theta - y_\xi sin\theta\right)}{\left({x_\xi}^2 + {y_\xi}^2\right)^{1/2}}\right]_{k=1} \tag{3.16a}$$

$$y_\eta|_{k=1} = \left[\frac{s_\eta\left(-y_\xi cos\theta + x_\xi sin\theta\right)}{\left({x_\xi}^2 + {y_\xi}^2\right)^{1/2}}\right]_{k=1} \tag{3.16b}$$

Then on the body curve, Equations 3.4 becomes: ($k = \eta + 1$, $j = \xi + 1$)

$$P(\xi, 0) = p(\xi) \tag{3.17a}$$

$$Q(\xi, 0) = q(\xi) \tag{3.17b}$$

Solving Equations 3.3 and 3.17, we have

$$p(\xi) = \left(\frac{y_\eta R_1 - x_\eta R_2}{J}\right)_{k=1} \tag{3.18a}$$

$$q(\xi) = \left(\frac{-y_\xi R_1 + x_\xi R_2}{J}\right)_{k=1} \tag{3.18b}$$

where

$$R_1 = \left[\frac{-\left(\alpha x_{\eta\eta} - 2\beta x_{\xi\eta} + \gamma x_{\xi\xi}\right)}{J^2}\right]_{k=1} \tag{3.18c}$$

$$R_2 = \left[\frac{-\left(\alpha y_{\eta\eta} - 2\beta y_{\xi\eta} + \gamma y_{\xi\xi}\right)}{J^2}\right]_{k=1} \tag{3.18d}$$

Similar equations can be obtained for $r(\xi)$ and $s(\xi)$ at the outer boundary where $k = k_{max}$.

In practice, in order to increase the stability of the algorithm, relaxation parameters and limitation factors, given by Reese L. Sorenson [30], have been added to solve the source terms

$$p^{(n+1)} = p^{(n)} + SIGN\left[min\left(\omega_p\left|p - p^{(n)}\right|, p_{lim} max\left(\left|p^{(n)}\right|, 1\right)\right), p - p^{(n)}\right] \tag{3.19}$$

where the SIGN function gives the magnitude of the first argument and the sign of the second argument. p_{lim} is the limitation factor that controls the change of P in the first several iterations. ω_p is the relaxation parameter that controls the change of P in the rest iterations, which is restricted as $0 < \omega_p < 1$. Similar procedures are used for $q^{(n+1)}$, $r^{(n+1)}$ and $s^{(n+1)}$. The default value of the limitation factor is 1, and the default value of the relaxation parameter is 0.3. Increasing these factors may lead to more rapid convergence, but it may also cause numerical instability, and the results will blow up. If such a case happens, smaller values of the limitation factors and the relaxation parameters should be selected to recalculate the grids.

It should also be noticed that the source terms p, q, r, and s calculated using the equations above would be significantly large when dealing with points at sharp corners, which would cause instability. At such points, the computed value of the source terms p, q, r, and s should be replaced by the average values on either side of the points. This procedure would give smaller values of source terms at sharp corners, and thus a more stable grid result can be achieved. The

criteria of the sharp corner, in this case, is defined as the internal angle is smaller than $\frac{2\pi}{3}$ or larger than $\frac{4\pi}{3}$.

In order to calculate p, q, r, and s, it is necessary to get all the values of the derivatives appearing in Equations 3.18. At the inner and outer boundaries, x, y, and η are fixed while ξ varies. We can directly calculate x_ξ, y_ξ, $x_{\xi\xi}$ and $y_{\xi\xi}$ by finite differencing boundary points. x_η and y_η can be calculated using equation 3.16 with given θ and s_η as input. $x_{\xi\eta}$ and $y_{\xi\eta}$ can also be solved by differencing x_η and y_η with respect to ξ. The only unknown terms in Equations 3.18 are $x_{\eta\eta}$ and $y_{\eta\eta}$ which can be computed by Taylor expansion as

$$x_{\eta\eta}|_{k=1} = \frac{-7x|_{k=1} + 8x|_{k=2} - x|_{k=3}}{2(\Delta\eta)^2} - \frac{3x_\eta|_{k=1}}{\Delta\eta} \qquad (3.20a)$$

$$y_{\eta\eta}|_{k=1} = \frac{-7y|_{k=1} + 8y|_{k=2} - y|_{k=3}}{2(\Delta\eta)^2} - \frac{3y_\eta|_{k=1}}{\Delta\eta} \qquad (3.20b)$$

With all the equations given above, we can finally solve the Poisson equations iteratively. The general iterative procedure can be described as follows:

1. Give the body shape an appropriate outer boundary and an initial guess for all grid points by linear interpolation between inner and outer boundary grid points.
2. Set the value for θ, s_η, a, b, c, and d at body curve and outer boundary and compute first and second order derivates.
3. Use the initial setting or the previous results to compute x_η and y_η.
4. Compute p, q, r, and s using values got from step 1 and 2.
5. Compute source terms P and Q for all grid points.

6. Solve the partial differential Equation 3.3 numerically using one step successive line over-relaxation (SLOR) technique. Block tridiagonal matrix is generated for each line and is solved by the Thomas algorithm.

7. Repeat 2-6 until the error comes to the tolerance.

After adding the source terms, the result from the previous square shape example becomes as Figure 27

(a) Far field view (b) Close-up view

Figure 27: Grids around a square shape using Poisson equations.

From Figure 27, it can be seen that the orthogonality near the body curve has been achieved, and gridlines cluster towards the body curve. Therefore, the geometry requirements for CFD research have been satisfied.

The direction of lines in SLOR (step 6) is a topic that needs to be mentioned. One can use the lines in the ξ direction or the η direction, or even both directions (e.g., changing line direction in every other iteration), and the computational speed and stability might be different based on the choice of the direction. This phenomenon can be explained by analyzing the transformed Poisson equation. When taking the difference scheme for Equations 3.3, a tridiagonal matrix is made with the value $(-2 * (\alpha + \gamma))$ on the diagonal and $2 * \alpha$ off-diagonal if the lines run in ξ xi direction ($2 * \gamma$ if the lines run in η xi direction). If the aspect ratio on the

inner boundary is smaller than one, i.e., $\alpha < \gamma$, making equations along ξ direction will maximize diagonal dominance and thereby increase numerical stability compared with that along γ direction. However, smaller off-diagonal values also mean that the grid points can hardly be affected by their neighbor points, and thus more iterations are required for the equations to converge. Opposite effects occur for lines running in η xi direction. Several tests about convergence speed are computed in Table 4. It can be seen from this table that taking SLOR in η direction is significantly faster when the aspect ratio is less than unity, and the computational speed for SLOR in both directions lies in the middle.

Table 4: Convergence speed for different line directions in SLOR

Case number	Case name	Number of layers	Stretching ratio	Average aspect ratio	SLOR in ξ direction (s)	SLOR in η direction (s)	SLOR in both directions (s)
1	Y-shaped body	30	1.2	0.11	3.351	1.076	1.361
2	Y-shaped body	30	1.1	0.11	3.12	0.214	0.291
3	Y-shaped body	45	1.1	0.11	11.262	1.369	2.203
4	Y-shaped body (Fine grids)	30	1.1	0.21	6.193	1.019	1.33
5	Z-shaped body	30	1.1	0.19	2.112	0.178	0.304
6	Z-shaped body	45	1.1	0.19	9.217	0.553	0.847

Note: Cases are compiled and run in Clion software from JetBrains on a laptop with Windows 10 operating system, Intel(R) Core i7-7700HQ CPU 2.80 GHz processor, and 16GB RAM. Outer boundaries and points distribution along the boundaries are calculated based on the

hyperbolic grid generation result with the given stretching ratio. Grid spacing along the initial curve is reduced by half for fine grids (case 4).

Another statement of the elliptic grid generation scheme that can be put forward is that the computational time increases significantly as the grid size escalates. Comparing results in case 2 and case 3 (same for case 5 and case 6), we find that adding the number of grid layers by 50% will increase computational time several times. This phenomenon is quite different from the hyperbolic grid generation scheme, for which the computational time is almost linearly related to the number of grid layers. It will also take much more computational time if the number of grid points increases by comparing case 2 and 4.

3.5.2 Boundary Conditions for Elliptic Surface Grid Generation

For the grid generation on a curved surface, the equation for spacing control at the inner boundary is given as

$$s_\eta{}^2|_{k=1} = \left(x_\eta{}^2 + y_\eta{}^2 + z_\eta{}^2\right)|_{k=1} \tag{3.21}$$

The orthogonality gives

$$\left(x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta\right)|_{k=1} = 0 \tag{3.22}$$

Another constraint added for the surface grid is that the grid lines should be perpendicular to the surface normal

$$\left(\hat{n}_1 x_\eta + \hat{n}_2 y_\eta + \hat{n}_3 z_\eta\right)|_{k=1} = 0 \tag{3.23}$$

where $\hat{n} = (\hat{n}_1, \hat{n}_2, \hat{n}_3)^T$ is the local unit surface normal.

Combining Equation 3.22 and 3.23, we can get

$$\left[\left(x_\xi \hat{n}_3 - z_\xi \hat{n}_1\right)x_\eta\right]|_{k=1} = \left[\left(z_\xi \hat{n}_2 - y_\xi \hat{n}_3\right)y_\eta\right]|_{k=1} \tag{3.24a}$$

$$\left[\left(y_\xi \hat{n}_1 - x_\xi \hat{n}_2\right)y_\eta\right]|_{k=1} = \left[\left(x_\xi \hat{n}_3 - z_\xi \hat{n}_1\right)z_\eta\right]|_{k=1} \tag{3.24b}$$

$$\left[\left(z_\xi \hat{n}_2 - y_\xi \hat{n}_3\right)z_\eta\right]|_{k=1} = \left[\left(y_\xi \hat{n}_1 - x_\xi \hat{n}_2\right)x_\eta\right]|_{k=1} \tag{3.24c}$$

Taking Equation 3.24 into Equation 3.21, we have

$$x_\eta|_{k=1} = \left[\left(1 + \frac{\left(x_\xi \hat{n}_3 - z_\xi \hat{n}_1\right)^2 + \left(y_\xi \hat{n}_1 - x_\xi \hat{n}_2\right)^2}{\left(z_\xi \hat{n}_2 - y_\xi \hat{n}_3\right)^2}\right)^{-\frac{1}{2}} s_\eta\right]_{k=1} \tag{3.25a}$$

$$y_\eta|_{k=1} = \left[\left(1 + \frac{\left(y_\xi \hat{n}_1 - x_\xi \hat{n}_2\right)^2 + \left(z_\xi \hat{n}_2 - y_\xi \hat{n}_3\right)^2}{\left(x_\xi \hat{n}_3 - z_\xi \hat{n}_1\right)^2}\right)^{-\frac{1}{2}} s_\eta\right]_{k=1} \tag{3.25b}$$

$$z_\eta|_{k=1} = \left[\left(1 + \frac{\left(z_\xi \hat{n}_2 - y_\xi \hat{n}_3\right)^2 + \left(x_\xi \hat{n}_3 - z_\xi \hat{n}_1\right)^2}{\left(y_\xi \hat{n}_1 - x_\xi \hat{n}_2\right)^2}\right)^{-\frac{1}{2}} s_\eta\right]_{k=1} \tag{3.25c}$$

Like the elliptic planar grid generation, the governing equation of elliptic surface grid generation at the inner boundary ($\eta = 0$) is given as

$$P(\xi, 0) = p(\xi) \tag{3.26a}$$

$$Q(\xi, 0) = q(\xi) \tag{3.26b}$$

$p(\xi)$ and $q(\xi)$ can then be computed by taking all the derivatives into Equation 3.5. However, since there are three equations and only two unknown variables, the normal equation is also

applied here to get a set of results that best fit all three equations. A matrix form for solving $p(\xi)$ and $q(\xi)$ is given as

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \\ z_\xi & z_\eta \end{bmatrix}_{k=1} \begin{pmatrix} p(\xi) \\ q(\xi) \end{pmatrix} = -\left[\frac{1}{J^2} \begin{pmatrix} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} \\ \alpha z_{\xi\xi} - 2\beta z_{\xi\eta} + \gamma z_{\eta\eta} \end{pmatrix} \right]_{k=1} \tag{3.27}$$

Once $p(\xi)$ and $q(\xi)$ are calculated, they can be taken into Equation 3.5 to update grid points and derivatives by one step successive line over-relaxation (SLOR). The choice of line direction is the same as elliptic planar grid generation.

3.5.3 Boundary Conditions for Elliptic Volume Grid Generation

The geometry constraints for the three-dimensional elliptic grid generation are imposed on the lines in ζ direction which intersects face. First, the lines in ζ direction should be normal to the surface, i.e., normal to the lines in ξ and η direction. Second, the length along ζ direction must be controlled. These constrain at the inner boundary can be expressed in algebraic form as

$$\left(x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta\right)|_{k=1} = 0 \tag{3.28a}$$

$$\left(x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta\right)|_{k=1} = 0 \tag{3.28b}$$

$$s_\zeta{}^2|_{k=1} = \left(x_\zeta{}^2 + y_\zeta{}^2 + z_\zeta{}^2\right)|_{k=1} \tag{3.28c}$$

From these equations, expressions can be obtained using cofactors mentioned in Equation 3.9, which is given as

$$x_\zeta|_{k=1} = \frac{\gamma_{13} s_\zeta}{\pm\sqrt{\gamma_{13}^2 + \gamma_{23}^2 + \gamma_{33}^2}} \tag{3.29a}$$

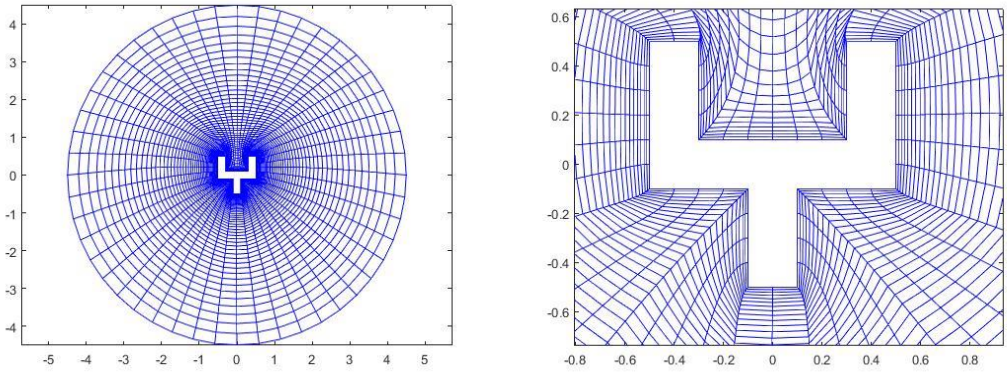
$$y_\zeta|_{k=1} = \frac{\gamma_{23} s_\zeta}{\pm\sqrt{\gamma_{13}^2 + \gamma_{23}^2 + \gamma_{33}^2}} \tag{3.29b}$$

$$z_\zeta|_{k=1} = \frac{\gamma_{33} s_\zeta}{\pm\sqrt{\gamma_{13}^2 + \gamma_{23}^2 + \gamma_{33}^2}} \tag{3.29c}$$

The positive sign for the radical is chosen for a right-handed coordinate system. This set of derivatives can be differenced with respect to ξ and η to obtain the $\xi\zeta$ and $\eta\zeta$ mixed second-partial derivatives. $\zeta\zeta$ second-partial derivative can be computed in the same form as Equation 3.20. These derivatives can be used to solve for p_1, q_1, and r_1 in Equation 3.11, and P_1, Q_1, and R_1 are then known. Terms P_n, Q_n, and R_n for $2 \leq n \leq 6$ are found similarly.

3.6 Sample Results

Several examples of elliptic grid generation in two dimensions, including on a planar face and on a curved surface, and elliptic grid generation in three dimensions are presented in this section. The first case uses points on a circle as outer boundary points, and the rest cases use the results from hyperbolic grid generation to calculate the outer boundary.



(a) Far field view (b) Close-up view

Figure 28: Case 1. Elliptic planar grid generation around Y-shaped body

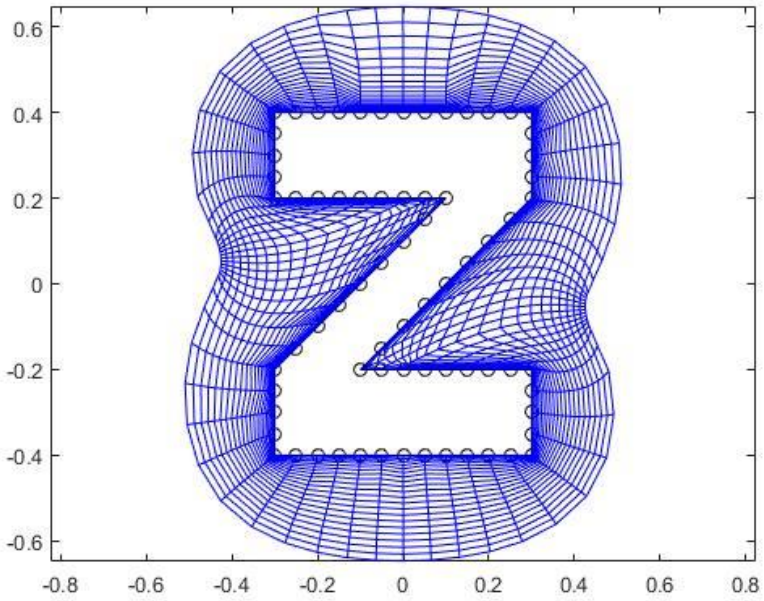


Figure 29: Case 2. Elliptic planar grid generation around Z-shaped body.

