Advancements in Automation of Overset Structured Mesh Generation: Algorithmic Developments and Aerodynamic Applications

By

Andrew Manhei Chuen

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Mechanical and Aerospace Engineering

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Mohamed M. Hafez, Chair

William M. Chan

C.P. van Dam

Committee in Charge

2025

To my friends and family

whose support has been invaluable in this journey.

Contents

List of Figures

List of Tables

Abstract

Advancements in Automation of Overset Structured Mesh Generation: Algorithmic Developments and Aerodynamic Applications

Grid generation continues to be an onerous task when performing flow computations using computational fluid dynamics (CFD), especially in the case of generating structured overset meshes. However, the use of structured overset meshes provides significant advantages over other CFD methodologies, such as highly efficient numerical methods, ease of use of higher-order schemes, and efficient resolution of boundary layers. Despite these advantages, minimal research effort has been invested in automating the mesh generation prior to performing flow computations. However, in recent years, a surface domain decomposition approach utilizing the topology of a Boundary Representation (BRep) Computer-aided design (CAD) solid allowed for a viable starting point for automation of overset meshes. Efforts in completely automating the overset mesh generation process is still currently underway.

The present work details relevant efforts from both an algorithmic development and applied perspective that seek to further automation of overset mesh generation. The first element introduces a new boundary condition for hyperbolically-generated meshes that couples the mesh boundaries together as the meshes are generated or marched away from the initial data. This boundary condition maintains consistent and sufficient overlap (provided consistent and sufficient overlap already exists in the initial data), which is necessary for successful generation of overset grids. Functionally, the boundary condition is identical in concept to overset boundaries used for solving the Navier-Stokes equations in CFD but now applied in the context of the hyperbolic mesh generation for multiple overlapping initial curves or surfaces. As such, the process is largely automated, since existing algorithms for overset boundaries, such as performing domain connectivity, can be easily leveraged. This type of boundary condition is demonstrated for two-dimensional field meshes, three-dimensional surface meshes, and three-dimensional volume meshes.

The second element of this work is investigating the usage of the automatically-generated meshes in CFD applications for aerodynamics analyses. The meshes generated by the automation procedure are different compared to conventional, manually-generated meshes with regards to the increased quantities of overset boundaries. This is mainly due to the domain decomposition approach inherited from the input BRep topology, and it is an open question pertaining to any potential adverse effects on the flow computations. In this effort, various RANS applications are considered ranging from simple benchmark cases to full configurations of aircraft. Special attention is given to examining the numerical accuracy and the computational efficiency for each of the cases presented. Overall, it is found that the increased quantity of overset boundaries appear to have a neutral effect on the resulting flow computations.

Acknowledgments

Throughout my doctoral journey, I have benefitted greatly from the mentorship and collaboration of many of my colleagues and professors. I am especially grateful for the guidance of my advisor, Professor Mohamed Hafez, who originally encouraged me to consider graduate school. Over the past few years, I have deeply appreciated the incredible breadth and depth of his knowledge during our discussions, and I aspire to emulate his approach to problem-solving. I would also like to thank Dr. William Chan, who mentored me at NASA Ames Research Center and served on my dissertation committee. Much of this work would not be possible without our numerous discussions and his feedback. I am also further grateful for his perspectives on the field of mesh generation, which have helped to shape my own intellectual curiosity of the field. I would also like to thank Dr. Thomas Pulliam for his illuminating discussions on the nuances of the Overflow solver. I also would like to thank other members of the CGT/RVLT team at Ames: Dr. Shishir Pandya, Dr. Sheida Hosseini, and James C. Jensen. I have also benefitted from their mentorship, intellectual curiosities, and collaborations, which also appear in this work. I also would like to thank the Computational Aerosciences Branch at NASA Ames for their support, and I would also like to especially thank Dr. Cetin Kiris for having provided the opportunity to work in this branch. Additionally, I also want to thank the current branch chief Jared Duensing and deputy branch chief Jamie Meeroff for their support during the latter half of my dissertation work. I want to also express my gratitude to the faculty at the Mechanical and Aerospace Engineering Department at UC Davis for their professional and intellectual support in both my undergrad and in grad school. I would also like to give additional thanks to Professor Case van Dam for serving on my dissertation committee.

I am also grateful and indebted to my parents, whose sacrifices, efforts, and encouragement have afforded me the opportunities to pursue my dreams. I am also thankful for my friends, who have supported and walked through this journey with me. You all have helped to make my life much richer and fuller, and helped to remind me that there is more to life than work.

Finally, I would like to thank and acknowledge the funding provided by the NASA Revolutionary Vertical Lift Technology (RVLT) Project and Transformational Tools and Technologies (TTT) Project, which have been critical in supporting this work. I would also like to thank the NASA Advanced Supercomputing (NAS) Division for providing the computing resources for all the work presented in this dissertation. Lastly, I would also like to thank the Pathways Program, which provided the foundations by which all these collaborations and efforts have been made possible.

Chapter 1

Introduction

1.1 Motivation

Computational fluid dynamics (CFD) now plays a critical role in the design and analysis of many engineered products, such as commercial and military aircraft, spacecraft, and motor vehicles. In the case of modern civilian aircraft, CFD played a significant role in the design of the Boeing 777 wing at high-speed cruise, as well as the fuselage cab geometry [1]. Other aspects of the design, such as the tail, flap-track fairings, and wing-body fairings were also augmented by CFD analysis. Similarly, the Airbus A380 wing also extensively utilized CFD along with wind-tunnel testing to design a wing that satisfies both required cruise and high-lift performance to be compatible with existing airport infrastructure [2]. Additionally, as of 2010, Airbus's design process implements CFD largely for external shape design and local regions with increased geometric complexity, with increasing use in highly complex geometries or on areas requiring multi-disciplinary coupling such as aero-thermal or aero-acoustic analysis. CFD has also been responsible for significant reductions in wind-tunnel tests for validation, especially during the development of the Boeing 777 and 787 [3]. In the realm of space, CFD has helped to improve designs for launch vehicles during ascent and re-entry [4–6]. More recently, CFD has been utilized extensively to build a database for ascent and booster separation of the NASA Artemis I Space Launch System vehicle [7, 8]. The examples listed here are, of course, simply a select assortment of numerous possible examples and instances of the usage of CFD in modern aerospace engineering design.

Structured overset meshing [9, 10] is one of the main meshing and domain decomposition paradigms used for CFD analysis. This approach enables the usage of logically-rectangular, curvilinear, body-fitted meshes in complex flow domains where the domain cannot be easily represented by a single structured grid. This is achieved through decomposing the flow domain into multiple smaller, overlapping grids, and inter-grid communication occurs through interpolation. This approach offers more flexibility than a *multi-block* approach, since there is no requirement that the boundaries of the individual grids must abut and match with another grid. Overset also provides advantages over other meshing approaches such as unstructured

or Cartesian meshing. The usage of logically-rectangular, curvilinear grids allows for improved resolution of boundary layer regions through high-aspect ratio cells, as well as more efficient solvers due to easy traversal through the grid. Overset is also especially favorable for situations involving relative motion, such as simulating rotating propeller or rotor blades. In such scenarios, only the inter-grid connectivity must be updated without requiring the grids to also be regenerated during the dynamic motion. This meshing approach has been successfully used in CFD in a large variety of applications [11–20] and is also intimately tied with the history of CFD usage in engineering [1], especially through the Overflow flow solver [21].

However, one common roadblock in CFD analysis is the effort required to properly generate a mesh. In their conception of CFD in the year 2030, Slotnick et al [22] envisioned the state-of-the-art CFD capability with "a much higher degree of automation in all steps of the analysis process...including...mesh generation". At the time of writing in 2014, the authors also listed several issues with the current state-of-the-art, including inadequate linkage with CAD and poor mesh generation performance and robustness. As a result, the mesh generation process in general often requires significant human intervention and is onerous when dealing with complex geometries. Interestingly, this has been a prevalent issue that has been raised many times throughout the last four decades of CFD and numerical simulation research, although without much successful efforts in tackling such problems. Chawner et al [23] traces the history of similar issues and the evolution of the problems discussed as far back as 1984. In particular, Thompson et al [24] outlined specific challenges with surface meshing and working with complex geometries in 1988, and Thompson [25] called for the need for automation and close interfacing with geometry as early as 1996.