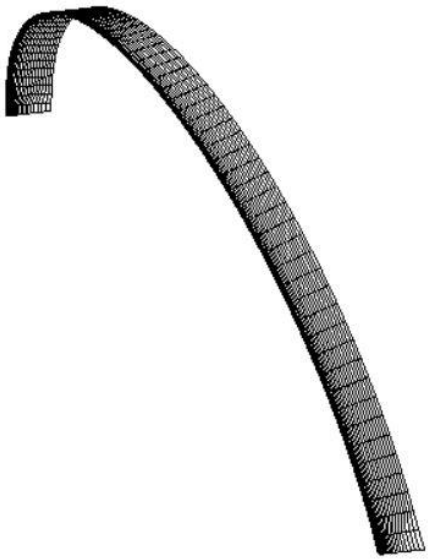


Figure 30: Case 3. Elliptic surface grid generation from a curve

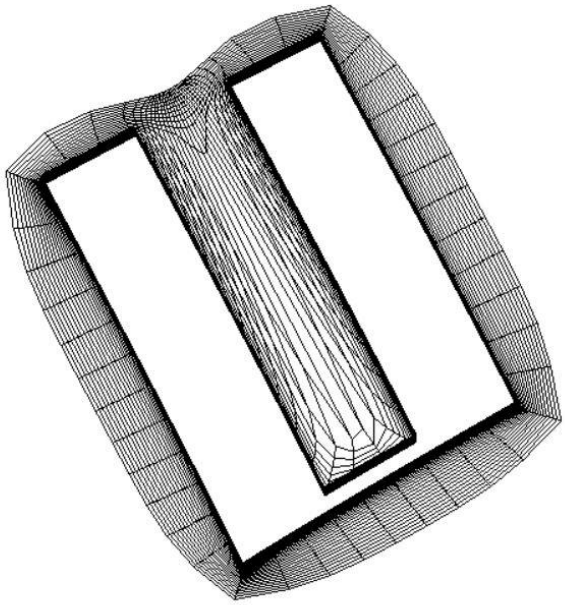
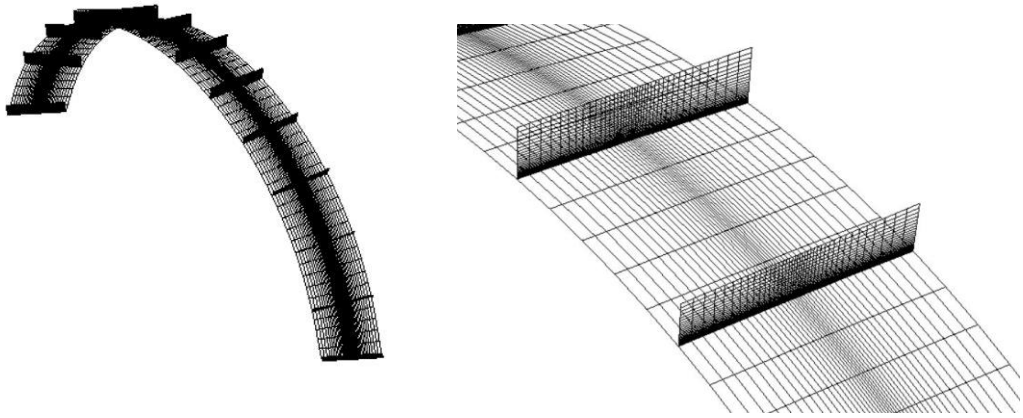
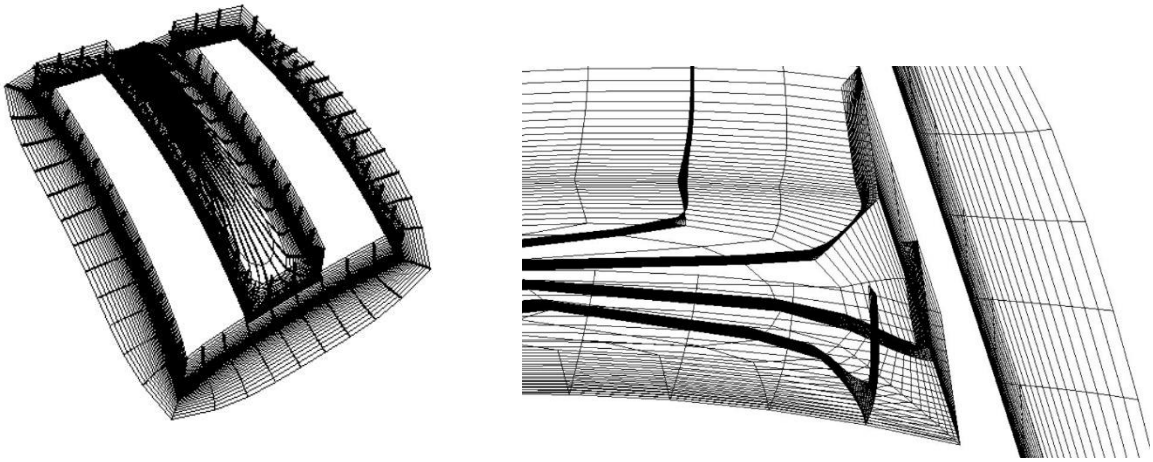


Figure 31: Case 4. Elliptic surface grid generation around U-shaped body



(a) Far field view (b) Close-up view

Figure 32: Case 5. Slices of elliptic volume grid starting from a curve.



(a) Far field view (b) Close-up view

Figure 33: Case 6. Slices of elliptic volume grid around U-shaped body

The computational time for these cases is listed in Table 5. It can be seen that the computational speed is much slower compared with that using the hyperbolic scheme. But the grid lines do not overlap in any of these cases.

Table 5: Computational time of grid generation using the elliptic scheme

Case number	Name	Grid type	Grid size	Computational time (s)
1	Y-shaped body	Planar grid	46×48	1.369
2	Z-shaped body	Planar grid	23×36	1.937
3	Curve	Volume grid	21×62	0.067
4	U-shaped body	Surface grid	21×66	2.471
5	Curve	Volume grid	41×62×21	2.500
6	U-shaped body	Volume grid	41×66×21	26.872

Note: Computational time may be different on different software and operating system. The first two cases are compiled run in Clion software from JetBrains, and the last four cases are compiled and run in x64 Native Tools Command Prompt for VS 2019 from Visual Studio on a laptop with Windows 10 operating system, Intel(R) Core i7-7700HQ CPU 2.80 GHz processor, and 16GB RAM.

3.6 Comments for elliptic grid generation

The main problem of the elliptic grid generation scheme is that the computational speed is relatively slow compared with hyperbolic grid generation because the grids are computed iteratively, and every grid point needs to be updated in each iteration. Several limitation factors are utilized to increase the stability of the scheme, but they will also reduce the speed at the same time. Besides, the problem of computational speed will be significantly amplified if the number of grid points and the number of grid layers are large.

Although using the elliptic technique to generate grids seems to work on any geometry shape, there are still some situations that can lead to convergence failure for the elliptic grid generation scheme in practice. The most likely reason for elliptic grid generation failure is that a physically impossible situation is made based on a given number of layers, initial spacing at the inner and outer boundary, exponential factors, and the size of the outer boundary. In other words, the region defined by the outer boundary is too big for the grids, which have small initial spacing and exponential factors, to fill in within the few numbers of layers. In this case, more layers of grids, larger initial spacing, larger exponential factors, the smaller size of the outer boundary, or any combination of these would be suggested to deal with the problem. On the other hand, too many layers of grid lines and too small size of outer boundary will invalidate the definition of initial spacing and exponential factors, leaving a nearly equally spaced grid. Therefore, these parameters should be appropriately selected in order to generate a useful grid.

4. Combined Scheme

The idea of the combined scheme is to take advantage of both grid generation techniques, i.e., the grid can be generated fast as the hyperbolic scheme, and the grid can work on any boundary shape as the elliptic scheme. Based on the discussion in Chapter 2, we can find the main problems of the hyperbolic grid generation. Therefore, we come up with a new method that uses hyperbolic grid generation to get a profile of the grid and makes some corrections at the places where hyperbolic grid generation works poorly, i.e., where grid overlapping or grid clustering occurs by the elliptic grid generation method, which is introduced in Chapter 3.

The general procedure of the combined scheme is:

1. Generate grids with the hyperbolic scheme to get a mesh with “bad” points

2. Detect the “bad” points and cut them out

3. Generate an outer boundary and generate grids in the revised area by elliptic grid generation method

4. Revise the outer boundary and redo the grid generation until converge criterion is satisfied

5. Revise the intersection regions between hyperbolic and elliptic grids

By using the combined scheme, a default smoothing coefficient for hyperbolic grid generation can be applied because it is unnecessary to make all grids work. The definition of the outer boundary for the elliptic grid generation method is also not required since we can calculate it with the help of the result from hyperbolic grid generation. Therefore, user input about the procedure for grid generation can be eliminated.

4.1 Bad Points Detection

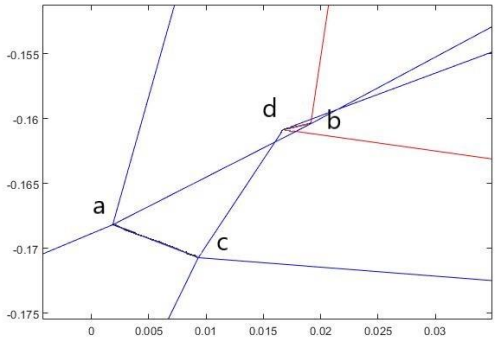
One of the essential steps in the combined scheme is to know where we should keep the result from hyperbolic grid generation and where we need to fix the grids by elliptic techniques, i.e., to detect the "bad" points that tangle with other grid points. First, a good grid cell should be a simple quadrilateral, in which the corners of the cell are convex corners and edges cannot intersect with each other. Any grid point that fails to pass this criterion will be viewed as bad points, and the elliptic technique will be applied in those areas. An example figure about gridlines twisting is shown in Figure 34 (a). In this picture, red lines are the hyperbolic grid lines calculated based on the grid lines on the bottom left. Line ab and cd are two grid lines along η direction. Points along line ab and line cd can be written as

$$\vec{p}_a = (1-s)\vec{r}_a + s\vec{r}_b \tag{4.1a}$$

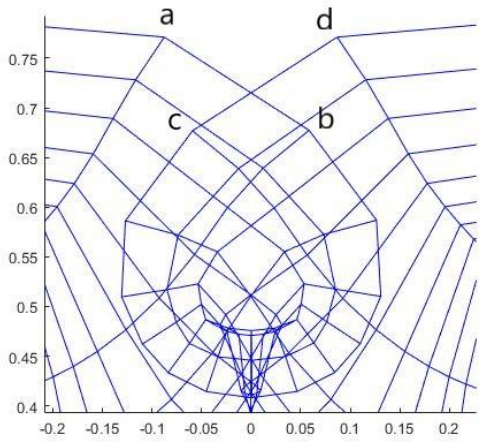
$$\vec{p}_c = (1-t)\vec{r}_c + t\vec{r}_d \tag{4.1b}$$

where s and t are the parameters along the grid lines. If we let $\vec{p}_a = \vec{p}_c$, we can have the intersection of the two grid lines, and if both s and t are in the range from 0 to 1, we can define these points as bad points. Similar equations can be applied along ξ direction.

Except for the situation mentioned above, we also need to see if grid lines twists due to the geometry of the initial curve, i.e., the grid lines self-intersect in a concave corner even though each grid cell looks good. Another example of self-intersection is presented in Figure 34 (b). In this case, we can use the same equations as above but apply them on every two grid lines along ξ direction in the same layer but not consecutive. Once we find an intersection point, we will treat all points from Point a to Point d as bad points.



(a) Gridlines twisting



(b) Self-intersection

Figure 34: Different kinds of bad points in hyperbolic grid generation

Equation 4.1 is relatively straightforward to find an intersection point on a plane. For the surface grids and the volume grids where the grid lines are typically not coplanar, we can use pseudoinverse or normal equations to solve the parameters in Equation 4.1, and it will give the closest points on the two grid lines instead of the intersection point. However, we should also check the distance of the two closest points (minimum distance of two grid lines) to see if the

lines really "intersect," i.e., the distance is smaller than a critical value. In this paper, the critical values for gridline twisting c_{gt} and self-intersection c_{si} are set as

$$c_{gt} = 0.5 * \min\left(\left|\overrightarrow{r_\xi}\right|, \left|\overrightarrow{r_\eta}\right|\right) \tag{4.2a}$$

$$c_{si} = 10 * \left|\overrightarrow{r_\xi}\right| \tag{4.2b}$$

where $\left|\overrightarrow{r_\xi}\right|$ and $\left|\overrightarrow{r_\eta}\right|$ are the length of the local grid line along ξ and η respectively. Sometimes two grid lines are quite far away from each other, and there is no chance to "intersect." Therefore, a precheck about the bounding boxes of the two grid lines can be applied before finding the intersection point using the following inequations

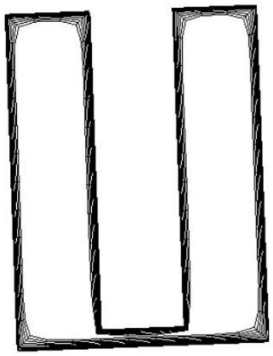
$$max\left(\vec{r}_{i,a}, \vec{r}_{i,\mathrm{b}}\right) < min\left(\vec{r}_{i,c}, \vec{r}_{i,d}\right) \tag{4.3a}$$

$$max\left(\vec{r}_{i,c}, \vec{r}_{i,d}\right) < min\left(\vec{r}_{i,a}, \vec{r}_{i,b}\right) \tag{4.3b}$$

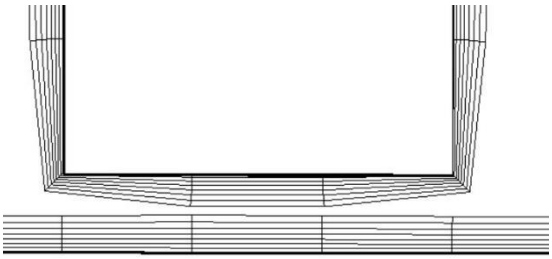
where $i = 1, 2, 3$ refers to the x, y, and z component of each point, respectively. This set of inequalities refers that if the coordinate component of all points on one grid line is smaller (or larger) than those on the other, there will not intersect.

The bad points detection discussed above is applied after each layer of hyperbolic grid generation, and a vector is used to save the position and layer number of the bad points. In order to avoid too much grid twisting when generating grids inside a closed loop, we set the maximum number of bad points in one layer as half of the total number of the grid points. If the number of bad points in one layer exceeds this maximum, we will discard the latest layer of grids and terminate hyperbolic grid generation. An example of this criterion is shown in Figure 35 when generating hyperbolic grids inward a U-shaped body with a thin neck. All points will be viewed

as bad based on the criteria above if we generate one more layer of hyperbolic grids. Thus, the grid generation is terminated, and the last layer is removed.



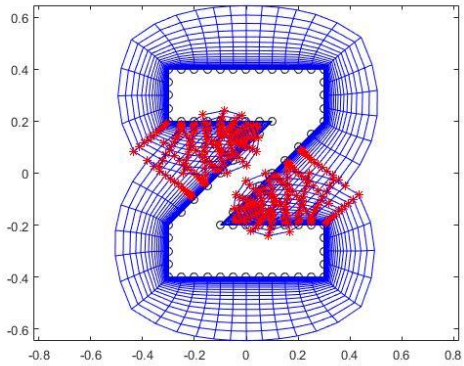
(a) Overview of the hyperbolic grids generating inward a U-shaped body



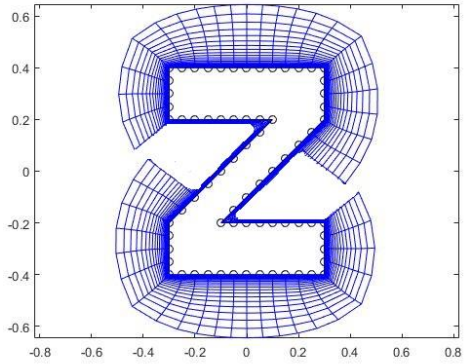
(b) Closed-up view at the neck of the body

Figure 35: Example for hyperbolic grids generating inward a U-shaped body with a thin neck

After finding the bad points using the criteria mentioned above, we can build bad points region(s), which include all bad points and are rectangular regions in computational space. Here is the same example in Figure 17. The bad points regions are detected and marked in red, shown in Figure 36 (a). We can just cut these points out and take elliptic calculation in the blank regions, shown in Figure 36 (b).



(a) Hyperbolic grids with bad points marked in red



(b) Hyperbolic grids with bad points removed

Figure 36: Example for hyperbolic grids with bad points

4.2 Outer Boundary Definition

Before calculating the elliptic grids, we need to define outer boundary points to enclose the patching area. One of the choices of defining outer boundary points is to make a quadratic Bezier curve. The two endpoints of the Bezier curve are directly derived from the hyperbolic grid result, and the control point is the intersection point of the grid lines on the last layer of hyperbolic grids, which get through the endpoints. Another example of a hyperbolic grid is

shown in Figures 37 (a) and (b), with all bad points removed. We can write an equation of the line P_1P_2 in a parametric form as

$$P_a = P_1 + s_a(P_2 - P_1) \tag{4.4a}$$

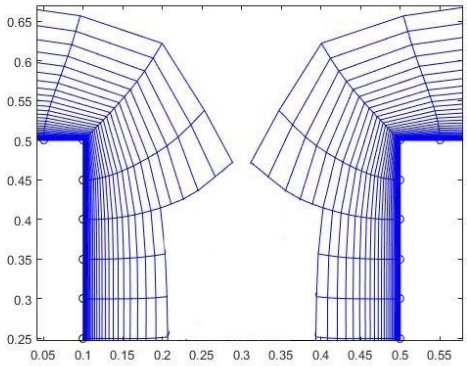
and the line P_3P_4

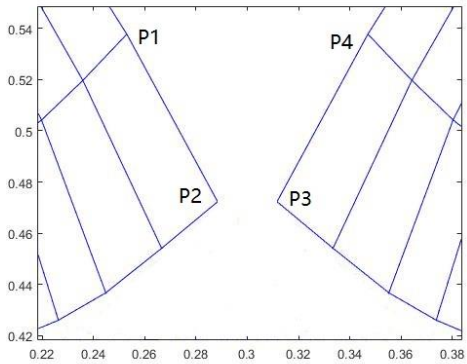
$$P_b = P_4 + s_b(P_3 - P_4) \tag{4.4b}$$

where s_a and s_b are the parameters along line P_1P_2 and line P_3P_4 separately. Then we can find the intersection point P_5 of the two lines by setting $P_a = P_b$ and draw a Bezier curve using points P_2, P_3, and P_5 as

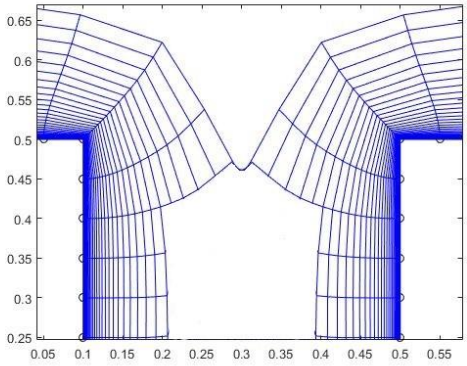
$$P = (1-t)^2P_2 + t(1-t)P_5 + t^2P_3, \quad 0 \le t \le 1 \tag{4.5}$$

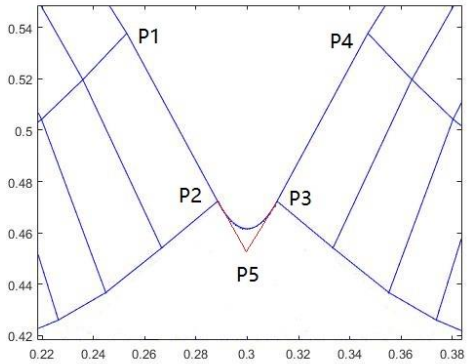
where t is the parameter along the curve, and it can be derived by using the same points distribution ratio on the initial body curve. Figure 35 (c) and (d) show the calculated outer boundary. After setting the outer boundary points, we can then calculate the grids in the patching area by the elliptic grid generation scheme.





(a) Hyperbolic grids with bad points removed (b) Closed-up view



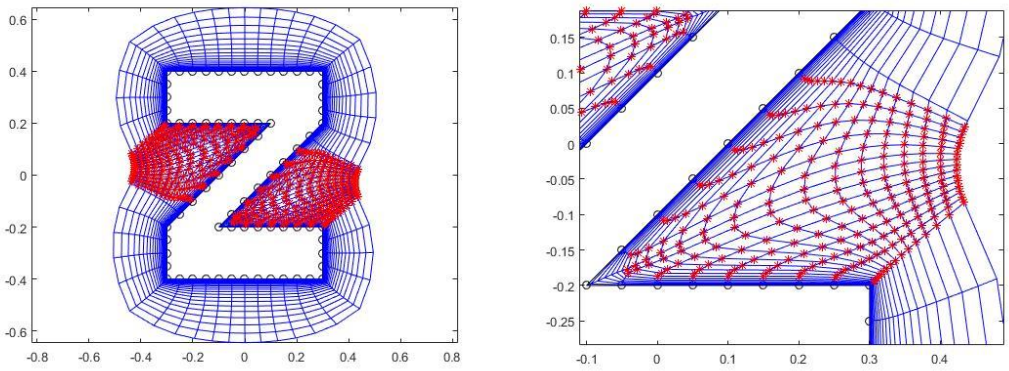


(c) Using Bezier curve as outer boundary (d) Closed-up view

Figure 37: Making outer boundary of elliptic grids using quadratic Bezier curve

The advantage of using the Bezier curve is that it can make a smooth outer boundary curve for the elliptic region where the tangents at each endpoint are the same for both hyperbolic and elliptic grids, i.e., C^1 continuous. However, there are several special situations that we need to take into account. First, the intersection point may lie on either side of the two grid lines, i.e., s_a or s_b is in the range between 0 and 1. In this case, we can manually enlarge the bad points region on the corresponding side and recalculate the intersection point until both s_a and s_b are greater than one. A good example of this case is the Z-shaped body (Figure 34). It can be seen

that the intersection point will lie on the left grid line if we use the method mentioned above (s_a is about 0.6 in this case). Thus, we need to cut out one column of grids (along η direction) on the left of bad points regions and recalculate the outer boundary. The outer boundary after recalculation and the elliptic grids are shown in Figure 37 (b). Besides, the intersection point may be too far away from the endpoints (the slope of the grid lines are closed) or even does not exist (the grid lines are parallel), or the bad points region is on the non-periodic boundary of the hyperbolic grids. In any of these cases, we will use a straight line as the outer boundary of the elliptic region. This will lead to C^0 continuous at the intersection of hyperbolic and elliptic grids, but it is still solvable.



(a) Far-field view (b) Closed-up view

Figure 38: Generate elliptic grids in bad points regions

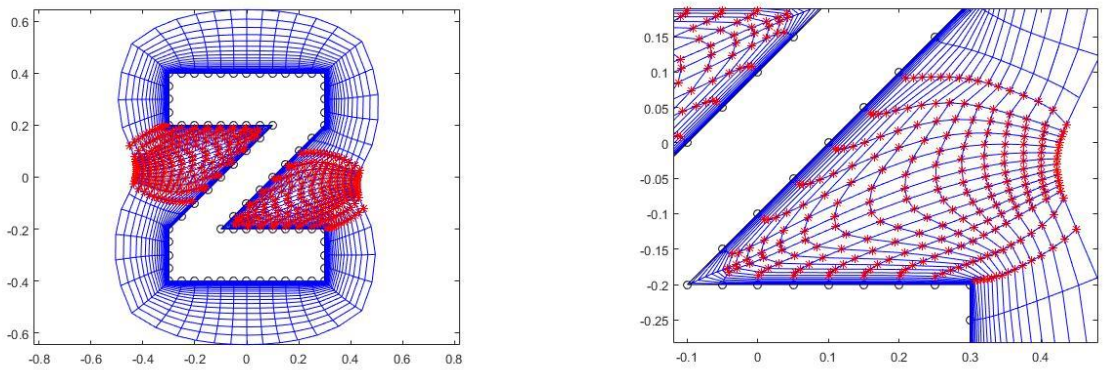
It should also be noticed that some modifications have been applied to the elliptic grid generation in the combined scheme. First, the source terms (P, Q, and R) on the outer boundary of the elliptic grid generation scheme may lead to the spacing decrease along the direction normal to the geometry body near the outer boundary, which is different from the hyperbolic grid generation results. Also, orthogonality is not highly required near the outer boundary of the

combined scheme. Therefore, only the source terms on the inner boundary are applied in the combined grid generation scheme. Second, the source terms in the original elliptic grid generation method are calculated based on spacing and orthogonality requirements on the boundaries as an approximation of the derivatives in the normal direction. However, in the combined scheme, we can get the derivatives in the normal direction more accurately by directly differencing the hyperbolic grids nearby.

4.3 Intersection Regions Correction

The last step in the combined scheme is to revise the intersection regions (the left and right boundary of the patching area) because there is a massive difference in cell sizing between hyperbolic grids and elliptic grids. To improve grid quality at that place, we first use neighbor points of the left and right boundary as the fixed region and recalculate the grid points by the elliptic method. For example, if the elliptic region locates in $a \leq j \leq b$, we will recalculate points on $j = a$ using grids $j = a - 1$ and $j = a + 1$, which are calculated in the previous step. Same procedure is applied on $j = b$. This intersection regions correction can be further applied on $j = a + 1$ and $j = b - 1$ to improve the grid quality if necessary. Since we only need to solve one column of points in each step, the computational speed will be much faster compared with the main elliptic grids calculation.

In our example of grids around Z-shaped body, from Figure 38 (a) and (b), it is clear to see that grid size calculated from elliptic grid generation is much smaller than the size of hyperbolic grid nearby. But if we recalculate the left and right boundaries of the patching area, as shown in Figure 39 (a) and (b), the problem can be reduced.



(a) After connecting curve correction (b) Closed-up view

Figure 39: Example for connecting curve correction

4.4 Bad points detector and outer boundary correction in elliptic volume grid generation

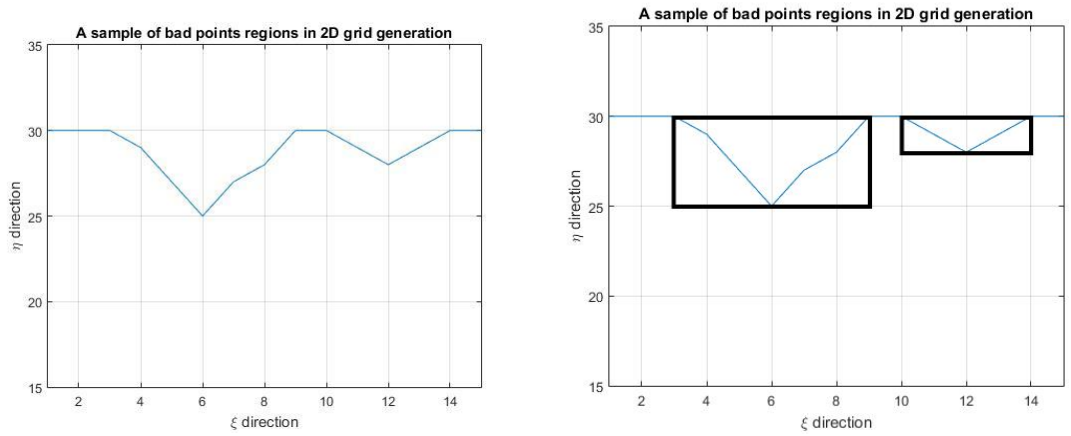
The basic idea for bad points criteria for three-dimensional grids is an extension of the criteria mentioned in Chapter 4.1, which is applied to on the slices of constant ξ face, constant η face, and constant ζ face. If a grid point satisfies Equation 4.1 on any slices of three-dimensional grids, it will be treated as a bad point. And the bad points will then group into bad point regions using the method, which will be discussed in Chapter 4.6. In practice, bad grids are typically generated due to the concave corner on the surface grid. Thus, if we combine two surface grids and then generate a volume grid and define the intersection curve of the two surfaces as η direction, it is most likely that bad points occur on the constant η faces.

The outer boundary definition of the three-dimensional elliptic region is also an extension of the two-dimensional cases. As a matter of fact, outer boundary points are calculated in the same way as two-dimensional cases on each constant η face. Transfinite interpolation using the good grids on the last layer of the hyperbolic volume grid is another optional choice of outer boundary definition. However, we found that the outer boundary may bend into the surface and

make negative regions in some cases using such a scheme, and it is thus not preferred. Once the outer boundary of the volume grids is defined, the initial guess of the entire grid can be retrieved by three-dimensional transfinite interpolation.

4.5 Bad Points Grouping

One big problem about bad point detection for volume grids is how to group the bad points. In order to apply the elliptic grid generation technique, the bad points region should be defined, which can enclose all the bad points and should be as small as possible to reduce the computational effort, which is also known as a bounding box. Sometimes the bad points will cluster in several regions, and then elliptic grid generation should be applied separately. The side boundaries of these regions can be easily found in two-dimensional grids by sweeping each ξ line. An example of this is shown in Figure 40(a).

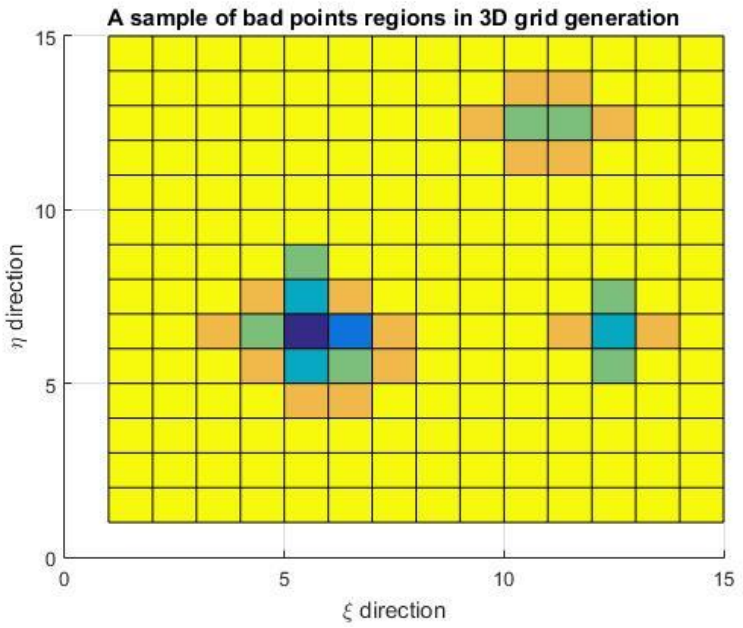


(a) Sketch map of good points in each ξ line (b) Result of bad points regions

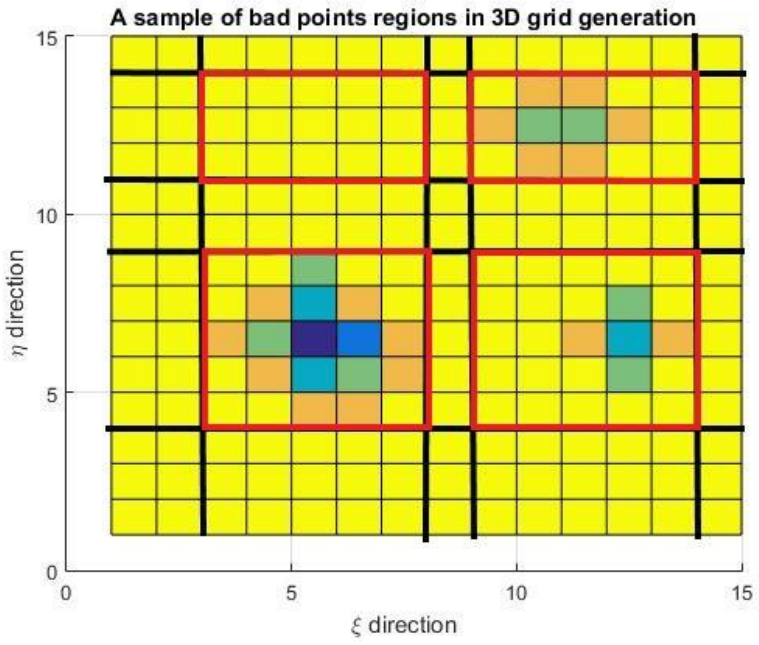
Figure 40: Bad points Regions Grouping in two-dimensional grids.

The curve in the picture refers to the number of good points in each ξ line in computational space, i.e., the bad points are above the curve (and below ξ_{max}). Then bad points regions can be easily found by sweeping along the ξ direction, shown in Figure 40 (b).

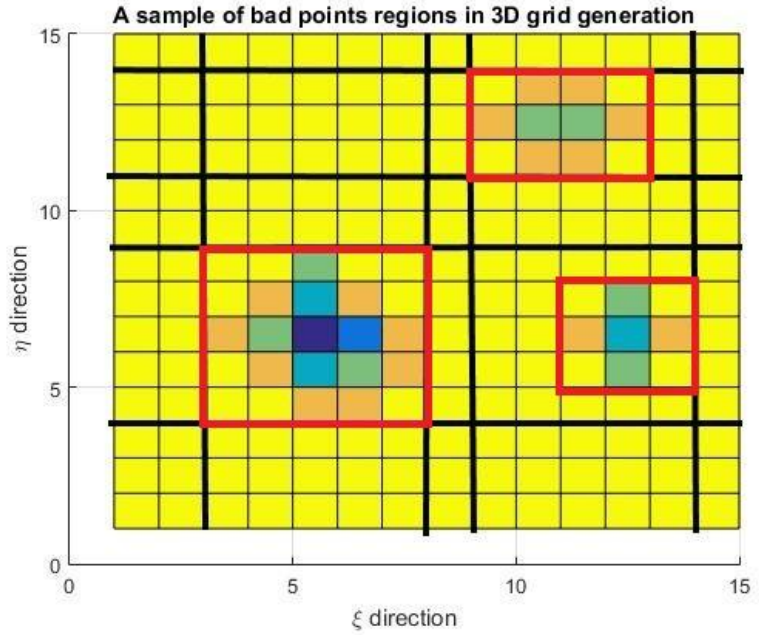
However, this method cannot be achieved easily for a volume grid because sweeping can only go in either ξ or η direction while the bad points regions require consecutive bad points in both directions. If we combine all the bad points information in the same constant ξ or η line, the information from different bad points regions might be blocked or overlapped with each other. An example sketch of bad points regions in computational space is shown in Figure 41 (a). In this picture, different colors refer to different good (or bad) points along ζ direction, which is supposed to be out of the plane. And the colors other than yellow refers to the existence of bad points along ζ direction. It will be hard to find decent bad points regions if we use the method mentioned above.