Specifically for overset meshes, mesh generation continues to be an onerous task, especially with regards to the task of surface mesh generation. Of the three steps necessary to perform overset mesh generation (surface meshing, volume meshing, and domain connectivity), surface mesh generation can take up to 80% of the total time needed by the mesher [26]. This is because the surface mesh generation step typically requires the user to process nearly every geometric feature (faces, edges, nodes) that is present in a CAD model. The user must then decompose the surface geometry into a set of overlapping surfaces while enforcing the desired local parameterizations or refinements. This effort scales significantly as the geometry increases in complexity and/or features. Volume mesh generation, on the other hand, is not as labor intensive since the meshes need only march outwards using hyperbolic grid generation [27]. This step is often iterated along with the domain connectivity step to ensure that there is sufficient overlap between all volume grids, as well as trimming parts of meshes lying outside of the flow domain through hole-cutting [28]. The deep level of manual intervention required at every step illustrates how daunting the task can be to generate a mesh by hand for geometries more complex than a box or a sphere. It underscores the need to develop methodologies to automate such a task, especially to take advantage of

the various benefits with regards to computational efficiency and accuracy that structured meshes offer.

Recently, Chan et al. [29] demonstrated an approach to automate the domain decomposition and surface mesh generation starting from a Boundary Representation (BRep) CAD model. This nascent technology shows potential towards a path for automation of structured overset mesh generation for more complex configurations and can reduce the overall time and effort required to perform CFD analyses. However, since the technology is still in its early stages, additional development efforts are necessary to enable automation at all steps of overset mesh generation and enhance the viability of the approach in typical CFD workflows. This work investigates various advancements towards this effort through technical development and applications.

1.2 Background

The preliminary objective of any CFD analysis is to solve some form of the (or in some cases, the full) Navier-Stokes (NS) equations. The NS equations are given by

$$\frac{\partial}{\partial t}\begin{vmatrix}\rho\\ \rho\vec{u}\\ \rho E\end{vmatrix} + \nabla\cdot\begin{vmatrix}\rho\vec{u}\\ \rho\vec{u}\otimes\vec{u}+p\overline{I}-\overline{\tau}\\ \rho\vec{u}H-\overline{\tau}\cdot\vec{u}-k\nabla T\end{vmatrix} = \vec{0} \tag{1.1}$$

which represent the conservation of mass, momentum, and energy. Here, $\vec{u}$ is the fluid velocity, $\overline{\tau}$ and $\overline{I}$ are the viscous shear-stress and unit tensors, respectively. The scalar variables ρ, p, k, E, H, T, represent density, pressure, heat transfer coefficient, specific total energy, specific total enthalpy, and temperature, respectively. Note the zero vector on the right-hand side indicates the fluid is adiabatic and not subject to any external forces. With the assumption of a Newtonian fluid, the viscous stress tensor $\overline{\tau}$ is given by

$$\overline{\tau}_{ij} = \mu\left(\partial_i u_j + \partial_j u_i\right) + \lambda\left(\nabla\cdot\vec{u}\right)\delta_{ij} \tag{1.2}$$

where μ and λ are the dynamic and second viscosity coefficients, respectively. The Stokes relation $3\lambda + 2\mu = 0$ is typically used to define λ. Under the assumption of an ideal gas, the system is closed by the following thermodynamic relation,

$$E = c_v T + \frac{1}{2}\|\vec{u}\|^2 = \frac{1}{\gamma-1}\frac{p}{\rho} + \frac{1}{2}\|\vec{u}\|^2, \tag{1.3}$$

where c_v is the heat capacity of the fluid at constant volume and γ is the heat-capacity ratio. These equations are a system of nonlinear partial differential equations, which are numerically solved within a finite, closed domain (or control volume) such that the important flow features lie within the domain, and the boundary conditions can be easily applied by the CFD analyst. This closed fluid domain is what is meshed by the analyst. Figure 1.1 shows examples of the reduced flow domains for both external and internal flow.

Structured, curvilinear meshes offers several advantages when performing CFD, especially in the case of solving the Reynolds-Averaged Navier-Stokes (RANS) equations. The logically-rectangular structure

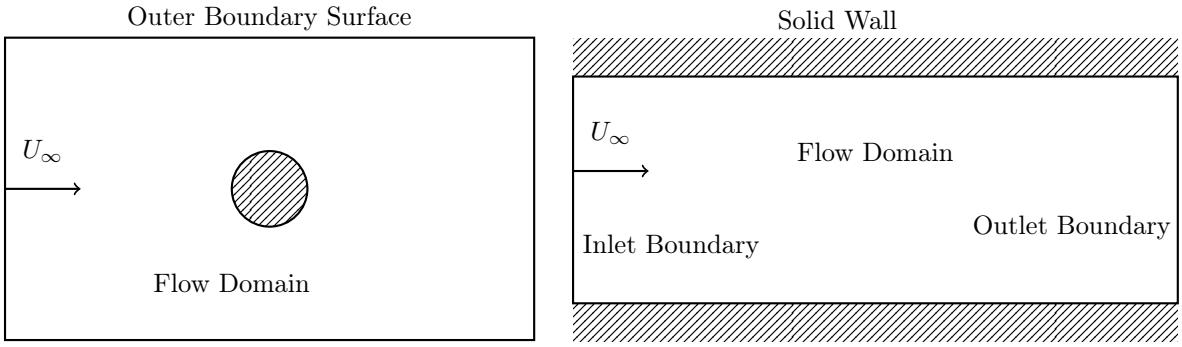


Figure 1.1: Closed fluid domain regions for both (a) external flow and (b) internal flow modeled in CFD. The solid lines represent the boundaries of the fluid domain, and the hashed fill pattern indicates boundaries with solid walls. Boundary lines without hashed fill patterns indicate flow-through boundaries.

of the mesh allows for efficient traversal through the discretized domain. Additionally, because curvilinear meshes are *body-conforming*, the meshes can provide efficient resolution of near-wall turbulence using high aspect-ratio cells, which is critical for computing RANS solutions of boundary layers.

Another advantage is that the equations can be easily discretized using finite-difference schemes. This also allows for the discretizations to be easily extended to higher order if desired. This can be shown by re-casting Equation (1.1) into the curvilinear coordinate system (ξ,η,ζ), which is given by,

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial \xi} + \frac{\partial \vec{F}}{\partial \eta} + \frac{\partial \vec{G}}{\partial \zeta} = \frac{\partial \vec{E}_v}{\partial \xi} + \frac{\partial \vec{F}_v}{\partial \eta} + \frac{\partial \vec{G}_v}{\partial \zeta} \tag{1.4}$$

where,

$$\begin{gathered}
\vec{Q} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho\vec{u} \\ \rho E \end{bmatrix}, \vec{E} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho U\vec{u} + p\vec{\nabla}_{xyz}\xi \\ U\left(\rho E + p\right) \end{bmatrix}, \vec{F} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho V\vec{u} + p\vec{\nabla}_{xyz}\eta \\ V\left(\rho E + p\right) \end{bmatrix}, \vec{G} = \frac{1}{J}\begin{bmatrix} \rho W \\ \rho W\vec{u} + p\vec{\nabla}_{xyz}\zeta \\ W\left(\rho E + p\right) \end{bmatrix}, \\
\vec{E}_v = \frac{1}{J}\begin{bmatrix} 0 \\ \overline{\tau}_{xyz}\cdot\vec{\nabla}_{xyz}\xi \\ \vec{\nabla}_{xyz}\xi\cdot\vec{f}_5 \end{bmatrix}, \vec{F}_v = \frac{1}{J}\begin{bmatrix} 0 \\ \overline{\tau}_{xyz}\cdot\vec{\nabla}_{xyz}\eta \\ \vec{\nabla}_{xyz}\eta\cdot\vec{f}_5 \end{bmatrix}, \vec{G}_v = \frac{1}{J}\begin{bmatrix} 0 \\ \overline{\tau}_{xyz}\cdot\vec{\nabla}_{xyz}\zeta \\ \vec{\nabla}_{xyz}\zeta\cdot\vec{f}_5 \end{bmatrix}
\end{gathered} \tag{1.5}$$