(a) Sketch map of bad points in computational space viewed in ζ direction



(b) First step of finding bad points regions



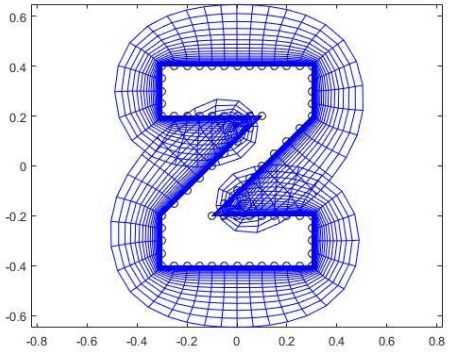
(c) Second step of finding bad points regions

Figure 41: Bad points Regions Grouping in three-dimensional grids.

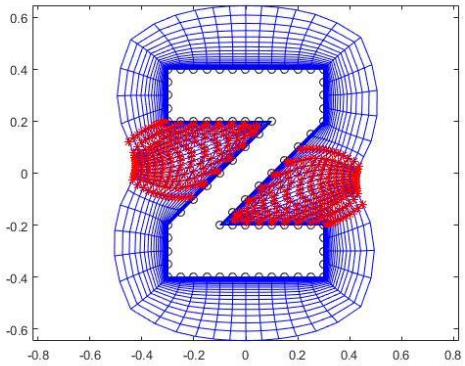
A two-step bad points grouping method is then introduced to find the boundary volume of bad points regions. In the first step, we still combine all the bad points information in the same constant ξ and η line, but we multiply the results together as the first guess of bad points regions. In this case, for example, $2 \times 2 = 4$ possible bad points regions are found, which is shown in Figure 41(b). In the second step, we check every possible bad points region. If no bad points exist in the region, the elliptic revise is skipped, and the number of bad points regions will be reduced. If bad points are found in the region, the bounding box of the bad points region can be further refined by the minimum and maximum ξ and η value of the bad points in this region. The final result about bad points regions of the example case is shown in Figure 41 (c) and marked in red, in which all bad points are found and grouped into minimum bad point regions.

4.6 Sample Results

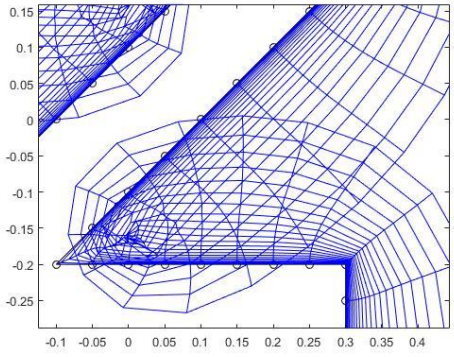
Grids in two dimensions and three dimensions using the combined grid generation scheme are presented in this section. By comparing the result from the hyperbolic scheme with that from the combined scheme, we can find that the combined scheme solves the problems about grid overlapping and grid clustering successfully, while the hyperbolic scheme may have such problems.



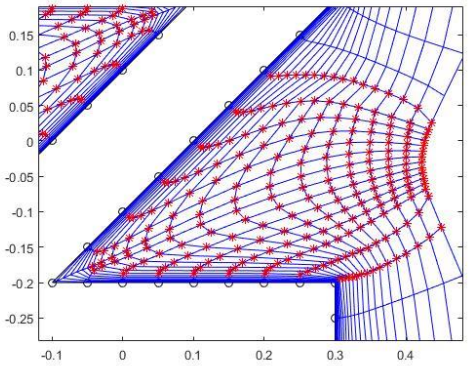
(a) Hyperbolic grid result



(b) Combined scheme result

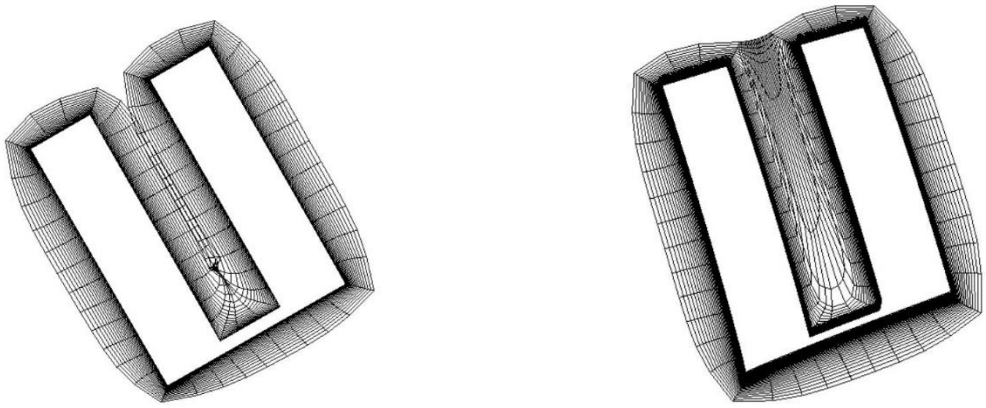


(c) Close-up view of hyperbolic grid result



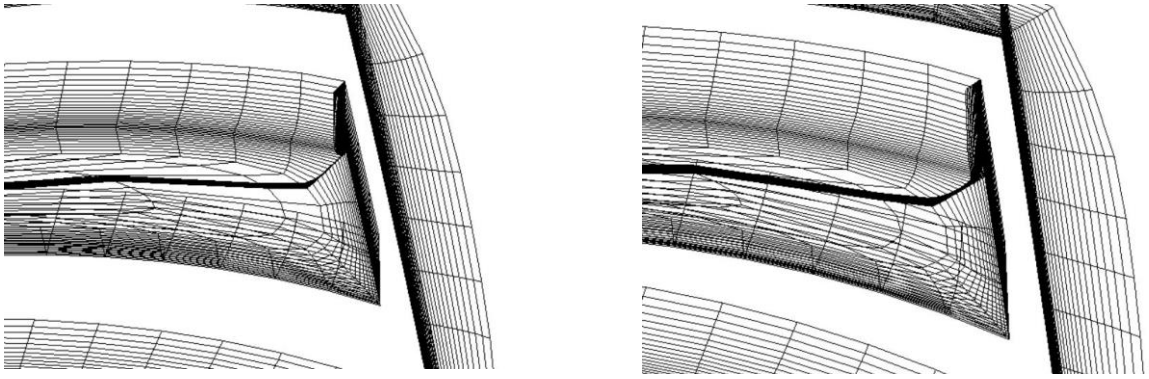
(d) Close-up view of the combined scheme result

Figure 42: Grids around a Z-shaped body.



(a) Hyperbolic grid result (b) Combined scheme result

Figure 43: Surface grids around a U-shaped body.



(a) Hyperbolic grid result (b) Combined scheme result

(c) Close-up view of hyperbolic grid result (d) Close-up view of the combined scheme result

Figure 44: Slices of volume grids around a U-shaped body.

4.7 Computational speed

The computational speed of these examples by using hyperbolic grid generation, elliptic grid generation, and the combined scheme is shown in Table 6. It should be mentioned that the results marked with asterisks (hyperbolic grid generation results for cases 2, 4, and 6) are not desirable because the grids will cluster or overlap in some places.

From the result shown below, it can be seen that the combined scheme takes only a little bit more time than hyperbolic grid generation, while the elliptic grid generation method takes significant time. This is because in the combined scheme, most grid points are generated by hyperbolic grid generation, and only a few grid points need to be revised by the elliptic method. Case 1, 3, and 5 show that all grid points from hyperbolic grid generation are acceptable, and no grids need to be revised. Then the combined scheme will use the result from hyperbolic grid generation, but it still needs some extra time for bad points detection.

Table 6: Comparison of computational time between hyperbolic grid generation, elliptic grid generation, and combined scheme

Case number	Name	Grid Type	Grid size	Hyperbolic grid generation (s)	Elliptic grid generation (s)	Combined scheme (s)	Ratio between hyperbolic and combined scheme	Ratio between elliptic and combined scheme
1	Y-shaped body	Planar grid	46×48	0.007*	1.369	0.108	0.064	12.676
2	Z-shaped body	Planar grid	23×36	0.003*	1.937	0.039	0.077*	49.667
3	Curve	Surface grid	21×62	0.007	0.067	0.009	0.778	7.444
4	U-shaped body	Surface grid	21×66	0.031*	1.316	0.358	0.087*	3.676
5	Curve	Volume grid	41×62×21	0.243	2.500	0.358	0.679	6.983
6	U-shaped body	Volume grid	41×66×21	0.264*	26.872	1.586	0.166*	16.943

Note: Computational time may be different on different software and operating system. The first two cases are compiled and run in Clion software from JetBrains, and the last four cases are compiled are run in x64 Native Tools Command Prompt for VS 2019 from Visual Studio on a laptop with Windows 10 operating system, Intel(R) Core i7-7700HQ CPU 2.80 GHz processor, and 16GB RAM.

5. Application of the Combined Scheme in Engineering Sketch Pad (ESP)

Engineering Sketch Pad (ESP) is a browser-based system that allows users to interact with a configuration by building or modifying the design parameters and feature tree that define the configuration. It is explicitly designed to support the analysis and design of aerospace vehicles. ESP is built both upon the WebViewer (which is a WebGL-based visualizer for three-dimensional configurations and data) and upon OpenCSM (which is a constructive solid modeler; it, in turn, is built upon the EGADS and OpenCASCADE systems) [12]. With the help of these open-source software components and some of the distinguishing features of ESP, the combined grid generation scheme can be applied and examined in ESP.

5.1 Edge and Points Information for Grid Generation

The first step for automatic near-body grid generation starting from a given geometry is to find the intersection curves between different parts and decide the distribution of points along the curves. In order to find these intersection curves, attributes in ESP are applied such that all faces in the same part, like fuselage, wing, flap, slat, and symmetry plane, are grouped in one attribute. By taking these attributes, surface grids will be generated only on the intersection curves of different parts. Besides, the surface grids can be generated on different faces as long as they are in the same group of attributes. This feature makes it feasible to generate surface grids on a complicated geometry.

After finding the intersection curve, the next step is to rearrange the edges along the curve. Function EG_makeLoop in Engineering Sketch Pad can be applied to adjust the sequence of the edges, but the edges collected by this function may have opposite directions in one loop. For a two-dimensional grid, the grid should be on the left-hand side of the edges, or the edges should be clockwise to generate a grid outside of the loop. Therefore, any edges which are not clockwise should be flipped, which means points distribution on that edge is computed and saved back to front. Besides, sometimes even the sequence of the edges is not clockwise and should be flipped as well.

Once the intersection curve and the sequence of edges are decided, a proper points distribution is required to start the grid generation. Equal spacing is generally feasible for a curve with no prescription. However, sometimes we want the grids clustered at some points or edges, like the leading edge and trailing edge of the wing or the nose and tail of the fuselage. Then attributes can also be applied on corresponding Nodes or Edges to mark these distinguishing features. Points spacing along the intersection curve will be set smaller where these attributes are found and will gradually increase on both sides with a given stretching ratio until a standard size, usually the value of equal spacing, is reached. Once all the points on the intersection curve are computed, the surface and field grid can then be calculated using the combined grid generation scheme.

5.2 Locating and Projecting Grid Points using Tessellation

When taking the projection step in generating a hyperbolic surface grid, except for using the Jacobian matrix mentioned in Chapter 2.3, the EG_invEvaluate function in ESP is also a helpful tool to find the point that is on the reference surface and closest to the calculated point.

However, this function is inefficient because it takes iteration to find the point. Besides, if the hyperbolic surface grid develops into the other surfaces, the EG_invEvaluate function may need to run several times on all the neighbor surfaces to find the closest surface as well as the closest point. This is also a problem if we want to use the Jacobian matrix because the parameter coordinates are usually different on different surfaces. Plus, it would be unstable to use EG_invEvaluate if the point is out of the domain of the surface.

To better find the closest points and surfaces for hyperbolic surface grid generation on multiple surfaces, tessellation in Engineering Sketch Pad is utilized to help to locate the points. Tessellation is a group of unstructured grids on each face. A tessellation contains not only the coordinates of the node points and supporting triangle indices but also other data, such as the underlying surface parameters for each point and the connectivity of the triangles, assists in traversing through dissecting a complex part. However, these kinds of information are counted separately on each surface. In order to better utilize the tessellation, the information of nodes, edges, and triangles in each face are connected such that nodes and edges can be mapped through different faces.

Once the mapping among all faces is generated, an extended barycentric coordinate system can be applied to locate the grid points. A barycentric coordinate system is usually used for points in a two-dimensional plane. When it turns to a surface, the equations of the barycentric coordinate system are extended as

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \tag{5.1a}$$

$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 \tag{5.1b}$$

$$z = \lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3 \tag{5.1c}$$

where λ_1, λ_2, λ_3 are the barycentric coordinates of the point $\vec{r} = (x, y, z)$, $\vec{r_i} = (x_i, y_i, z_i)$ $i = 1,2,3$ are the triangle vertices, and $\lambda_3 = 1 - \lambda_1 - \lambda_2$ since the sum of the barycentric coordinates should always be unity. Taking λ_3 into Equation 5.1, the system of equations becomes

$$\lambda_1(x_1 - x_3) + \lambda_2(x_2 - x_3) + x_3 - x = 0 \qquad (5.2a)$$

$$\lambda_1(y_1 - y_3) + \lambda_2(y_2 - y_3) + y_3 - y = 0 \qquad (5.2b)$$

$$\lambda_1(z_1 - z_3) + \lambda_2(z_2 - z_3) + z_3 - z = 0 \qquad (5.2c)$$

λ_1, λ_2 (and λ_3) can be solved if the Cartesian coordinates of all three nodes in a triangle and the Cartesian coordinate of a given point $\vec{r} = (x, y, z)$ are given. However, it can be easily found that there are two unknowns, but there are three equations, which means these variables are overdetermined. Therefore, pseudoinverse or normal matrix is needed to solve for λ_1, λ_2 (and λ_3). As a matter of fact, this set of barycentric coordinates gives a point in the triangle surface and is closest to the given point, which is also the projection of the given point to the triangle surface. And if the three barycentric coordinates λ_1, λ_2 and λ_3 are all between 0 and 1, the projection point will be in the triangle, and the corresponding surface is the actual projection surface.

In order to find the actual projection surface, a loop search for all the triangles is generally required. However, a better searching method can be applied if a known point and its triangle are given. Taking the Cartesian coordinate of the new point and the given triangle nodes into Equation 5.2, and if λ_1, λ_2 and λ_3 are all between 0 and 1, the new point is in the given triangle. If one of these values is negative, another triangle that shares the edge corresponding to this negative barycentric coordinate with the previous triangle should be checked. If there are

two negative barycentric coordinates, we can pick either of them and take the same steps. If negative values still exist, go to the next neighbor triangle until all the three barycentric coordinates are positive.

Several points should be pointed out when using the extended barycentric coordinate system. First, the uniqueness and existence of the result are not guaranteed in the extended scheme. There is a unique mapping between a given point and a projection point for a two-dimensional planar barycentric coordinate system. However, in the extended barycentric coordinate system, a space point may project onto more than one triangle at a concave corner or not project onto any of the triangles at a convex corner. In these cases, the projection point is on the triangle closest to a previous point (also the first searched) for a concave corner and on the intersection edge for a convex corner.

Another problem with this procedure is that this projection method may have more than one triangle that satisfies the criteria of the barycentric coordinates. Just take a point on a cylinder surface as an example. Two triangles can be found with all three coordinates lying between 0 and 1: one near this point and one on the opposite side of the cylinder. Therefore, the distance between the given point and the projection triangle should be checked, and the triangle with the smallest distance should be picked.

Besides, the size of the tessellation may affect the result of the projection. A tessellation with a smaller size can better represent a curved surface, and thereby the grid can be computed more accurately. But if the size of the tessellation is not tiny enough, bad grids could be generated at some places. For a surface grid generation, the initial points are evaluated directly on the intersection curve, and they are usually not on the edges of the tessellation. This will become a problem when generating grids on one side while some points are on the other side. An

example is shown in Figure 45. In this picture, green lines refer to the edge of tessellation while red lines refer to the initial grid curve for grid generation with two grid points staying on the left Face. The problem will happen when generating grids on the right Face because no triangle can be found with all three barycentric coordinates λ_1, λ_2, and λ_3 between 0 and 1 for these two points and the first layer of grids. This might lead to an incorrect evaluation of the position of the points and their normal vectors and then make a bad grid. This problem can be solved by reducing the size of the tessellation.

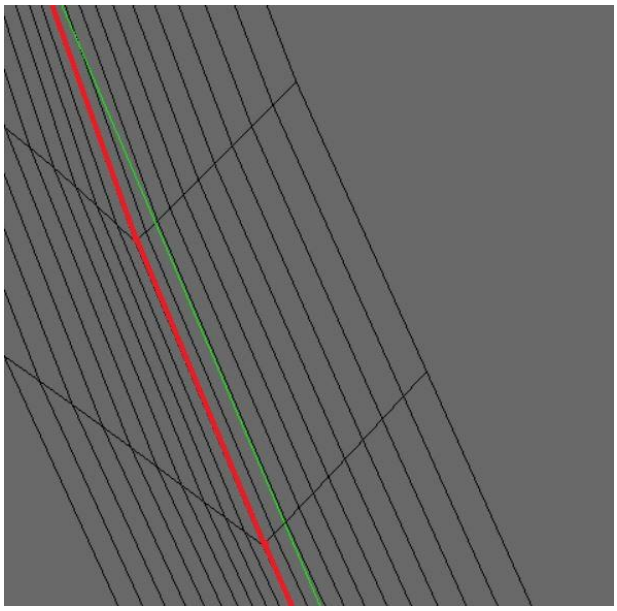
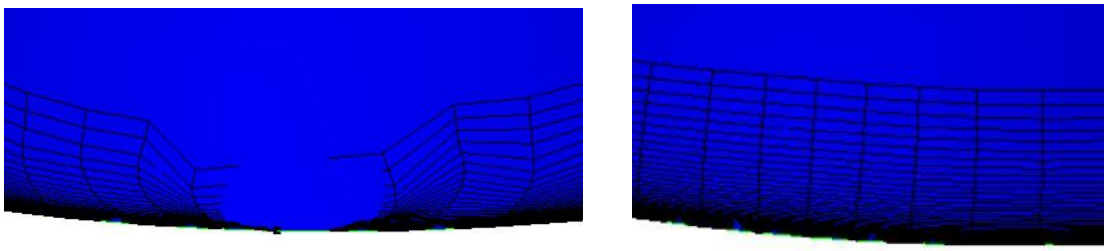


Figure 45: Initial grid curve (red) and the edge of tessellation (green).