Here, $\overline{\tau}_{xyz}$ and $\vec{u}$ indicate the viscous stress tensor shown in Equation (1.2) and fluid velocity expressed in Cartesian coordinates. The vector $\vec{f}_5$ represents the work done by viscous forces and heat conduction, which is given by,

$$\vec{f}_5 = \overline{\tau}_{xyz}\cdot\vec{u} + \frac{\mu}{(\gamma-1)\,Pr}\vec{\nabla}_{xyz}a^2 \tag{1.6}$$

where a is the speed of sound and Pr is the Prandtl number. The *contravariant* velocity components U,V,W are given by,

$$U = \vec{\nabla}_{xyz}\xi\cdot\vec{u}, \quad V = \vec{\nabla}_{xyz}\eta\cdot\vec{u}, \quad W = \vec{\nabla}\zeta_{xyz}\cdot\vec{u} \tag{1.7}$$

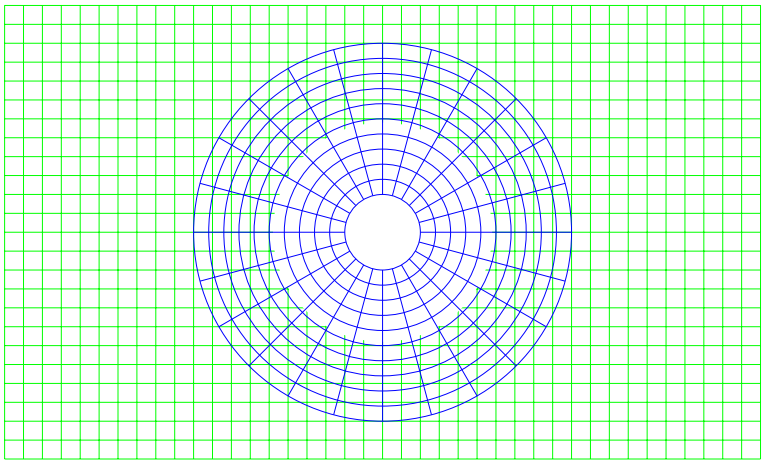


Figure 1.2: Example overset mesh for external 2D flow over a cylinder (see fig. 1.1a). The region surrounding the cylinder is meshed with a conformal near-body mesh (blue), and the rest of the flow domain is meshed with a background Cartesian mesh (green).

Here, J represents the Jacobian of transformation, which is given by $J = \det\left(\partial X(\xi, \eta, \zeta)/\partial(\xi, \eta, \zeta)\right)$. The spatial derivatives along ξ, η, and ζ in Equation (1.4) can be discretized using finite-differences in each direction using the desired discretization and order of accuracy. Likewise, the time derivatives can be separately discretized using the method of lines and the desired scheme and accuracy. This discretization is much easier to apply and solve in contrast to unstructured approaches, which involve complex volume integrations that are not as easy to increase the order of accuracy. Additionally, for structured meshes, the discretizations also result in a sparse, banded matrix structure when solving the equations implicitly. These systems can be solved efficiently in comparison to the sparse matrix produced by unstructured meshes, which may not have as nice properties.

The trade-off of these specific advantages over other paradigms (unstructured, Cartesian) is that the mesh generation can be much more complex and onerous. For example, even the simple flow domain of external flow over a cylinder in Figure 1.1a can require more effort comparatively for structured grids. In a multi-block approach, the CFD analyst must break up the domain into multiple abutting, logically-rectangular regions. The overset approach, on the other hand, eases the effort here somewhat by allowing a user to generate a near-body mesh for the cylinder and embed it in a Cartesian background mesh (see Figure 1.2). *Hole-cutting* and *domain connectivity* procedures can be performed to ensure proper communication between both grids through interpolation, which occurs at *fringe points*. Fringe points are usually located at the external boundaries of a grid that overlaps with another grid, or at the points surrounding internal holes that do not contain sufficient support for evaluating the flow solver numerical discretization. The user effort here is comparable to unstructured and Cartesian meshing, since these procedures are largely automated with some moderate user input. However, for more complex and three-dimensional flow cases (e.g., the outer-mold line, or OML, of an aircraft), an additional *surface*

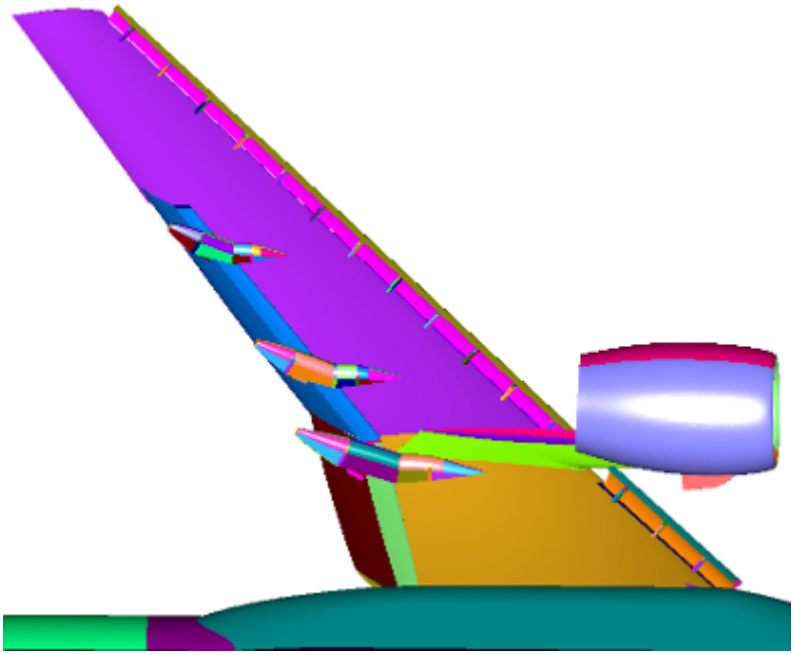


Figure 1.3: High-lift configuration with detailed high-lift devices.

meshing step is required. Consider an example high-lift configuration shown in Figure 1.3. Both the multi-block and overset approaches require high user attention and intervention to properly decompose and generate the surface mesh of the walled boundaries into logically-rectangular regions. Using a single mesh to represent the geometry surface is not possible here, as the surface cannot be represented by a single mesh without severely poor resolution of important geometric features. On the other hand, the unstructured and Cartesian meshing paradigms can almost automatically generate a suitable surface mesh or representation, respectively, on the first attempt. However, additional effort is typically necessary to ensure proper local resolution on the geometry.

1.2.1 Algebraic and Elliptic Grid Generation, and Multi-Block Grids

The earliest form of structured grid generation in CFD involved algebraic techniques, or transfinite interpolation (TFI), where the mesh generator constructs the interior mesh points of a structured block by using univariate interpolation from the boundary data of the block in each dimension [30–33]. The result is a multi-dimensional polynomial that maps the computational domain ξ, η, ζ to the physical domain $\vec{r}(\xi, \eta, \zeta) = [x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta)]^T$. Various interpolation approaches can be used, such as linear interpolation, Lagrangian interpolation, or Hermite cubic interpolation. Linear interpolation simply allows for the mesh coordinates to be specified at the boundary, but the higher-order polynomial interpolations also enable derivatives to be specified which can be used to further control the mesh orthogonality.

Eventually, techniques utilizing partial differential equations (PDE) were developed to generate struc-

tured meshes. The first type of PDE-based approaches were derived from elliptic systems. This approach was originally developed by Thompson [34], and there is a rich history of development of this technique [35–40]. Elliptic grid generation can produce smooth, high quality grids due to the maximum principle inherent in an elliptic formulation, and numerous techniques exist that can solve the system. Additionally, the development of elliptic and TFI methods are also closely tied with the development of the multi-block approach for handling complex geometries [41–48]. However, numerical techniques for solving elliptic systems can be computationally expensive relative to other differential approaches, and all the boundary data must either be available or generated first before the mesh can be generated. In the case of three-dimensional meshes, this may sometimes require solving a separate elliptic system on the boundary surfaces before the full block can be generated. The approach is also further complicated through specifications of boundary control terms that appear on the right-hand side as a Poisson-type formulation. These boundary control terms are also necessary to ensure near-wall clustering for viscous simulation of boundary layers, which further adds additional complications for RANS simulations.

1.2.2 Hyperbolic Grids

An alternative to the elliptic approach is the hyperbolic-based PDE approach. The system for generating a two-dimensional field mesh was first developed independently by Starius [49] and Steger and Chaussee [50], where grids were generated on a two-dimensional xy plane where $(x, y) = (x(\xi, \eta), y(\xi\eta))$. Steger and Chausse's approach builds a hyperbolic system from enforcing orthogonality conditions. A second condition that is applied to close the system is to enforce a positive Jacobian of transformation. This results in the following system,

$$\begin{aligned} x_\xi x_\eta + y_\xi y_\eta &= 0 \\ x_\xi y_\eta - y_\xi x_\eta &= J^{-1} = \Delta A \end{aligned} \tag{1.8}$$

where the Jacobian determinant is also set to the local cell area $\Delta A = r_\xi \cdot r_\eta$. The values r_ξ and r_η are the local grid spacing along ξ and η, and the quantity r_η is user specified. Furthermore, the approach only requires geometry data for the initial curve at $\eta = 0$ before the grid is generated. The grids are generated by marching along η, which functions as the "time-like" coordinate. The resulting system can be efficiently solved using numerical methods for hyperbolic conservation laws. Additionally, boundary information is not needed at the end of the η parametric since it is an initial-value problem. Starius's approach [49] has a similar formulation,

$$\begin{aligned} x_\eta &= -F y_\xi \\ y_\eta &= F x_\xi \end{aligned} \tag{1.9}$$

but instead, the aspect ratio $F = f/r_\xi$ is user-controlled. The user must select a function f such that $\partial f/\partial r_\xi < 0$ or is strictly decreasing with respect to the arc-length of the front. Following the

developments of Steger and Chaussee, improvements to the implicit marching scheme were explored by Kinsey and Barth [51], and Cordova and Barth [52] incorporated non-orthogonal source terms.

The approach was then extended to three-dimensions for volume mesh generation by Steger and Rizk [53, 54], where the grids march orthogonally from an initial surface. The three-dimensional system enforces orthogonality of the marching direction with respect to the surface derivatives, which define two of the equations in the system. The final equation closes the system by enforcing a positive Jacobian of transformation in the three-dimensional mapping. The system is given by the following,

$$\begin{aligned} \vec{r}_\xi \cdot \vec{r}_\zeta &= 0 \\ \vec{r}_\eta \cdot \vec{r}_\zeta &= 0 \\ \vec{r}_\zeta \cdot (\vec{r}_\xi \times \vec{r}_\eta) &= \Delta V \end{aligned} \tag{1.10}$$

where ΔV is the local cell volume. The solutions are now marched in ζ, which now functions as the time-like derivative. An initial surface is provided, which is parameterized in ξ and η.

Chan and Steger [27] introduced several enhancements for the robustness of hyperbolic volume mesh generation. This includes incorporation of spatially-varying smoothing terms, as well as additional extrapolation boundary conditions such as "splay" boundaries when grid boundaries were not coincident with symmetry planes or were not periodic. An extension for surface meshing was initially introduced by Steger [55, 56] and further developed by Chan and Buning [57]. This system is similar in form to Equation (1.8), with added terms to ensure local conformity with the reference surface through the surface normal. The system is given by,

$$\begin{aligned} \vec{r}_\xi \cdot \vec{r}_\eta &= 0 \\ \vec{n} \cdot (\vec{r}_\xi \times \vec{r}_\eta) &= \Delta S \\ \vec{n} \cdot \vec{r}_\eta &= 0 \end{aligned} \tag{1.11}$$

where $\vec{n}$ is the surface normal of the reference surface.

1.2.3 Overset (Chimera) Grids

Although Steger et al [9, 10] are credited for introducing the usage of overset, or "Chimera" grids, in the 1980s for CFD and general aerodynamics applications, the usage of overlapping grids for solving PDEs actually has a slightly longer history. Overlapping grids were first used by Volkov [58,59] in the late 1960s for solving elliptic PDEs. Nearly a decade later in the late 1970s, Starius [49,60,61] developed a composite grid approach for solving elliptic and hyperbolic PDEs in two-dimensions, as well as an early form of hyperbolic grid generation. This approach exhibits some of the basic features of the modern technique of overset meshing, such as a hyperbolic mesh that is generated around a complex boundary and then overlaid onto a background Cartesian mesh. In this case, however, the complex boundary is actually the outer boundary of a closed domain instead of the wall boundaries surrounding a closed geometry

in external flow. Kreiss [62] also demonstrated a simplified version of this approach by replacing the hyperbolic meshing with TFI applied to an annular region.

Overset in its modern form, however, was independently developed at NASA in the early 1980s by Steger et al [9, 50] for the purposes of addressing external flow problems with relative motion [63]. This, then, resulted in the development of a set of CFD analysis tools specifically for usage with overset grids. The main approach for CFD analysis in this framework is to generate the mesh (surface and volume) and perform domain connectivity, which includes hole-cutting and locating donor stencils for the interpolation points. The user can then compute the flow solution and perform the desired analysis. The development of hyperbolic grid generation and its extension to three-dimensions [27,53,54] detailed in the previous section are closely tied to the development of overset grid technology. Parks et al [64] introduced the concept of a *collar* mesh, which creates a mesh for connecting two geometric components at junction regions. Meakin [65] also developed an approach for automatic generation of off-body background Cartesian meshes. Chan then compiled the mesh generation tools into a set of pre and post-processing utilities called Chimera Grid Tools (CGT) [66]. Rogers et al [67] developed the PEGASUS5 code, which performs hole-cutting using Cartesian maps and attempts to optimize domain connectivity based on compatibility of donor and interpolant cells. Later on, Meakin [28] developed a simplified approach for performing hole-cutting using object X-rays that enabled efficient hole-cutting for bodies in relative motion. The Overflow flow solver [21,68] was developed at around the same time motivated by the need to perform flow analyses of the Space Shuttle in the late 1980s, which helped to drive much of the aforementioned development and demonstrated the viability of the overset approach [69]. Two decades later, Liou and Zheng [70] introduced the DRAGONFLOW solver, which combined OVERFLOW and USM3D (an unstructured CFD code at NASA) to solve the Navier-Stokes equations solvers on hybrid grids. This approach maintained the benefits of structured meshes while replacing mesh overlap regions with unstructured meshes, allowing the solver to avoid potential errors in interpolation and flux conservation across overset boundaries.

Efforts in developing the overset approach were not limited solely to NASA. Overture [71] is an overset framework for a general-purpose (multi-) physics PDE solver. This effort came from further developments [72] upon the CMPGRD code, which came from the methods developed by Kreiss [62]. The overset approach has also been successfully applied to unstructured meshes [73], which allowed for capability to handle relative motion problems without significant additional developmental effort.

1.2.4 Development of Automatic Structured Overset Grid Generation

In the decades following the introduction of overset meshing for aerodynamic applications in the 1980's, only a modest effort has been made to automate the mesh generation. One of the first approaches to automation by Chan and Gomez [74] in 1999 attempted to automatically generate surface meshes around feature curves and nodes in the geometry. This approach identified these topological features based on heuristic measures of the local geometry, and then generated either a hyperbolic or polar mesh for

each extracted curve or node, respectively. This approach was not fully automatic, since the hyperbolic marching distances had to be specified by the user. Additionally, the polar mesh geometry exhibits characteristics that were inefficient for a flow solver. In 2009, Dannenhoffer and Davis [75, 76] introduced an approach where they utilized the information directly from a geometry's CAD feature tree. The feature tree guided the mesh generation process by determining which meshes to generate based on the primitives in the CAD. Once the meshes are generated, connectivity can be performed by utilizing the same Boolean operations (i.e. union, intersect, and difference) present in the CAD on the generated meshes. The main difficulty with this approach, however, is the fact that most applications do not have such a feature tree available. Another early approach by Pandya et al [77] attempted to automate mesh generation through expansion of the script library in CGT. However, this approach was only limited to rocket geometries. This approach was also not fully automatic in that the user first had to manually create a grid generation script.

Nearly a decade later, Chan [78] introduced another strategy to automate the surface mesh generation process through utilizing a surface triangulation. Developing an approach based on surface triangulations was chosen since this allows for the approach to be CAD "agnostic", since most CAD software can output a triangulation. This strategy operates by generating a mesh using TFI at all regions that can be represented by a logically-rectangular mesh. The remaining un-meshed surface regions would then be meshed using hyperbolic meshes.