Sometimes the tessellation cannot represent the geometry of a curved surface exactly, which will also lead to a bad grid generation. One particular case about this exists at the nose of the fuselage, which is shown in Figure 46 (a). In this case, the grid points fail to project onto proper tessellation and make a terrible grid. This failure is because u and v coordinates work like polar coordinates and the nose point is a singular point. The v coordinate could be any value from 0 to π at the nose point but only one value is saved in the tessellation and miscalculated in

the barycentric coordinate system. To solve this problem, we can search this kind of singular point by checking if the intersection angle between ξ-lines and η-lines is less than $\frac{\pi}{4}$ or larger than $\frac{3\pi}{4}$, or if the grid point in the new layer is too close to that in the previous layer. Once such singular points are found, we can recalculate the new projection point by inverse evaluation without any guess from tessellation. This check is only applied when calculating the first layer of grids, but the points in the rest layers of grids should take inverse evaluation without any guess to avoid further miscalculation if they are developed from the singular points. The revised result of the case at the nose of the fuselage is shown in Figure 46 (b).



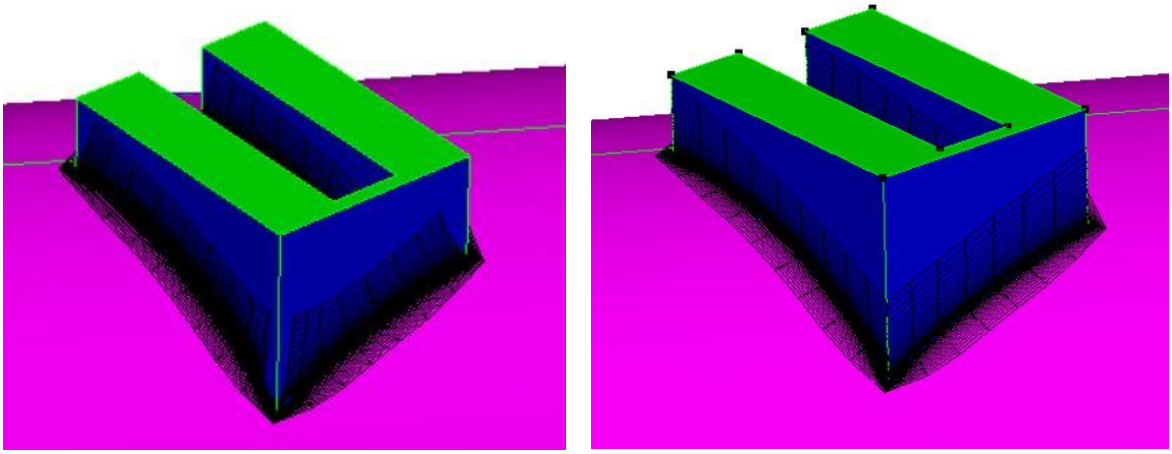
(a) Use tessellation only (b) Use both tessellation and inverse evaluation

Figure 46: Grids at the nose of the fuselage.

5.3 Separated grids

Smoothing is essential for hyperbolic grid generation, especially for the grids near a corner. However, smoothing will also cause problems if the marching direction (η direction) of the hyperbolic surface grids is along the corner edge. An example is shown in Figure 47 (a). Grids are generated at the intersection of blue and pink faces in this case. For the grids on the blue faces, grid points at the corner will be smoothed and projected onto a face on either side

rather than the corner edge. This will make the grids go through the body and lose the geometry information at this corner. In order to avoid this problem, the surface grids can be calculated separately on each side of the corner. And the corner edges become boundary conditions of surface grids. This forces the grid points to stay on the edges and keeps the shape of the corner. The criterion of such a corner is that the normal vectors of the same point on two faces are significantly different (with an angle of over 30 degrees). The result after using separate grids is shown in Figure 47 (b).



(a) One entire grid (b) Separated grids

Figure 47: Grids around a corner edge.

5.4 Imaginary grid points

Generally, the surface grids are generated in the same group of faces. When some grid points reach another group of faces, grid generation should be stopped at these points. However, surface grids with the same number of grid points in marching direction for all points are required in order to calculate a three-dimensional grid. We can terminate grid generation when

any point reaches another group of faces, but we can also keep generating grids using imaginary points. Imaginary points are the grid results before projecting onto reference surfaces. These results will be good enough to generate a smooth surface grid when the curvature of the surface is small. For each layer of grid points, differencing and smoothing terms for imaginary points are calculated in the same way as that for regular grid points. The only problem is that the normal vector cannot be evaluated directly on the surfaces. Then an equation for approximately calculating normal vector is derived as

$$\vec{n}_{j,k} = \begin{cases} \vec{n}_{l,k} + \left(\vec{n}_{j,k-1} - \vec{n}_{l,k-1}\right)\dfrac{\vec{n}_{r,k} - \vec{n}_{l,k}}{\vec{n}_{r,k-1} - \vec{n}_{l,k-1}}, & \vec{n}_{r,k-1} - \vec{n}_{l,k-1} \neq 0 \\ \vec{n}_{l,k} + \left(\vec{n}_{j,k-1} - \vec{n}_{l,k-1}\right), & \vec{n}_{r,k-1} - \vec{n}_{l,k-1} = 0 \end{cases} \tag{5.3}$$

where l and r refer to the first points on the left and right sides that are still in the given group of faces (regular points), and $(k-1)$ refer to the points in the previous layer.

Imaginary grid points can be calculated using the approximate normal vector, and the grid result is quite reasonable in a few layers of grids. An example is shown in Figure 48, with points in the grey region being imaginary grids. However, it should be mentioned that the tessellation near the intersection should be fine enough to avoid grid points projecting onto the wrong face and to separate regular points and imaginary points clearly.

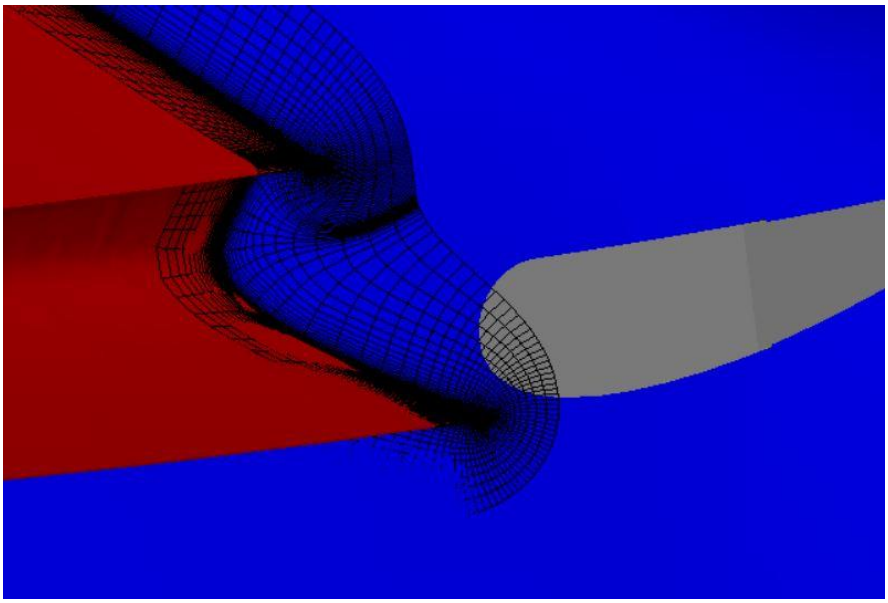


Figure 48: Imaginary grid points (grey region).

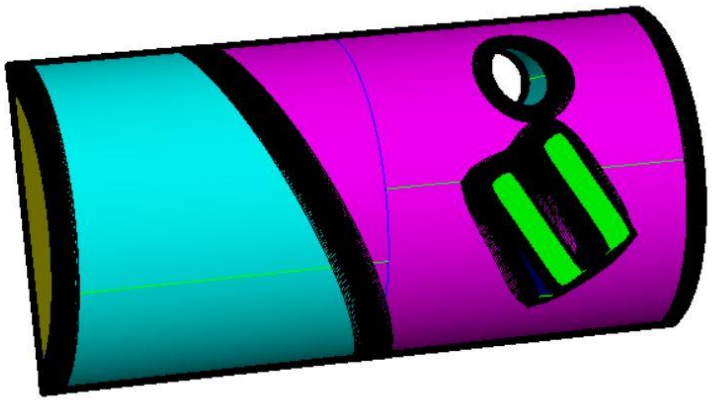
5.5 Sample Results of Grid Generation using Engineering Sketch Pad

Several examples of grids using combined scheme are presented in this section. In order to generate the grids, users need to provide an input file which includes the geometry of the configuration, the group number of each surface, along with the essential requirements of the surface and volume grids, i.e., number of points along the intersection curve, number of layers, initial spacing, and stretching ratio of surface and volume grids. Users will also need to run serveCSM in Engineering Sketch Pad to obtain the grids.

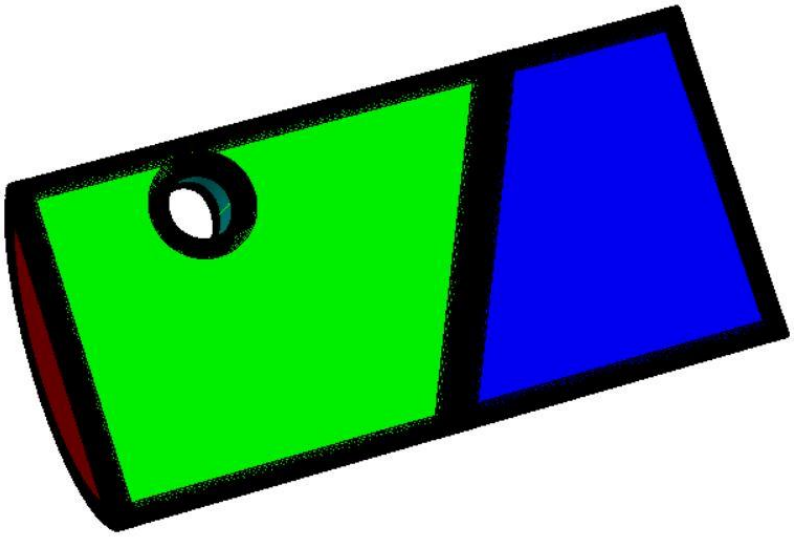
In these examples, different parts in this model are grouped and shown in different colors. Surface grids are generated at the intersection of different groups of faces on both sides. Then the surface grids are connected, and volume grids are calculated based on those surface grids. Sample cases of generating surface volume grids using Engineering Sketch Pad are shown below.

5.5.1 Semicylinder case

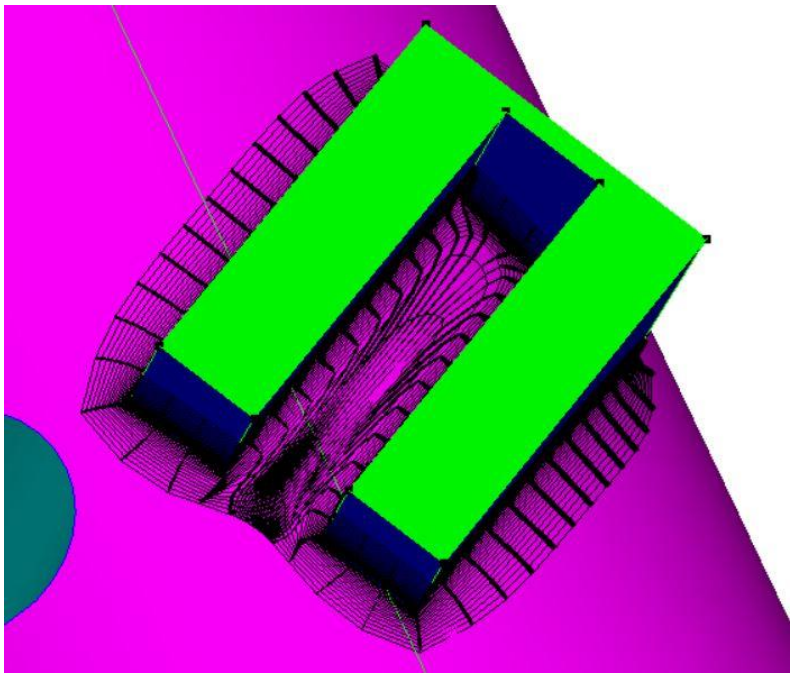
The geometry of the first case is a semicylinder that contains a U-shaped body and a cylindrical hole on its curved surface. This case is made up by using Engineering Sketch Pad to test the features mentioned in previous sections. Several views of this case are shown in Figure 49. Other views, including hyperbolic and combined scheme grids, are also shown in Figure 43 and 44. Different groups of faces are shown in different colors from their neighbor groups. The U-shaped body contains concave corners for both surface grids and volume grids, and those are the places where the combined scheme applies (Figure 49c). The vertical faces of the U-shaped body are not smoothly connected, so the grids should be generated separately to avoid grids getting inside the body (Figure 49e). The inclined curve on the semicylinder surface crosses several faces, and the tessellation feature can be tested (Figure 49f). The computational time of generating surface and volume grids using the combined scheme is shown in Table 7 and Table 8. The grid around the U-shaped body is the 8^{th} grid.



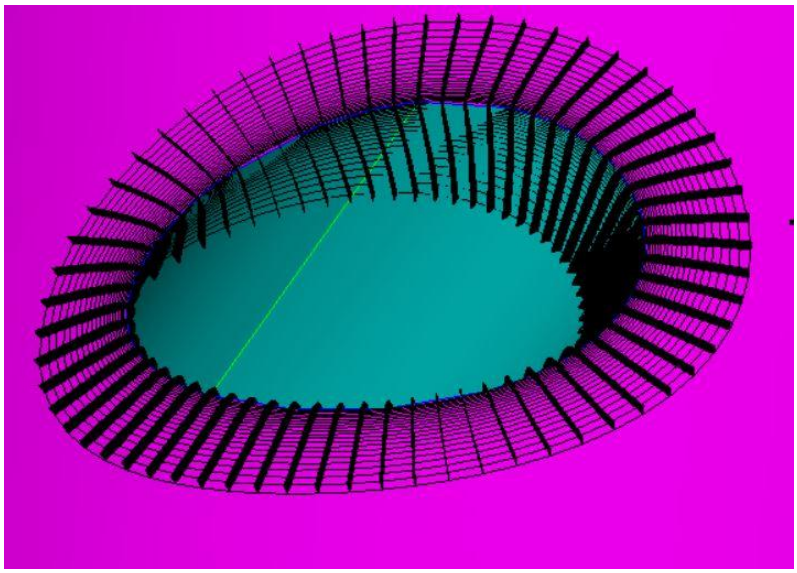
(a) Far field front view



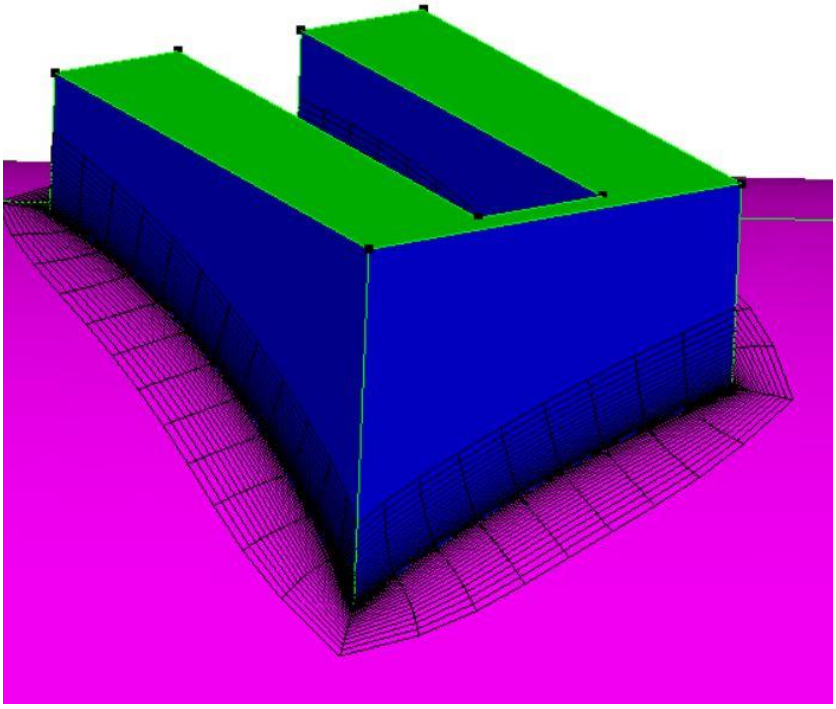
(b) Far field back view



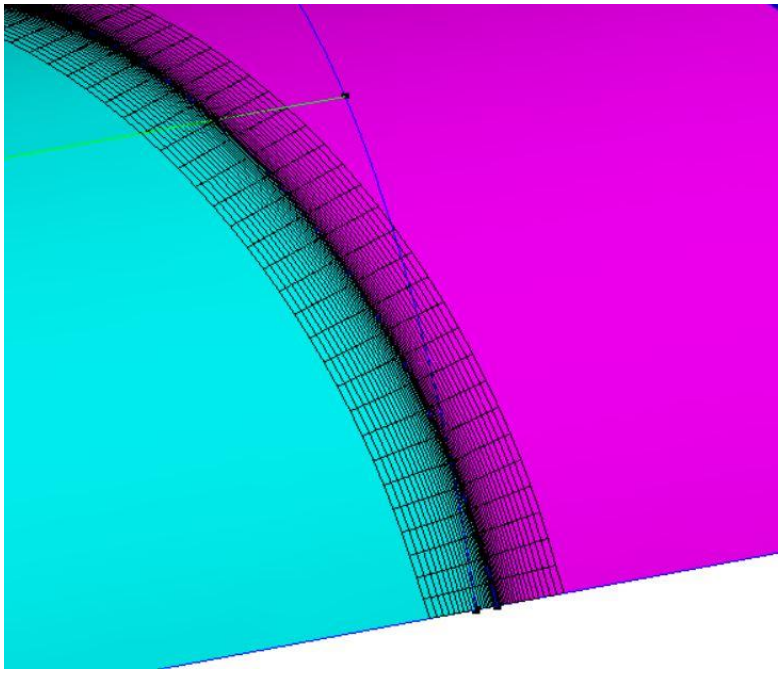
(c) Grids around U-shaped body



(d) Grids at the cylindrical hole



(e) Surface grids on the vertical faces generated separately



(f) Surface grids cross different faces in the same group

Figure 49: Surface and slices of volume grids of semicylinder case.

Table 7: Computational time of generating surface grids for semicylinder case

Grid number	Grid Size	Hyperbolic grid generation (s)	Elliptic revise (s)	Total time (s)
1	41×62	0.023	0	0.023
2	41×61	0.034	0	0.034
3	41×62	0.033	0	0.033
4	41×62	0.035	0	0.035
5	41×62	0.068	0	0.068
6	41×62	0.054	0	0.054
7	41×62	0.064	0	0.064
8	41×66	0.050	0.327	0.387
9	41×62	0.049	0	0.049
10	41×62	0.048	0	0.048
11	41×62	0.056	0	0.056
12	41×61	0.061	0	0.061
13	41×63	0.045	0	0.045
14	28×65	0.045	0	0.045
total				1.002

Table 8: Computational time of generating volume grids for semicylinder case

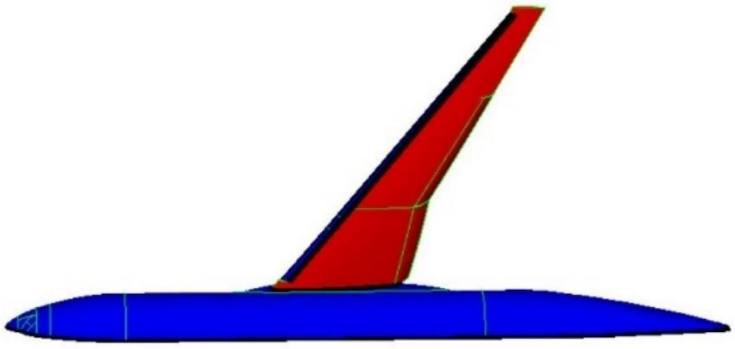
Grid number	Grid Size	Hyperbolic grid generation (s)	Elliptic revise (s)	Total time (s)
1	41×62×21	0.358	0	0.358
2	41×61×21	0.353	0	0.353
3	41×62×21	0.359	0	0.359
4	41×62×21	0.356	0	0.356
5	41×62×21	0.351	0	0.351
6	41×62×21	0.361	0	0.361
7	41×62×21	0.355	0	0.355
8	41 ×66×21	0.387	1.199	1.586
9	41×62×21	0.362	0	0.362
10	41×62×21	0.354	0	0.354
11	41×62×21	0.347	0	0.347
12	41×61×21	0.349	0	0.349
13	41×63×21	0.371	0	0.371
14	28×65×21	0.254	0	0.254
total				7.533

5.5.2 Semi-airplane case

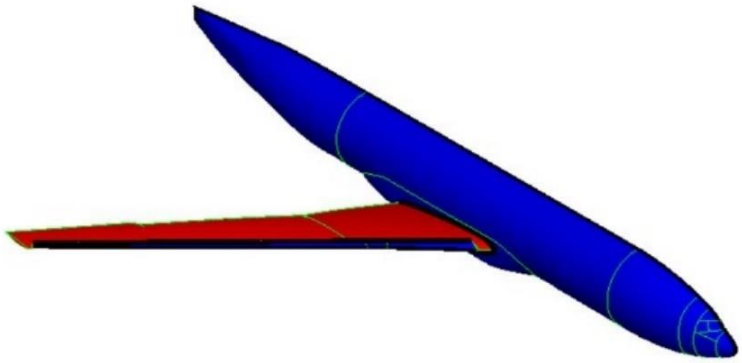
The second case is derived from the High Lift Common Research Model (HL-CRM) provided in the 2nd AIAA Geometry and Mesh Generation Workshop. Surfaces of the geometry

are grouped based on different airplane sections, including symmetry plane, fuselage, wing, slat face, and the pylon of the flap.

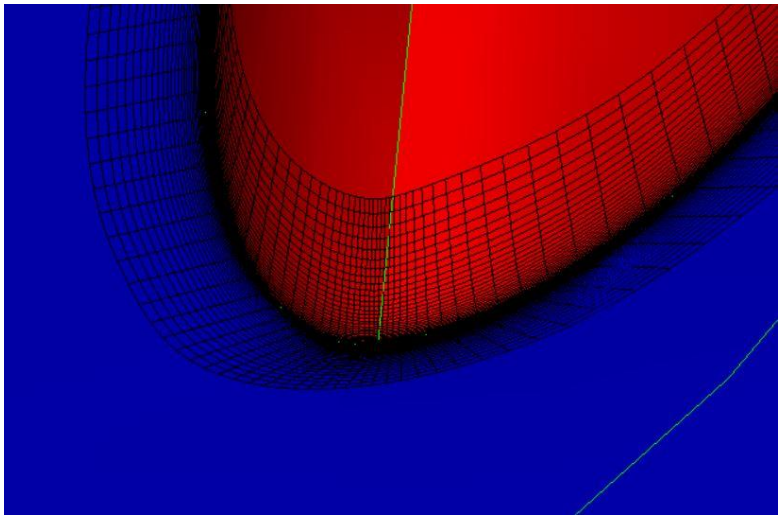
Overviews of surface and slices of volume grids are shown in Figure 50 a and b. The main problem of this case is that the hyperbolic volume grid will overlap near the trailing edge of the wing-fuselage intersection where the surface grid is convex in one direction but concave in the other direction (Figure 50 d). Elliptic grids are successfully applied in this region, and no twisting is found after the modification (Figure 50 e). Another problem about grid generation exists when generating grids on the wing, shown in Figure 50 g, that a highly sharp corner exists in the green region, and it is not desirable to generate surface and volume grids on that region. Besides, no edges in this figure are suitable as a side boundary when generating surface grids on the red face. Thus, an imaginary side boundary, which is normal to the red-blue intersection curve, is made up. Similar situations are also applied near the root of the wing. Other features, including the nose of the fuselage and imaginary grids, are also shown in Figure 46 and Figure 48. The computational time of generating surface and volume grids using the combined scheme is shown in Table 9 and Table 10, where grids at the wing-fuselage intersection are the 1st grids.



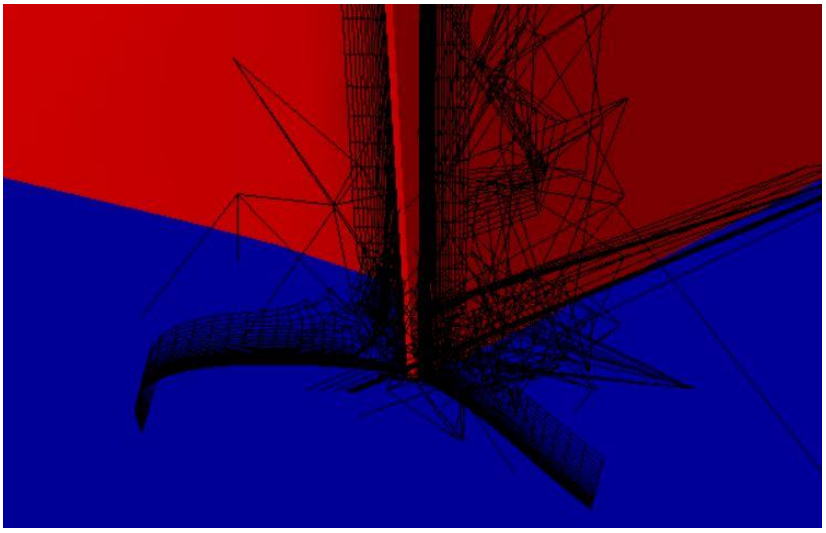
(a) Far field front view



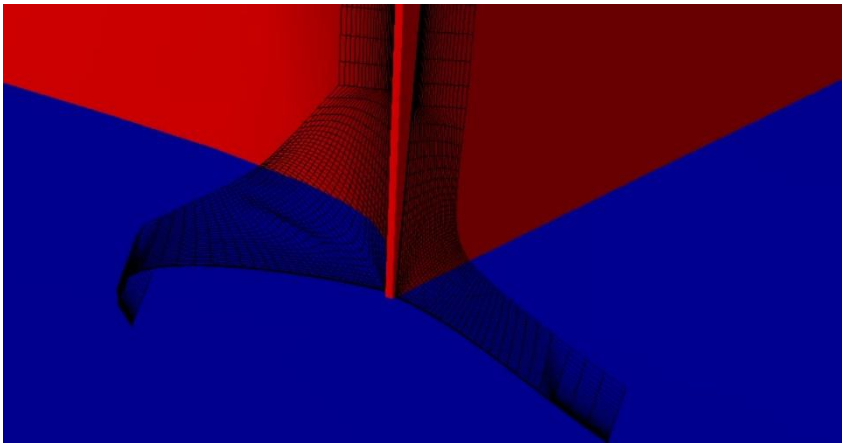
(b) Far-field side view



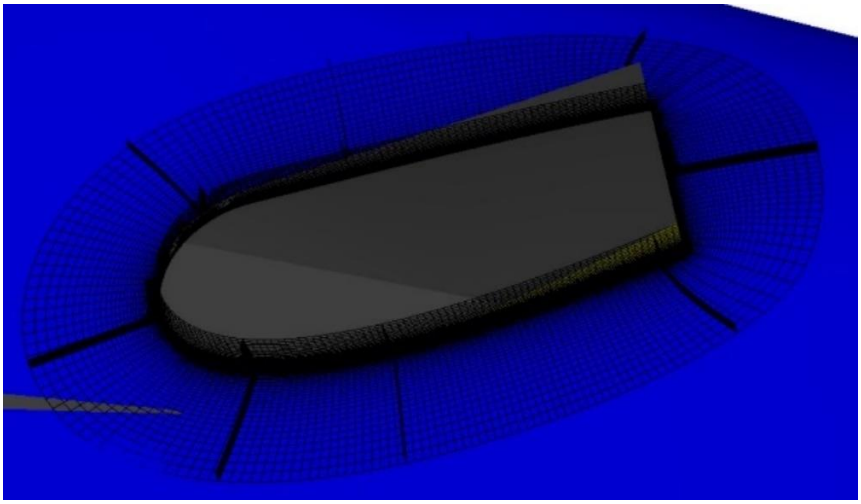
(c) Grids at the leading edge of the wing-fuselage intersection



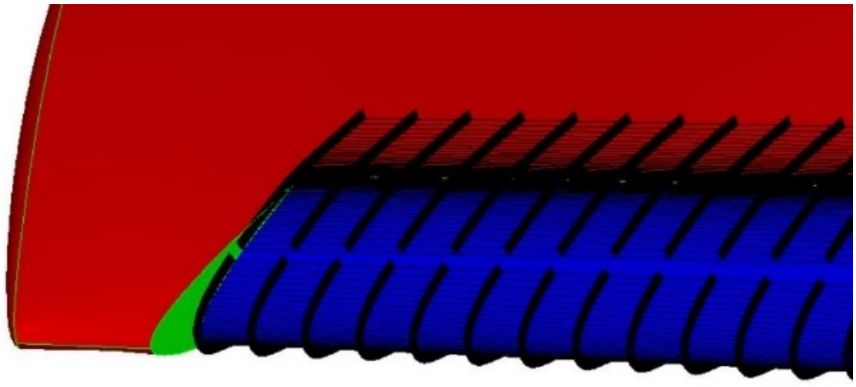
(d) Hyperbolic grids at trailing edge of wing-fuselage intersection (blow up)



(e) Combined scheme grids at trailing edge of the wing-fuselage intersection



(f) Grids at the pylon-fuselage intersection



(g) Grids of wing-slat face intersection near the tip of the wing

Figure 50: Surface grids and slices of volume grids of semi-airplane.

Table 9: Computational time of generating surface grids for semi-airplane case

Grid number	Grid Size	Hyperbolic grid generation (s)	Elliptic revise (s)	Total time (s)
1	61×339	1.704	0	1.704
2	52×203	0.632	0	0.632
3	61×244	2.632	0	2.632
4	52×215	0.704	0	0.704
5	61×204	0.804	0	0.804
6	61×205	0.808	0	0.808
total				7.284

Table 10: Computational time of generating volume grids for semicylinder case

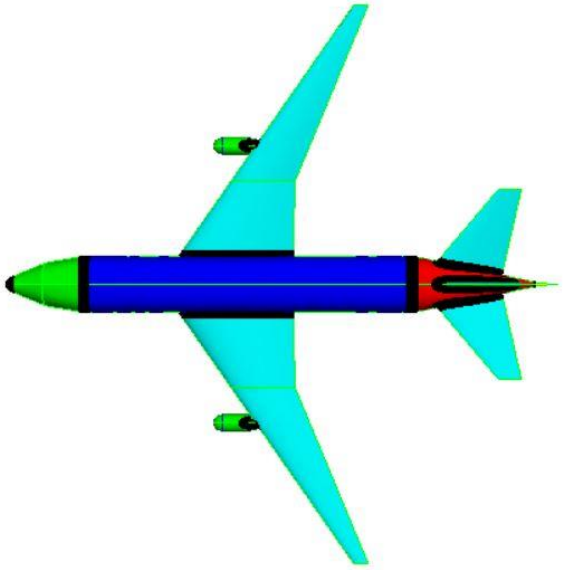
Grid number	Grid Size	Hyperbolic grid generation (s)	Elliptic revise (s)	Total time (s)
1	61×339×31	5.017	85.777	90.794
2	52×203×31	2.484	0	2.484
3	61×244×31	3.701	0	3.701
4	52×215×31	2.655	0	2.655
5	61×204×31	2.934	0	2.934
6	61×205×31	2.975	0	2.975
total				105.543

5.5.3 Open Parametric Aircraft Model case

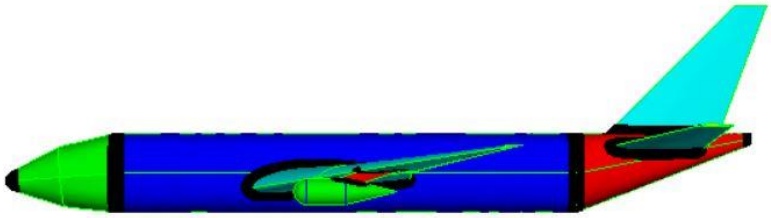
The third case is derived from the Open Parametric Aircraft Model #1 (OPAM-1) model that is also provided in the 2nd AIAA Geometry and Mesh Generation Workshop. This aircraft model is defined by 53 design parameters, including wing sweep, fuselage length, semi-span location of the pod, and created by a list of primitives like spheres, cones, cylinders, and ruled surfaces. Surfaces of different parts of the model are grouped manually, and surface and volume grids are generated at the intersection of different groups using the combined scheme provided in this paper.

Overviews of the geometry and the surface and slices of volume grids are shown in Figures 51 a and b. In this case, hyperbolic grid lines twist at the leading edge of the pylon-pod intersection and the end of the fuselage (Figure 51 c and e). Besides, grid lines also overlap at the

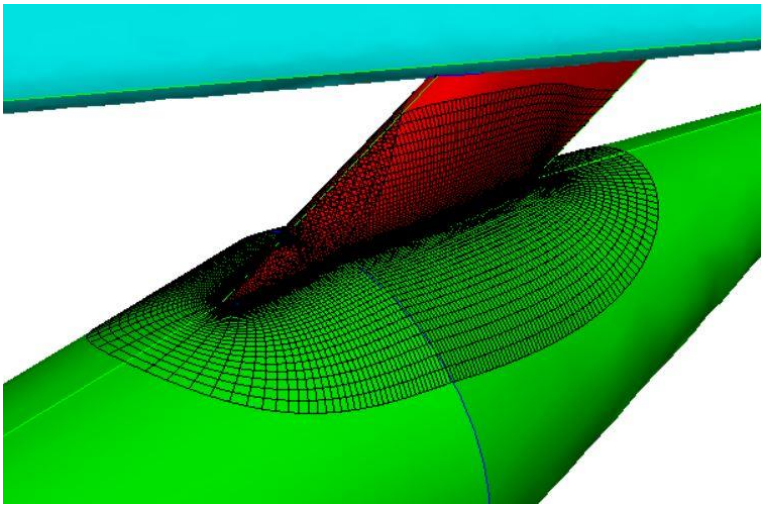
trailing edge of the wing-pylon intersection (Figure 51 g) because the inclined edges are viewed as side boundaries of hyperbolic grids. Laplace equation is applied here to guarantee smooth grids. These problems are solved using the combined scheme and are shown in Figure 51 d, f, and h separately. For the volume grids, bad grids occur in concave corners shown in Figure 51 i and k, and grid results using the combined scheme are shown in Figure 51 j and l. The computational time of generating surface and volume grids using the combined scheme is shown in Table 11 and Table 12. Grids at the tail of the fuselage, left and right wing-pylon intersection, left and right pylon-pod intersection are 6^{th}, 10^{th}, 11^{th}, 12^{th}, 13^{th} grid, respectively.