In recent years, an approach based on decomposition of a solid CAD model, or BRep solid [79], has shown promising success in automating overset grid generation. This approach began with Pandya et al [80] improving upon the original efforts of [74] by adding a procedure to automate the marching distance prediction through path tracing on the CAD geometry and detecting collisions with geometric features that can result in poor grid quality. They also replaced polar meshes with a more conventional patch grid without any singularities to make the grids more amenable with the solver. A pre-processor step of the CAD geometry allowed for the scheme to interface with the original geometry, and to automate the extraction of the surface feature curves. The Engineering Sketch Pad (ESP) [81] provided the interface to the solid geometry representation through its EGADS API. In 2019, Chan et al [29] continued this effort by combining an updated seam mesh generation approach with utilization of the face mesh parameterization from the CAD model. Here, the geometric entities that make up a solid in a BRep CAD model, such as faces, edges, and nodes, can be utilized to produce an overset surface decomposition for generating an overset mesh. *Node* and *edge* surface meshes are now generated, which can be constructed directly from the CAD model in lieu of seam meshes. The meshes then patch together the *face* meshes that come from the original face definitions from the CAD model. The automatic generation of these meshes are also facilitated by the *topology*, or relationships of the various geometric entities with each other.

Additional improvements were also made to the node and edge mesh generation, including improvements to grid point distributions. A new mesh type was also incorporated that allowed for handling of finite-thickness trailing edge topologies. This effort first demonstrated that the automation of structured overset surface mesh generation can be achieved using a multi-stage approach with a BRep solid definition of the geometry. This effort has since been improved upon to further automate the subsequent volume mesh [82] and domain connectivity steps [83]. Figure 1.4 shows the relationships and data of the topology and geometry stored within a BRep CAD model, which are exploited in the structured overset mesh automation scheme.

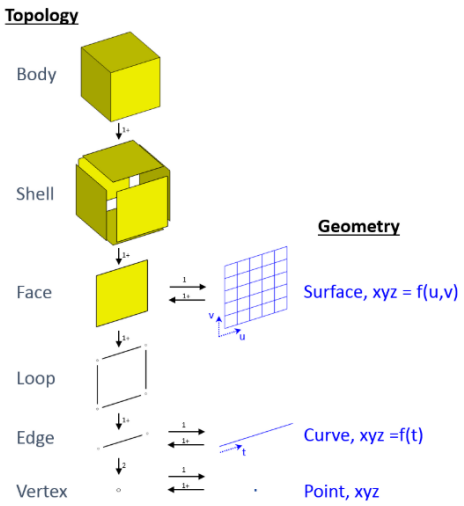


Figure 1.4: Topological and geometric data and relationships stored within a BRep solid CAD model as illustrated by Gammon et al [79, Figure 1].

However, while this recent progress presents a viable path to automation of structured overset grid generation, in practice robustness issues with the individual steps of the procedure prevent it from currently being fully automatic. In a recent paper, Chan et al [82] outlined five over-arching problems that must be addressed to ensure successful automation of structured overset mesh generation:

1. Surface domain decomposition - the geometry to be meshed must be able to be decomposed into a set of structured surfaces

2. Surface grid-point distribution - the placement of grid points must be able to resolve geometric features, as well as anticipated flow-field features and gradients

3. Surface marching scheme - edge and node mesh generation must be able to be generated through automatic selection of either algebraic or hyperbolic grid generation, as well as an appropriate estimate of the surface marching distance

4. Mitigation of negative cell areas and cell volumes for hyperbolic meshes - selection of mesh generation parameters for choice of boundary conditions and smoothing to ensure high-quality grids can be a convoluted process, especially if the initial surfaces are highly concave.

5. Ensuring mesh overlap quality - sufficient overlap between individual grid blocks (i.e. $2 \times n_f$ cell overlap, where n_f is the number of fringe points needed for supporting the chosen numerical scheme in the flow solver in a given dimension) must be guaranteed to ensure a high-quality overset mesh with proper inter-grid communication

The work of Chan et al [29] in 2019 thus far has addressed the issues of surface domain decomposition, surface grid-point distribution, and selection of surface marching type and distance. The usage of a BRep CAD model facilitates surface domain decomposition, since the CAD model already contains the actual geometry data, along with the relationships between each geometric entity (face, edge, or node) in the topology data. The geometry information also informs the surface grid-point distribution through local surface gradients. Eliminating or reducing negative cell areas and volumes, however, is still an on-going issue since the original development of hyperbolic mesh generation.

Chan et al has also partially addressed the issue of mesh overlap. In general, the lack of overlap originates from the process of hyperbolic mesh generation applied to many overlapping surfaces that are generated independently of each other. A common approach when generating a hyperbolic grid is to apply an extrapolation or "splay" boundary condition at the boundaries, unless the boundary is periodic or lies on a constant plane. A splay parameter is defined to adjust the extrapolation stencil to allow the boundaries to bend away from the interior of the grid, but the exact location of the resulting grid is not known a priori [27]. Typically, this parameter requires manual iteration until sufficient overlap is achieved. In the automation scheme, this is required if the resulting overlap is insufficient. Improper splay parameter selection can result in gaps in the surface or in the flow domain between grids. The current scheme attempts to fill these gaps with face meshes in the surface meshing step and the background Cartesian mesh in the volume meshing step. However, this can result in lower-quality overlap and can introduce orphan points, where the interpolated points are not able to find a sufficient quality donor stencil in a neighboring grid. This problem is also further exacerbated due to the surface meshing approach. The current automation scheme generates an overset surface mesh by effectively generating a grid for each face, edge, and node geometry present in the CAD model. Hyperbolic surface meshes can be generated from the extracted edges and nodes. Likewise, once the surface meshing process is complete, the volume meshes are then generated using all surface meshes as initial conditions in hyperbolic volume grid generation. The overlap issue can be present throughout the entire mesh since there can be many locations where two adjacent hyperbolic meshes are generated next to each other, which can make the repair efforts particularly burdensome.

An additional concern is related to the quantity of near-body meshes that the automation scheme generates. Because the scheme generates a near-body mesh for nearly all faces, edges, and nodes from the original CAD model, the quantity of individual grid blocks is highly dependent on how the original CAD model is constructed. It can generate a surface mesh consisting of potentially hundreds of individual grids, even for geometries as simple as a wing-body. In comparison, a typical CFD analyst may manually generate an overset mesh that contains far fewer grids. One aspect of this concern is with regards to *numerical accuracy* and turnaround time of the flow computation. The larger quantity of meshes also generally increases the interpolation and inter-grid communication, and it is currently not known how

this may impact aspects of the flow computation.

1.3 Goals of the Present Study

The focus of this dissertation is to continue on the efforts in automating meshing capabilities originally presented by Chan et al [29]. Specifically, this work seeks to address some of the gaps in the technological development detailed in the previous section, namely improving overlap issues present in hyperbolic grids and investigating the feasibility of the resulting grids for CFD analysis. Chapter 2 provides a review of the current overset mesh generation technology considered, such as the automatic mesh generation scheme itself and numerical methods for hyperbolic grid generation. Chapter 3 provides an overview of the numerical methods used in the flow computations. Chapter 4 is devoted to addressing the issue of consistent overlap between meshes generated using hyperbolic schemes. Here, taking advantage of the pre-existing sufficient overlap in the initial data, a different set of boundary conditions akin to overset is formulated and demonstrated for hyperbolic grid generation. Chapter 5 then addresses the verification and viability of the automatically-generated grids in CFD applications. Multiple different configurations at varying levels of complexities and flow regimes are explored to exercise the capabilities of the mesh. Finally, some concluding thoughts and directions for future efforts are discussed in Chapter 6.

Chapter 2

Overset Grid Generation Methods

2.1 Review of Differential Geometry

A brief overview of some fundamentals from differential geometry is presented here to facilitate discussions in later sections. In the context of grid generation, curves, surfaces, and volumes are considered. This is because hyperbolic grid generation produces a grid from initial data that is one dimension lower than the final grid (i.e. a surface is produced from an initial curve, and a volume mesh from an initial surface). A curve can be described by either a two or three-dimensional position vector $\vec{r}$ as a function of a single parameter ξ.

$$\vec{r} = \vec{r}(\xi)\,, \qquad \xi_0 \leq \xi \leq \xi_1 \tag{2.1}$$

A surface maps a two-dimensional parameterization (ξ, η) to a three-dimensional vector $\vec{r}(\xi, \eta)$, where,

$$\vec{r} = \vec{r}(\xi, \eta) : \mathbb{R}^2 \to \mathbb{R}^3 \tag{2.2}$$

Finally, volume maps a three-dimensional parameterization (ξ, η, ζ) to a three-dimensional vector.

$$\vec{r} = \vec{r}(\xi, \eta, \zeta) : \mathbb{R}^3 \to \mathbb{R}^3 \tag{2.3}$$

In this case, ξ, η, and ζ are the curvilinear coordinates. Similarly, it can be represented by $\vec{\xi} = \langle \xi, \eta, \zeta \rangle$. For the purposes of the following discussion, it is assumed that the parameterizations are smooth and invertible/non-degenerate.