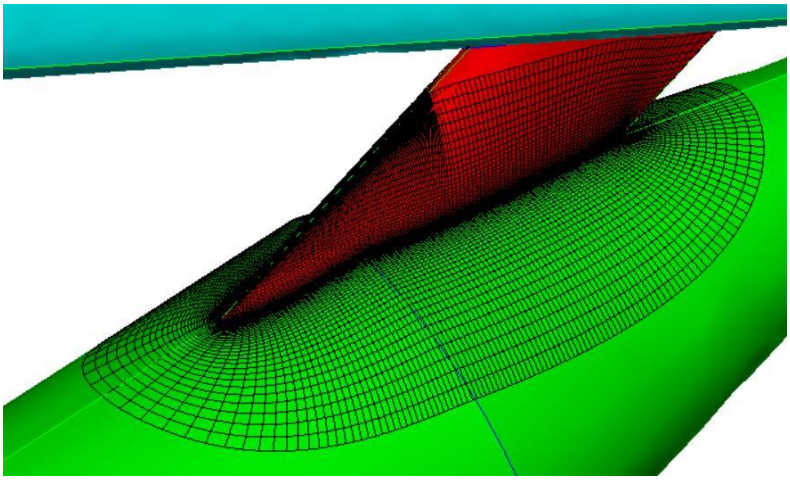
(a) Far field top view



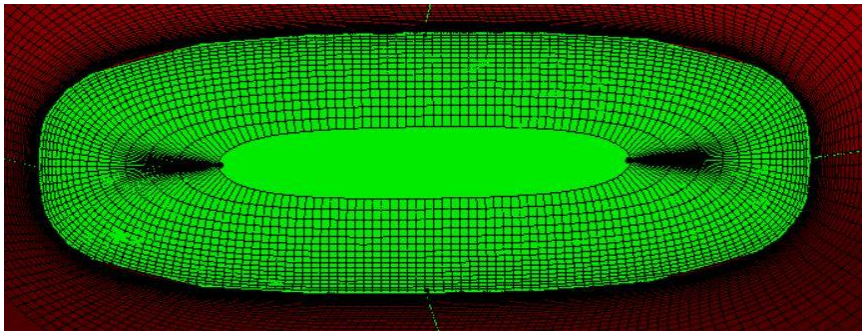
(b) Far field left view



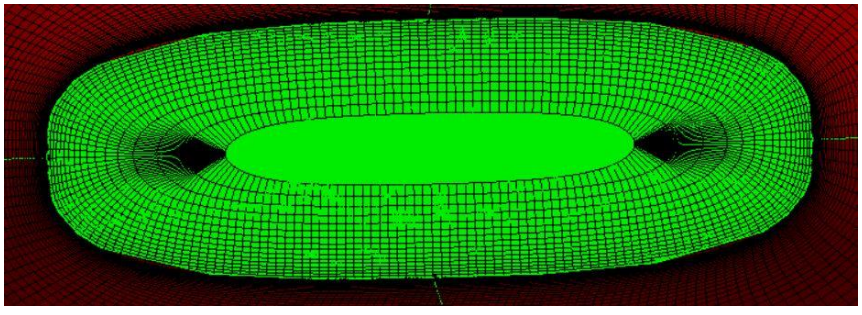
(c) Hyperbolic grids result at the leading edge of the pylon-pod intersection



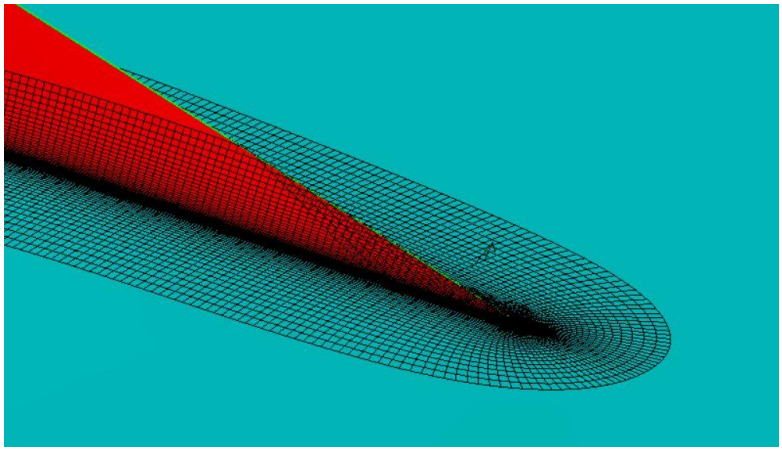
(d) Combined scheme grids result at the leading edge of the pylon-pod intersection



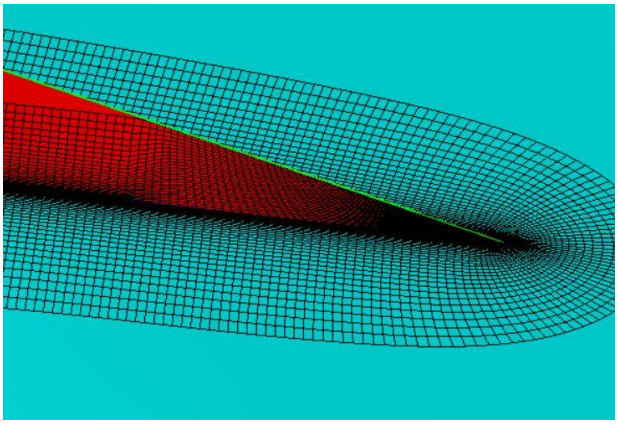
(e) Hyperbolic grids result at the end of the fuselage



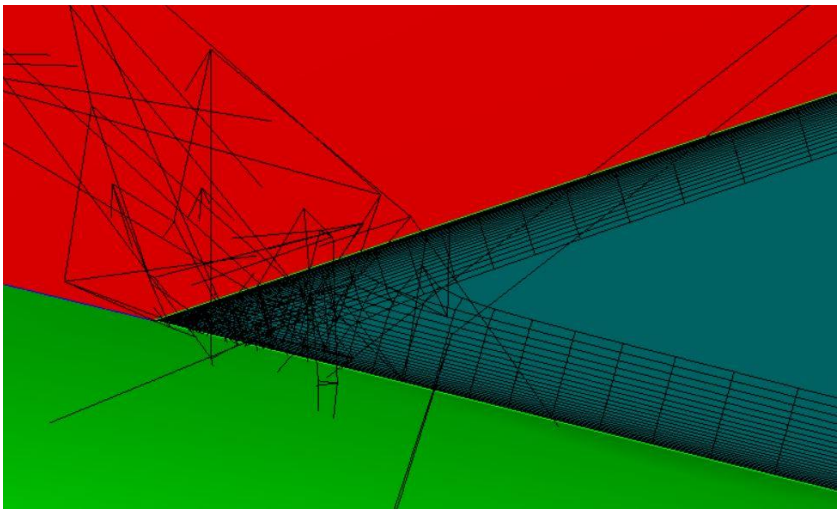
(f) Combined scheme grids result at the end of the fuselage



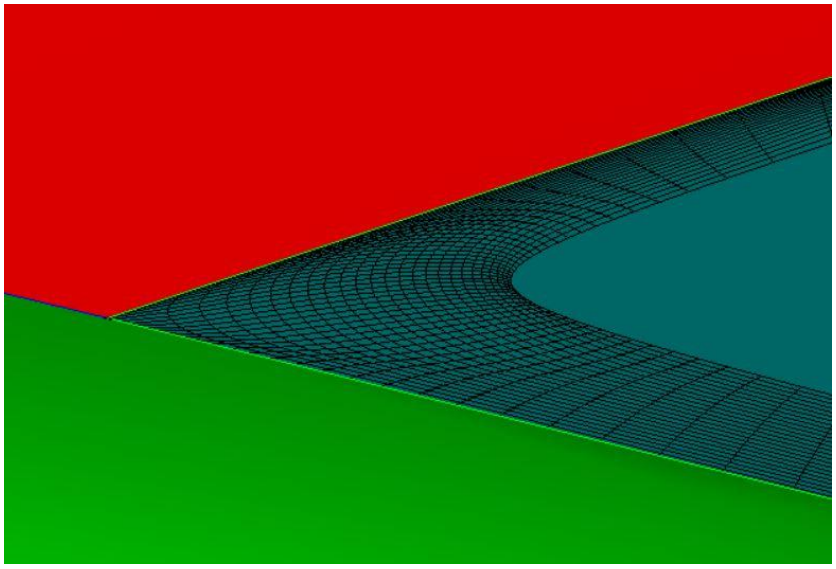
(g) Hyperbolic grids result at the trailing edge of the wing-pylon intersection



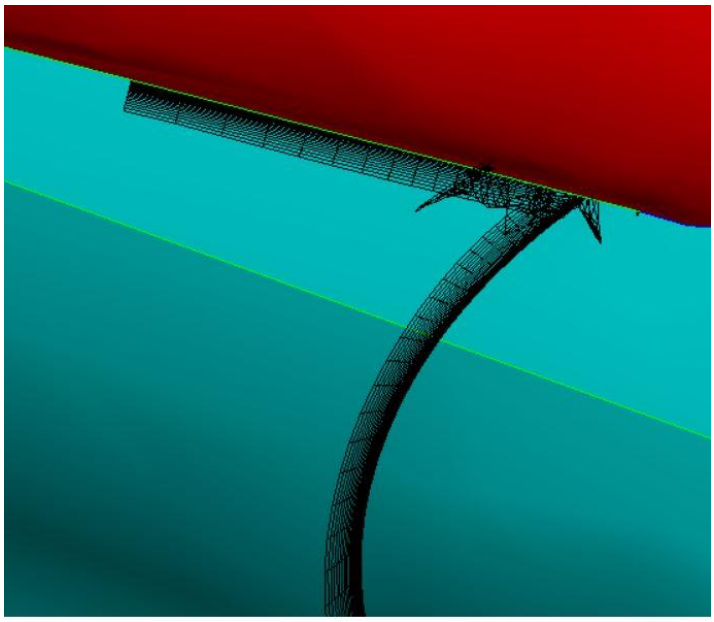
(h) Combined scheme grids result at the trailing edge of the wing-pylon intersection



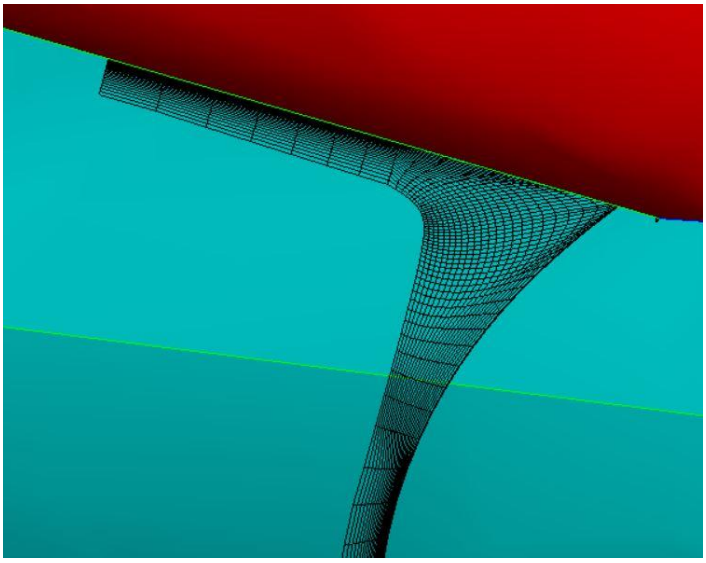
(i) Hyperbolic grids result at the trailing edge of the pylon-pod intersection



(j) Combined scheme grids result at the trailing edge of the pylon-pod intersection



(k) Hyperbolic grids result at the leading edge of the wing-pylon intersection



(l) Combined scheme grids result at the leading edge of the wing-pylon intersection

Figure 51: Surface grids and slices of volume grids of OPAM-1.

Table 11: Computational time of generating surface grids for OPAM-1 case

Grid number	Grid Size	Hyperbolic grid generation (s)	Elliptic revise (s)	Total time (s)
1	67×203	0.236	0	0.236
2	71×201	0.303	0	0.303
3	71×234	0.386	0	0.386
4	71×201	0.326	0	0.326
5	71×234	0.386	0	0.386
6	52×203	0.339	0.453	0.792
7	71×241	0.399	0	0.399
8	71×241	0.404	0	0.404
9	71×241	0.400	0	0.400
10	62×239	0.550	3.368	3.918
11	62×239	0.521	3.302	3.823
12	68×235	0.365	1.714	2.079
13	68×235	0.364	1.667	2.031
Total				15.483

Table 12: Computational time of generating volume grids for semicylinder case

Grid number	Grid Size	Hyperbolic grid generation (s)	Elliptic revise (s)	Total time (s)
1	67×203×26	2.627	0	2.627
2	71×201×26	2.812	0	2.812
3	71×234×26	3.266	0	3.266
4	71×201×26	2.798	0	2.798
5	71×234×26	3.294	0	3.294
6	52×203×26	1.947	0	1.947
7	71×241×26	3.401	0.314	3.715
8	71×241×26	3.374	48.803	52.177
9	71×241×26	3.371	0.316	3.687
10	62×239×26	2.861	22.288	25.149
11	62×239×26	2.882	20.387	23.269
12	68×235×26	3.201	76.742	79.943
13	68×235×26	3.175	67.941	71.116
total				275.80

Based on the three cases above, we can find that surface and volume grids can be automatically generated in all three cases within a reasonable time. Hyperbolic grid generation is good enough

for most of the grids. And elliptic grid generation is successfully applied at the place where hyperbolic grid generation fails.

5.6 Limitation

Although most surface and volume grids are successfully generated in Engineering Sketch Pad using the combined grid generation scheme, we still need to point out that there still are some cases that the combined technique cannot handle. One of the situations that the combined scheme will fail is generating grids at a face with a sharp convex corner like the yellow face shown in Figure 52. This figure comes from the horizontal tail of the case OPAM-1. If we want to generate grids on this surface, the grids will overlap with each other easily in the first several layers, and there is not enough place to take elliptic revise. Therefore, we will ignore these faces with sharp convex corners when generating surface and volume grids.

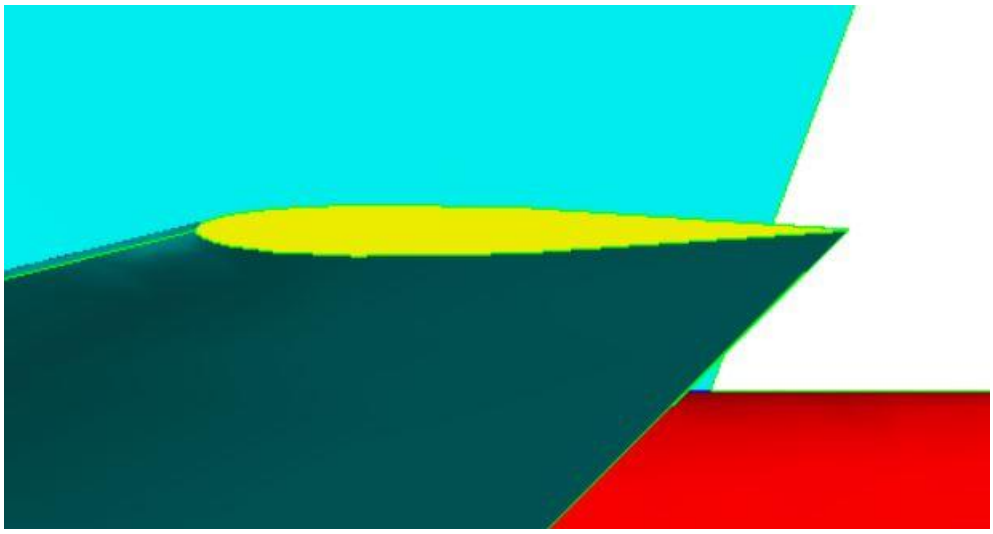


Figure 52: Example with a sharp convex corner.

Another restriction of the volume grid generation using the combined scheme is that the bad points cannot be on the four side boundaries of the volume grids. They can only exist inside the volume domain. This is because the bad points region cannot be adequately defined, and elliptic grid generation cannot be taken if bad points are found on the side boundaries. One example is shown in Figure 53, which is the last layer of the hyperbolic volume grid generation result. In this example, bad points exist on the inner loop boundary of the volume grid. It is not desirable to keep that boundary and generate an elliptic patch. But it is also problematic to discard this boundary and draw a new boundary because it crosses the geometry of the body (which is in the top right of the figure). Since the result of bad points from hyperbolic grid generation will affect the calculation of their neighbor points and the neighbor points will affect more points in the next layer, the best way to deal with this problem is to limit the number of layers of volume grid generation before the side boundaries get affected by the bad points.

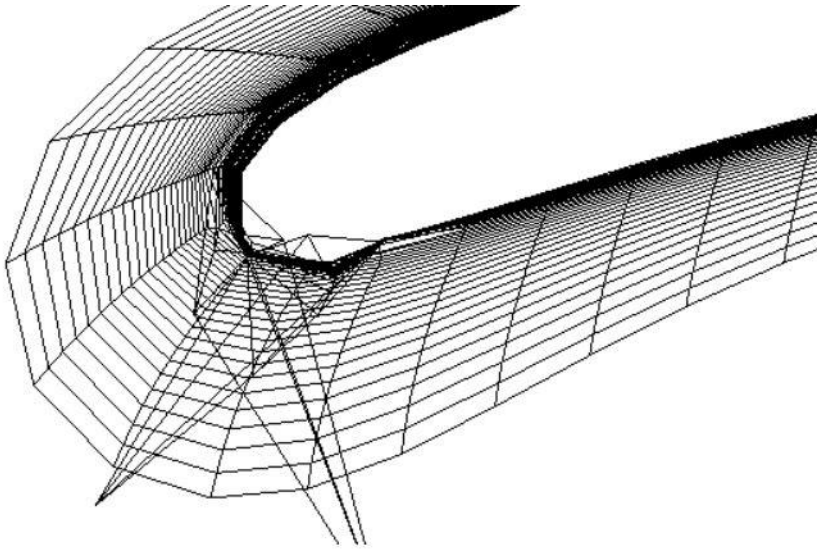


Figure 53: Example of bad grids that affect boundary points.

6. Conclusion and Future Work

6.1 Conclusion

A combined grid generation algorithm is developed, which employs the hyperbolic grid generation method in the near-body field and is further enhanced using the elliptic grid generation method in the regions where the hyperbolic grid generation method performs poorly. This algorithm exploits the advantages of both methods. The hyperbolic grid generation method creates grids quickly, and the elliptic grid generation modifies the grids resulting in no overlapping or clustering. The combined grid generation scheme is better than solely using either hyperbolic grid generation without changing smoothing parameters or the elliptic grid generation method. It is the first time that hyperbolic and elliptic methods are utilized together to generate structure grids. Additionally, the proposed method does not require user input about mechanisms like the smoothing coefficients in hyperbolic grid generation and the outer boundary definition in the elliptic grid generation method. Only essential user input like cell size and the number of grid layers are required. By implementing this proposed scheme, near-body structured grids can be generated automatically in a reasonable period of time without requiring the user to have background knowledge of grid generation. The computational speed of hyperbolic grid generation, elliptic grid generation, and combined grid generation are also investigated in several cases.

A set of bad points detection criteria is put forward for two-dimensional and three-dimensional structured grids. This set of criteria is raised based on the geometry of the grids by checking the distance between two neighbor points and the area of a grid cell in order to avoid grid lines clustering or grid lines overlapping. It can be used to check if bad points with such

problems exist in any two-dimensional or three-dimensional structured grids. A bad points grouping method is also provided that can combine the bad points into several regions, especially for three-dimensional structured grids. This method can help set apart bad points, minimize the size of bad points regions, and thereby improve the speed of calculation of the combined grid generation scheme.

The combined scheme for near-body structured grid generation is successfully tested and implemented in Engineering Sketch Pad. Initial grid points are automatically picked up on the intersection curves between every two groups of faces based on the geometry created in the software. Surface and volume structured grids are created by the hyperbolic grid generation scheme using these grid points and revised by the elliptic grid generation scheme where bad points occur. One great advantage of using the Engineering Sketch Pad to generate grids is that surfaces are grouped by attributes such that grids can be generated over several surfaces. Besides, the evaluation function in Engineering Sketch Pad is applied to calculate the derivatives and normal vector on the surfaces that can improve the accuracy of grid results.

Another contribution relies on using tessellation to locate and project grid points onto the surfaces. Projecting points onto surfaces are required in the hyperbolic grid generation scheme, and it is much faster when using tessellation and extended barycentric coordinate system to get the results than using inverse evaluation provided in Engineering Sketch Pad. However, this method should be carefully handled because it is extended from a set of two-dimensional equations. Further notices about the problems for using tessellation and the extended barycentric coordinate system can be followed based on the discussion in this paper.

6.2 Future work

Many different cases and adaptations are required and left for the future due to lack of time. Future work may concern deeper analysis of particular mechanisms like the other bad points detection criteria that are not provided in this paper, new proposals to try different parameters in hyperbolic and elliptic grid generation. Different methods of calculating outer boundaries of the bad points regions for two-dimensional and three-dimensional grid generation would also be considered to get a higher-order continuity between hyperbolic results and elliptic regions.

Two kinds of limitations of the combined scheme are discussed in Chapter 5.6: the grid cannot be generated successfully by the combined scheme on a surface with a sharp convex corner, and the number of volume grid layers should be limited to avoid bad points existing on boundaries. One of the solutions to the first problem is that grids can be generated separately based on the edge on the upper face and the lower face. Then the grid lines will not tangle with each other but overset on the surface. Extra imaginary grids points might be required to generate the grids. Other solutions may require C-type grids, which start and end in the far-field, instead of O-type grids, which are periodic around the geometry of the body.

For the second problem, it could be possible to pause hyperbolic grid generation for every five or ten layers and fix the bad points within these layers. Once the bad points are solved, another five or ten layers of hyperbolic grids can be generated based on the revised grid. Another idea to solve this problem is to build an imaginary surface boundary like the outer boundary curve for surface grid generation. However, it should be handled carefully to keep the shape of the volume grids and avoid getting across the geometry of the body. Different techniques for these problems can be discussed in the future.

Regarding overset grids, the surface and volume grids using the combined scheme in Engineering Sketch Pad are only generated at the intersection of two surfaces where grid lines clustering or overlapping is more likely to exist. This means that the entire model is not fully covered (like in the middle of the wing or fuselage). Other surface and volume grids along the geometry of the model are required to cover these areas for near body grid generation. These grids, along with far-field grids, will constitute the overset grids and be used to calculate computational fluid dynamics problems.

Reference

[1] A. Cary, J. Chawner, E. Duque, W. Gropp, W. Kleb, R. Kolonay, E. Nielsen, and B. Smith, "CFD Vision 2030 Road Map: Progress and Perspectives," *AIAA paper* no. 2021-2726, Aug 2021, https://doi.org/10.2514/6.2021-2726.

[2] K. Kao, M. Liou, Advance in Overset Grid Schemes: From Chimera to DRAGON Grids, *AIAA Journal* Vol. 33, No. 10, October 1995.

[3] J. L. Steger, F. C. Dougherty, J. A. Benek, A Chimera Grid Scheme, *Advances in Grid Generation*, K. N. Ghia and U. Ghia, eds., ASME FED-Vol. 5, June 1983.

[4] M. Akinlar, S. Salako, G. Liao, "A Method for Orthogonal Grid Generation," *Gen. Math. Notes*, Vol. 3, No. 1, pp.55-72 ISSN 2219-7184, March 2011.

[5] S. E. Rogers, N. E. Suhs, W. E. Dietz, PEGASUS 5: An Automated Preprocessor for Overset-Grid Computational Fluid Dynamics. *AIAA Journal*, 41(6), 1037–1045. https://doi.org/10.2514/2.2070

[6] R. W. Noack, D. A. Boger, R. F. Kunz, P. Carrica. Suggar++: An Improved General Overset Grid Assembly Capability. *19th AIAA Computational Fluid Dynamics Conference*, 10.2514/6.2009-3992, June 2009.

[7] W. M. Chan, S.A. Pandya, S. E. Rogers, J. C. Jensen, H. C. Lee, D. L. Kao, P. G. Buning, R. L. Meakin, D. A. Boger, S. M. Nash. Chimera Grid Tools. https://www.nas.nasa.gov/publications/software/docs/chimera/index.html

[8] W. M. Chan, The OVERGRID Interface for Computational Simulations on Overset Grids. AIAA Paper 2002-3188, 32nd AIAA Fluid Dynamics Conference, June 2002.

[9] W. M. Chan, P. G. Buning. Surface Grid Generation Methods for Overset Grids. *Computers and Fluids 24*, No. 5, 509--522, 1995.

[10] W. M. Chan, I. Chiu, P. G. Buning, User's manual for the HYPGEN hyperbolic grid generator and the HGUI graphical user interface. Technical report, NASA, October 1993.

[11] R. J. Gomez, D. Vicker, STS-107 Investigation Ascent CFD Support, American Institute of Aeronautics and Astronautics Paper 2004-2226

[12] R. Haimes, J. F. Dannenhoffer, The engineering sketch pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry. Paper presented at 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, United States, 2013

[13] J.F. Thompson, B.K. Soni, N.P. Weatherill, *Handbook of Grid Generation*. CRC Press, 1999.

[14] D. S. Chaussee, J. L. Steger. Generation of body-fitted coordinates using hyperbolic partial differential equations. *SIAM J.SCI.STAT.COMPUT.*, 1(4):431– 437, December 1980.

[15] R. Duraiswami, A. Prosperetti, Orthogonal mapping in two dimensions. *Journal of Computational Physics*, pages 254–268, April 1991.

[16] M. J. Berger, Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, August 1984.

[17] T. Tang, H. Tang. Adaptive mesh methods for one and two dimensional hyperbolic conservation laws. *Society for Industrial and Applied Mathematics*, 41 (2):487–515, 2003.

[18] Y. Rizk, J. L. Steger. Generation of three dimensional body fitted coordinates using hyperbolic partial differential equations. Technical Memorandum 85246, Ames Research Center, Moffett Field, Calif.94035, June 1985.

[19] J. L. Steger, Generation of three-dimensional body-fitted grids by solving hyperbolic partial differential equations. Technical Memorandum NASA TM-101069, January 1989.

[20] C. J.Nietubicz, E. N.Ferry. Interactive hyperbolic grid generation for projectile cfd. Technical Report BRL-MR-3971, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD 21005-5066, May 1992.

[21] C. Snell, T. Kwan, and J. Cary, User's manual for Dispak, an easy approach to DISSPLA graphics. United States: N. p., 1981.

[22] W. M. Chan, J. L. Steger, Enhancements of a three-dimensional hyperbolic grid generation scheme, *Applied Mathematics and Computation*, Oct 1992, pp. 181-205.

[23] H. A. Dwyerz, A geometric interpretation of hyperbolic grid generation. *Computers and Fluids*, pages 737–748, August 1994.

[24] C. Y. Soong, C. H. Tai, S. L. Yin, A novel hyperbolic grid generation procedure with inherent adaptive dissipation. *Journal of Computational Physics*, 116:173–179, July 1995.

[25] K. Matsuno, High-order upwind method for hyperbolic grid generation. *Computers and Fluids*, 28:825–851, August 1998.

[26] W. M. Chan, Advances in chimera grid tools for multi-body dynamics simulations and script creation. In 7th Symposium on Overset Composite Grid and Solution Technology, 2004.

[27] W. M. Chan, Overset grid technology development at NASA Ames Research Center. *Computers and Fluids*, 38:496–503, June 2009.

[28] W. M. Chan. Recent developments in chimera grid tools. In 10th symposium on overset composite grids and solution technology, 2010.

[29] T. J. Barth, D. W. Kinsey, Description of a hyperbolic grid generating procedure for arbitrary two-dimensional bodies. Technical report, Aerodynamics and Airframe Branch, Aeromechanics Division, Flight Dynamics Laboratory, July 1984.

[30] R. L. Sorenson, A computer program to generate two-dimensional grids about airfoil and other shapes by the use of Poisson's equation. NASA Technical Memorandum 81198, May 1980.

[31] J. F. Thompson, F. F. Thames, C.W. Mastin, Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, pages 299–319, March 1974.

[32] J. L. Steger, R. L. Sorenson, Simplified clustering of nonorthogonal grids generated by elliptic partial differential equations. Technical Memorandum A-6950, Ames Research Center, NASA, Moffett Field, Calif.94035, August 1977.

[33] R. L. Sorenson, J. L. Steger, Automatic mesh-point clustering near a boundary in grid generation with elliptic partial differential equations. *Journal of Computational Physics*, pages 405–410, February 1979.

[34] R. L. Sorenson, Three-dimensional zonal grids about arbitrary shapes by Poisson's equation. Technical Memorandum NASA TM-101018, Ames Research Center, Moffett Field, Calif.94035, August 1988.

[35] R. L. Sorenson, The 3DGRAPE book: Theory, users' manual, examples. Technical Memorandum NASA TM-102224, Ames Research Center, Moffett Field, Calif.94035, July 1989.

[36] S. Kim, Control functions and grid qualities measurements in the elliptic grid generation around arbitrary surfaces. *International Journal for Numerical Methods in Fluids*, 33:81–88, 2000.

[37] U. K. Kaul, New boundary constraints for elliptic systems used in grid generation problems. *Journal of Computational Physics*, 189:476–492, 2003.

[38] U. K. Kaul, Three-dimensional elliptic grid generation with fully automatic boundary constraints. *Journal of Computational Physics*, 229:5966–5979, 2010.

[39] B. K. Soni, S. H. Lee, The enhancement of an elliptic grid using appropriate control functions. *Applied Mathematics and Computation*, 159:809–821, 2004.

[40] V. Villamizar, S. Acosta, Elliptic grids with nearly uniform cell area and line spacing. *Electronic Transactions on Numerical Analysis*, 34:59–75, 2009.

[41] W. Wenli, Z. Pei, Y. l. Liu, Meshes generated by elliptic equation. In International Conference on Computational and Information Sciences, pages 58–61, Xi'an 710048, P. R. China, 2010. Xi'an University of Technology.

[42] D. A. Anderson, Equidistribution schemes, Poisson generators, and adaptive grids. *Applied Mathematics and Computation*, 24:211–227, 1987.

[43] Y. C. Liou, Y. N. Jeng. A new adaptive grid generation by elliptic equations with orthogonality at all of the boundaries. *Journal of Scientific Computing*, 7(1):63–80, January 1992.

[44] J. Sherman, W. J.Morrison "Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix (abstract)." Annals of Mathematical Statistics. 20: 621. doi:10.1214/aoms/1177729959

Curriculum Vitae

Personal Data

Name: Yuyan Hao

Date of Birth: Feb/11/1992

Address: 121 Lafayette Road Apt 231, Syracuse, NY

Zip code: 13205 Phone: 315-378-3387

E-mail: yhao08@syr.edu

School Education

Duration: 2016 – 2022

University: Syracuse University Major: Mechanical and Aerospace Engineering

GPA: 3.9 Degree: Doctor

Duration: 2014 - 2016

University: Syracuse University Major: Mechanical and Aerospace Engineering

GPA: 3.9 Degree: Master

Duration: 2010 - 2014

University: Shandong University Major: Mechanical Design Manufacture and Automation

GPA: 3.5 Degree: Bachelor

Academic Research

2016/08-2022/08 Automatic Generation of Near-Body Structured Grids

- Study hyperbolic surface and volume grid generation
- Study elliptic surface and volume grid generation
- Develop combined scheme to automatically generate surface and volume grid
- Apply the combined scheme into Engineering Sketch Pad

2015/06-2015/11 Research about B-splines and NURBS

- Used Levenberg–Marquardt algorithm to approximate a Bézier curve
- Compared B-splines with piecewise Bézier curves
- Approximated super ellipse by cubic B-splines as well as super ellipsoid by bicubic surfaces and derived asymptotic convergence rates of the approximation
- Develop the Matlab code for approximating Bézier curves as well as super ellipse and super ellipsoid approximation

2015/01-2015/04 Design for Manufacturing Analysis and Improvement about Waffle Makers

- Analyzed market requirement of waffle makers
- Examined product costs, including material and process selection, design for manufacturing and assembly analysis, life cycle analysis
- Improved product performance with four redesign solutions

2014/11-2014/12 Numerical Approximation of Traffic Flow Problems

- Analyzed existed traffic flow models
- Applied the equations in traffic flow models with different numerical approximation method, including Lax-Friedrichs, Lax-Wendroff and Runge-Kutta method
- Compared the Consistency, Accuracy, Stability and Efficiency in different method
- Analyzed local effect in traffic flow
- Develop the Matlab code for traffic flow approximation

2014/01-2014/06 Design and Kinematic Analysis about Five Degree of Freedom Printing Robot

- Analyzed the principle of RV reducer gearbox
- Selected the assembly parts in printing robots, including RV reducer gearboxes, servo motors, etc.
- Checked the stiffness requirement of the assembly parts
- Calculated the position and orientation of the printing robot based on robot kinematic equations
- Developed 3-D model and 2-D drawings of the printing robot

2012/05-2012/06 Helical Gear Reducer Transmission Design

- Selected and checked assembly parts, including the V belt, helical gears, shafts and rolling bears
- Developed CAD drawing of helical gear reducer

2011/11-2012/01 Innovative Design Based on Shaper Mechanism

- Analyzed the principle of shaper

- Compared the mechanism of shaper with two kinds of six-bar linkage
- Derived the displacement, velocity and acceleration plots by ADAMS (Automatic Dynamic Analysis of Mechanical Systems)

Internship Experience

- 2013/7-2013/8 Internship in China National Heavy Duty Truck Group CO.LTD.
- 2012/3 Internship in YTO Group Corporation
- 2012/2 Internship in Jinan First Machine Tool CO.LTD.

Publication

Yuyan H., John D., Automatic Generation of Near-Body Structured Grids, AIAA 2019-3672

Honors and Awards

- 2013/4 Shandong University 8th Innovation Competition of Electromechanical Products, First Prize
- 2012/5 Shandong University CAD Capability and Innovation Competition, Third Prize
- 2012/5 Shandong University Robot Dance Competition, Third Prize
- 2012/3 Shandong University 7th Innovation Competition of Electromechanical Products, Third Prize

Qualifications

- 2012/11 Kögel German Learning Certificate
- 2012/03 National Computer Rank Examination - 4, Network Engineering 2011/11 AutoCAD Senior Draughtsman
- 2011/09 National Computer Rank Examination - 3, Database Technology 2011/03 National Computer Rank Examination - 2, C Language

Social & Extracurricular Activities

- 2012/09-2012/10 Volunteer in Xingfujiayuan Geracomium 2011/09-2011/10 Volunteer in Shandong University Sport Festival 2011/07-2011/08 Quancheng Volunteer
- 2010/11-2011/06 Shandong University Student Union member