2.1.1 Theory of Curves

The arc-length Δs for a curve between two points $\vec{r}(\xi^0)$ and $\vec{r}(\xi^0 + \Delta\xi)$ can be approximated by,

$$\Delta s = |\vec{r}(\xi^0 + \Delta\xi) - \vec{r}(\xi^0)| = \left| \frac{d\vec{r}}{d\xi}\Delta\xi + O(\Delta\xi^2) \right| \approx \left| \frac{d\vec{r}}{d\xi} \right| \Delta\xi \tag{2.4}$$

where $d\vec{r}/d\xi$ is the tangent vector. As $\Delta\xi \to 0$, we obtain the differential arc-length,

$$ds = \left| \frac{d\vec{r}}{d\xi} \right| d\xi = \sqrt{\frac{d\vec{r}}{d\xi} \cdot \frac{d\vec{r}}{d\xi}}\, d\xi \tag{2.5}$$

and thus, the total arc-length is given by

$$s(\xi) = \int_{\xi^0}^{\xi} ds = \int_{\xi^0}^{\xi} \sqrt{\vec{r}_\xi \cdot \vec{r}_\xi} d\xi, \qquad \xi > \xi^0 \tag{2.6}$$

The unit tangent $\vec{t}$ can be shown as differentiating $\vec{r}$ to with respect to its arc-length,

$$\vec{t} = \frac{\vec{r}_\xi}{|\vec{r}_\xi|} = \frac{d\vec{r}/d\xi}{ds/d\xi} = \frac{d\vec{r}}{ds} \tag{2.7}$$

The dot product of $\vec{t}$ with itself becomes $\vec{t} \cdot \vec{t} = 1$. Differentiation of this with respect to s results in the following,

$$\vec{t} \cdot \frac{d\vec{t}}{ds} = 0 \tag{2.8}$$

which shows that $d\vec{t}/ds$ is orthogonal to $\vec{t}$. The normalized vector is known as the unit normal vector $\vec{n}$.

$$\vec{n} = \frac{d\vec{t}/ds}{|d\vec{t}/ds|} \tag{2.9}$$

The curvature κ is given by the magnitude,

$$\kappa = |d\vec{t}/ds| = \frac{|\vec{r}_\xi \times \vec{r}_{\xi\xi}|}{|\vec{r}_\xi|^3} \tag{2.10}$$

The notion of a *natural* or *intrinsic* property is considered here, where quantities are independent of, or invariant to, the orientation and parameterization, but only dependent on the shape of the curve itself. In this case, the arc-length s and the curvature κ are such properties[1] of the curve. These properties can determine the shape of the curve uniquely, although it does not determine its orientation in space. The arc-length can be shown to have this invariance by considering Equation (2.6). Assuming a smooth mapping $\phi : \widetilde{\xi} \to \xi$ is introduced, the integration in Equation (2.6) can be expressed with $\widetilde{\xi}$ using a change in dependent variables.

$$s(\widetilde{\xi}) = \int_{\widetilde{\xi}^0}^{\widetilde{\xi}} \sqrt{\vec{r}_{\widetilde{\xi}} \cdot \vec{r}_{\widetilde{\xi}}} d\widetilde{\xi} \tag{2.11}$$

The change in variables still results in the same arc-length computation due to the reverse chain rule. Likewise, the curve can be rotated by a rotation matrix R, where $\vec{\tilde{r}} = R\vec{r}$. Thus,

$$\begin{aligned} \int_{\xi^0}^{\xi} \sqrt{\vec{\tilde{r}}_\xi \cdot \vec{\tilde{r}}_\xi} d\xi &= \int_{\xi^0}^{\xi} \sqrt{(R\vec{r}_\xi) \cdot (R\vec{r}_\xi)} d\xi \\ &= \int_{\xi^0}^{\xi} \sqrt{(R\vec{r}_\xi)^T (R\vec{r}_\xi)} d\xi \\ &= \int_{\xi^0}^{\xi} \sqrt{\vec{r}_\xi^T R^T R \vec{r}_\xi} d\xi \\ &= \int_{\xi^0}^{\xi} \sqrt{\vec{r}_\xi^T \vec{r}_\xi} d\xi \end{aligned} \tag{2.12}$$

[1] Torsion is also another such property but is not discussed here for the sake of brevity.

and hence the arc-length is also invariant to the orientation. Similar properties can be shown for the curvature as well. A reparameterization of the curvature results in the following,

$$\begin{aligned}
\kappa\left(\widetilde{\xi}\right) &= \frac{|\vec{r}_{\widetilde{\xi}} \times \vec{r}_{\widetilde{\xi}\widetilde{\xi}}|}{|\vec{r}_{\widetilde{\xi}}|^3} \\
&= \frac{\left|\frac{1}{d\widetilde{\xi}/d\xi}\vec{r}_\xi \times \frac{1}{d\widetilde{\xi}/d\xi}\left(\frac{1}{d\widetilde{\xi}/d\xi}\vec{r}_{\xi\xi} - \frac{d^2\widetilde{\xi}/d\xi^2}{(d\widetilde{\xi}/d\xi)^2}\vec{r}_\xi\right)\right|}{\left|\frac{1}{d\widetilde{\xi}/d\xi}\vec{r}_\xi\right|^3} \\
&= \frac{\left|\frac{1}{d\widetilde{\xi}/d\xi}\vec{r}_\xi \times \left(\frac{1}{d\widetilde{\xi}/d\xi}\right)^2\vec{r}_{\xi\xi}\right|}{\left(\frac{1}{d\widetilde{\xi}/d\xi}\right)^3|\vec{r}_\xi|^3} \\
&= \frac{\left(\frac{1}{d\widetilde{\xi}/d\xi}\right)^3|\vec{r}_\xi \times \vec{r}_{\xi\xi}|}{\left(\frac{1}{d\widetilde{\xi}/d\xi}\right)^3|\vec{r}_\xi|^3} \\
&= \frac{|\vec{r}_\xi \times \vec{r}_{\xi\xi}|}{|\vec{r}_\xi|^3}
\end{aligned} \tag{2.13}$$

Likewise, the curvature is invariant to rotation due to the magnitude operators in the numerator and denominator. In addition to intrinsic properties, it is also noted that the unit tangent vector $\vec{t}$ and unit normal vector $\vec{n}$ are invariant to the parameterization due to the invariance of the arc-length.

2.1.2 Jacobi Matrix and Metric Tensor

The Jacobi matrix J can be formed by taking the Jacobian of the position vector with respect to its curvilinear parameterizations. This is given by,

$$J = \left\{\frac{\partial r_i}{\partial \xi_j}\right\}, \qquad i = 1, .., n, \quad j = 1, ..., m \tag{2.14}$$

where n and m are the dimensions of the position vector and curvilinear parameterizations, respectively. Providing position and curvilinear coordinates have the same dimensionality, the Jacobian of transformation $\mathbf{J}$ can be found from taking the determinant of the Jacobi matrix,

$$\mathbf{J} = \det\left\{\frac{\partial r_i}{\partial \xi_j}\right\}, \qquad i = 1, .., n, \quad j = 1, ..., m \tag{2.15}$$

The metric tensor g_{ij} can also be found by multiplying the transpose of the Jacobi matrix with itself. Each element of the metric tensor becomes the dot product of the various coordinate tangent vectors

$$g_{ij} = J^T J = \frac{\partial \vec{r}}{\partial \xi_i} \cdot \frac{\partial \vec{r}}{\partial \xi_j} \tag{2.16}$$

2.1.3 Theory of Surfaces

As indicated earlier, a surface with a two-dimensional parameterization (ξ,η) is indicated by a vector function $\vec{r} = \vec{r}(\xi, \eta)$. The tangent plane at a particular point $\vec{r}(\xi^0, \eta^0)$ is formed by the tangent vectors

$\vec{r}_\xi\left(\xi^0,\eta^0\right)$ and $\vec{r}_\eta\left(\xi^0,\eta^0\right)$. The local unit normal can be found by taking the cross-product of the tangent vectors, resulting in

$$\vec{n} = \frac{\vec{r}_\xi \times \vec{r}_\eta}{|\vec{r}_\xi \times \vec{r}_\eta|} \tag{2.17}$$

The local unit normal of a surface is also parameterization independent. A smooth mapping $\phi : (\widetilde{\xi}, \widetilde{\eta}) \rightarrow (\xi, \eta)$ is considered, where the Jacobian is given by,

$$J(\phi) = \begin{bmatrix} \frac{\partial \xi}{\partial \widetilde{\xi}} & \frac{\partial \xi}{\partial \widetilde{\eta}} \\ \frac{\partial \eta}{\partial \widetilde{\xi}} & \frac{\partial \eta}{\partial \widetilde{\eta}} \end{bmatrix} \tag{2.18}$$

We also note that the vector tangents in the new parameterizations can be obtained using the chain rule, which is given by,

$$\begin{aligned} \vec{r}_{\widetilde{\xi}} &= \frac{\partial \xi}{\partial \widetilde{\xi}}\vec{r}_\xi + \frac{\partial \eta}{\partial \widetilde{\xi}}\vec{r}_\eta \\ \vec{r}_{\widetilde{\eta}} &= \frac{\partial \xi}{\partial \widetilde{\eta}}\vec{r}_\xi + \frac{\partial \eta}{\partial \widetilde{\eta}}\vec{r}_\eta \end{aligned} \tag{2.19}$$

The unit normal described by the new parameterization is given by the following,

$$\begin{aligned} \vec{n}\left(\widetilde{\xi}, \widetilde{\eta}\right) &= \frac{\vec{r}_{\widetilde{\xi}} \times \vec{r}_{\widetilde{\eta}}}{|\vec{r}_{\widetilde{\xi}} \times \vec{r}_{\widetilde{\eta}}|} \\ &= \frac{\left(\frac{\partial \xi}{\partial \widetilde{\xi}}\frac{\partial \eta}{\partial \widetilde{\eta}} - \frac{\partial \xi}{\partial \widetilde{\eta}}\frac{\partial \eta}{\partial \widetilde{\xi}}\right)(\vec{r}_\xi \times \vec{r}_\eta)}{\left(\frac{\partial \xi}{\partial \widetilde{\xi}}\frac{\partial \eta}{\partial \widetilde{\eta}} - \frac{\partial \xi}{\partial \widetilde{\eta}}\frac{\partial \eta}{\partial \widetilde{\xi}}\right)|\vec{r}_\xi \times \vec{r}_\eta|} \\ &= \frac{\vec{r}_\xi \times \vec{r}_\eta}{|\vec{r}_\xi \times \vec{r}_\eta|} = \vec{n}\left(\xi, \eta\right) \end{aligned} \tag{2.20}$$

2.2 Hyperbolic Grid Generation

Three different formulations exist for hyperbolic grid generation, which have been presented earlier in the previous chapter in Equations 1.8, 1.10, and 1.11. The two-dimensional field mesh generation system [50] in $\mathbb{R}^2$ is constructed from enforcing orthogonality through the off-diagonal term in the two-dimensional metric tensor, and enforcing a non-zero Jacobian of transformation to ensure a mapping exists. This results in the following system,

$$\begin{aligned} x_\xi x_\eta + y_\xi y_\eta &= 0 \\ x_\xi y_\eta - y_\xi x_\eta &= J^{-1} = \Delta A \end{aligned} \tag{2.21}$$

The enforcement of the orthogonality condition turns the Jacobian of transformation into a local infinitesimal area ΔA. A more generalized approach to generate a hyperbolic surface mesh in $\mathbb{R}^3$ was presented in [57]. This is constructed by enforcing orthogonality in the off-diagonal term in the metric tensor in the first equation, and tangency between the reference surface normal and the generated surface normal. The second equation also functions to enforce a local cell area similar to the second equation in the

two-dimensional system. A third equation is added to close the system by enforcing orthogonality the reference surface normal and the marching-direction tangent vector $\vec{r}_\eta$. This is given by

$$\begin{aligned} \vec{r}_\xi \cdot \vec{r}_\eta &= 0 \\ \vec{n} \cdot (\vec{r}_\xi \times \vec{r}_\eta) &= \Delta S \\ \vec{n} \cdot \vec{r}_\eta &= 0 \end{aligned} \tag{2.22}$$

where $\vec{n}$ is the unit normal of the reference surface, and ΔS is the local infinitesimal surface area, although in practice this becomes the local cell area after the equations are numerically discretized. Finally, the third system enables generation of volume meshes in $\mathbb{R}^3$ [53, 57]. The approach enforces orthogonality of the tangents with the marching direction vector, which come from two of the off-diagonal terms in the volume metric tensor. A third equation attempts to enforce an invertible mapping through local surface normal of the "marching front". This results in the following system,

$$\begin{aligned} \vec{r}_\xi \cdot \vec{r}_\zeta &= 0 \\ \vec{r}_\eta \cdot \vec{r}_\zeta &= 0 \\ \vec{r}_\zeta \cdot (\vec{r}_\xi \times \vec{r}_\eta) &= \Delta V \end{aligned} \tag{2.23}$$

where ΔV is the local infinitesimal volume (or cell volume when the equations are discretized).

2.2.1 Hyperbolicity of the Equations

To demonstrate that the above mesh generation systems are indeed hyperbolic, a perturbation analysis is performed around an initial state $\vec{r}^0$. The following sections detail this analysis for all three equation systems. The resulting system can be analyzed for hyperbolicity by examining the linearized matrix operating on the partial derivatives of the space-analogous variables (ξ for field/surface, ξ and η for volume). Existence of only real eigenvalues indicates hyperbolicity.

2.2.1.1 Two-Dimensional Field Meshes

The perturbation analysis of Equation (2.21) results in the form,

$$A(\vec{r}^0)\delta\vec{r}_\xi + B(\vec{r}^0)\delta\vec{r}_\eta = \vec{f} \tag{2.24}$$

which is given by,

$$\begin{bmatrix} x_\eta^0 & y_\eta^0 \\ y_\eta^0 & -x_\eta^0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}_\xi + \begin{bmatrix} x_\xi^0 & y_\xi^0 \\ -y_\xi^0 & x_\xi^0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}_\eta = \begin{bmatrix} 0 \\ (J^{-1})^0 + \Delta J^{-1} \end{bmatrix} \tag{2.25}$$

Hyperbolicity can be determined if the matrix $B^{-1}A$ has real eigenvalues. This matrix is given by,

$$B^{-1}A = \begin{bmatrix} x_\xi^0 & y_\xi^0 \\ -y_\xi^0 & x_\xi^0 \end{bmatrix}^{-1} \begin{bmatrix} x_\eta^0 & y_\eta^0 \\ y_\eta^0 & -x_\eta^0 \end{bmatrix} = \frac{1}{x_\xi^2 + y_\xi^2} \begin{bmatrix} x_\xi x_\eta - y_\xi y_\eta & x_\xi y_\eta + x_\eta y_\xi \\ x_\xi y_\eta + x_\eta y_\xi & -x_\xi x_\eta + y_\xi y_\eta \end{bmatrix} \tag{2.26}$$

The matrix has two distinct eigenvalues given by,

$$\lambda_{1,2} = \pm\sqrt{\frac{x_\eta^2 + y_\eta^2}{x_\xi^2 + y_\xi^2}} \tag{2.27}$$

which shows that the system is indeed hyperbolic.

2.2.1.2 Surface Meshes

The perturbation analysis of Equation (2.22) also results in a similar form to Equation (2.24). This is given by,

$$\begin{bmatrix} x_\eta & y_\eta & z_\eta \\ n_3 y_\eta - n_2 z_\eta & n_1 z_\eta - n_3 x_\eta & n_2 x_\eta - n_1 y_\eta \\ 0 & 0 & 0 \end{bmatrix} \delta\vec{r}_\xi + \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ -n_3 y_\xi + n_2 z_\xi & -n_1 z_\xi + n_3 x_\xi & -n_2 x_\xi + n_1 y_\xi \\ n_1 & n_2 & n_3 \end{bmatrix} \delta\vec{r}_\eta = \begin{bmatrix} 0 \\ \Delta S + \Delta S^0 \\ 0 \end{bmatrix} \tag{2.28}$$

Hyperbolicity can be determined if the matrix $B^{-1}A$ has real eigenvalues. This matrix is given by,

$$B^{-1}A = \frac{1}{|\vec{r}_\xi|^2} \begin{bmatrix} x_\xi & n_2 z_\xi - n_3 y_\xi & n_1(\vec{r}_\xi \cdot \vec{r}_\xi) \\ y_\xi & -n_1 z_\xi + n_3 x_\xi & n_2(\vec{r}_\xi \cdot \vec{r}_\xi) \\ z_\xi & n_1 y_\xi - n_2 x_\xi & n_3(\vec{r}_\xi \cdot \vec{r}_\xi) \end{bmatrix} \begin{bmatrix} x_\eta & y_\eta & z_\eta \\ n_3 y_\eta - n_2 z_\eta & n_1 z_\eta - n_3 x_\eta & n_2 x_\eta - n_1 y_\eta \\ 0 & 0 & 0 \end{bmatrix} \tag{2.29}$$

The three eigenvalues are given by,

$$\lambda_{1,2,3} = 0, \pm\sqrt{\frac{\vec{r}_\eta \cdot \vec{r}_\eta}{\vec{r}_\xi \cdot \vec{r}_\xi}} \tag{2.30}$$

which indicates that the surface mesh generation system is hyperbolic due to the three real eigenvalues.

2.2.1.3 Volume Meshes

The perturbation analysis of the volume mesh generation system in Equation (2.23) results in the form given by,

$$A(\vec{r}^0)\delta\vec{r}_\xi + B(\vec{r}^0)\delta\vec{r}_\eta + C(\vec{r}^0)\delta\vec{r}_\zeta = \vec{f}\left(\vec{r}^0\right) \tag{2.31}$$

where the matrices are given by,

$$
\begin{aligned}
A(\vec{r}) &= \begin{bmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ (y_\eta z_\zeta - y_\zeta z_\eta) & (x_\zeta z_\eta - x_\eta z_\zeta) & (x_\eta y_\zeta - x_\zeta y_\eta) \end{bmatrix} \\
B(\vec{r}) &= \begin{bmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z_\zeta \\ (y_\zeta z_\xi - y_\xi z_\zeta) & (x_\xi z_\zeta - x_\zeta z_\xi) & (x_\zeta y_\xi - x_\xi y_\zeta) \end{bmatrix} \\
C(\vec{r}) &= \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ (y_\xi z_\eta - y_\eta z_\xi) & (x_\eta z_\xi - x_\xi z_\eta) & (x_\xi y_\eta - x_\eta y_\xi) \end{bmatrix}
\end{aligned}
\tag{2.32}
$$

To determine whether the system is hyperbolic, the eigenvalues of the matrices $C^{-1}A$ and $C^{-1}B$ must be evaluated to be real. These matrices are given by,

$$
\begin{aligned}
C^{-1}A &= \frac{1}{|\vec{r}_\xi \times \vec{r}_\eta|^2} \begin{bmatrix} x_\xi(\vec{r}_\eta \cdot \vec{r}_\eta) - x_\eta(\vec{r}_\xi \cdot \vec{r}_\eta) & x_\eta(\vec{r}_\xi \cdot \vec{r}_\xi) - x_\xi(\vec{r}_\xi \cdot \vec{r}_\eta) & y_\xi z_\eta - y_\eta z_\xi \\ y_\xi(\vec{r}_\eta \cdot \vec{r}_\eta) - y_\eta(\vec{r}_\xi \cdot \vec{r}_\eta) & y_\eta(\vec{r}_\xi \cdot \vec{r}_\xi) - y_\xi(\vec{r}_\xi \cdot \vec{r}_\eta) & -x_\xi z_\eta + x_\eta z_\xi \\ z_\xi(\vec{r}_\eta \cdot \vec{r}_\eta) - z_\eta(\vec{r}_\xi \cdot \vec{r}_\eta) & z_\eta(\vec{r}_\xi \cdot \vec{r}_\xi) - z_\xi(\vec{r}_\xi \cdot \vec{r}_\eta) & x_\xi y_\eta + x_\eta y_\xi \end{bmatrix} A \\
C^{-1}B &= \frac{1}{|\vec{r}_\xi \times \vec{r}_\eta|^2} \begin{bmatrix} x_\xi(\vec{r}_\eta \cdot \vec{r}_\eta) - x_\eta(\vec{r}_\xi \cdot \vec{r}_\eta) & x_\eta(\vec{r}_\xi \cdot \vec{r}_\xi) - x_\xi(\vec{r}_\xi \cdot \vec{r}_\eta) & y_\xi z_\eta - y_\eta z_\xi \\ y_\xi(\vec{r}_\eta \cdot \vec{r}_\eta) - y_\eta(\vec{r}_\xi \cdot \vec{r}_\eta) & y_\eta(\vec{r}_\xi \cdot \vec{r}_\xi) - y_\xi(\vec{r}_\xi \cdot \vec{r}_\eta) & -x_\xi z_\eta + x_\eta z_\xi \\ z_\xi(\vec{r}_\eta \cdot \vec{r}_\eta) - z_\eta(\vec{r}_\xi \cdot \vec{r}_\eta) & z_\eta(\vec{r}_\xi \cdot \vec{r}_\xi) - z_\xi(\vec{r}_\xi \cdot \vec{r}_\eta) & x_\xi y_\eta + x_\eta y_\xi \end{bmatrix} B
\end{aligned}
\tag{2.33}
$$

The eigenvalues of the matrix $C^{-1}A$ are given by

$$
\lambda_{C^{-1}A,1,2,3} = 0, \pm \frac{|\vec{r}_\eta \times \vec{r}_\zeta|}{|\vec{r}_\xi \times \vec{r}_\eta|} \tag{2.34}
$$

while the eigenvalues of the matrix $C^{-1}B$ are given by,

$$
\lambda_{C^{-1}B,1,2,3} = 0, \pm \frac{|\vec{r}_\xi \times \vec{r}_\zeta|}{|\vec{r}_\xi \times \vec{r}_\eta|} \tag{2.35}
$$

The system is indeed hyperbolic since all the eigenvalues for both matrices are real.

2.2.2 Numerical Discretization

The typical approach to solve the nonlinear PDE systems is to numerically solve the *linearized* version of the equations, which have been presented in the previous section. Techniques from solving hyperbolic conservation laws using central differencing schemes can be leveraged to discretize the system. The "spatial"-like terms (ξ in 2D and surface meshes, ξ and η in volume meshes) can be discretized similarly to the spatial derivatives in conservation laws. The derivatives in the spatial directions are discretized directly with second-order central finite differences. For example,

$$
x_\xi \approx \frac{x_{j+1} - x_{j-1}}{2} \tag{2.36}
$$

Similar to conservation laws, dissipation terms are commonly added to the discretization, which serve to dissipate sharp gradients and smooth the grid. This can occur when grid lines start to converge towards each other. Chan and Steger [57] introduced the following formulation,

$$\epsilon_\xi \left(\Delta\nabla\right)_\xi \vec{r} = \epsilon_\xi \left(\vec{r}_{j+1} - 2\vec{r}_j + \vec{r}_{j-1}\right) \tag{2.37}$$

where ϵ_{xi} is a variable dissipation coefficient. This coefficient is given by,

$$\epsilon_\xi = \epsilon_e R_\xi N_\xi \tag{2.38}$$

where ϵ_e is a user-specified constant of $O(1)$. The quantity N_ξ is the spectral-radius of the coefficient matrix of the original "spatial" derivatives. In field/surface meshes, the matrix is $B^{-1}A$, while volume meshes have $C^{-1}A$ (and correspondingly $C^{-1}B$ for the dissipation in η). The quantity R_ξ is a variable dissipation coefficient given by,

$$R_\xi = d^\xi_{j,k,l} \alpha^\xi_{j,k,l} \tag{2.39}$$

where $d^\xi_{j,k,l}$ is given by

$$d^\xi_{j,k,l} = \frac{|\vec{r}_{j+1,k,l-1} - \vec{r}_{j,k,l-1}| + |\vec{r}_{j,k,l-1} - \vec{r}_{j-1,k,l-1}|}{|\vec{r}_{j+1,k,l} - \vec{r}_{j,k,l}| + |\vec{r}_{j,k,l} - \vec{r}_{j-1,k,l}|} \tag{2.40}$$

and $\alpha^\xi_{j,k,l}$ is function of the local half angle $a_{j,k,l}$, where

$$\alpha^\xi = \begin{cases} 1/(1 - \cos^2 a_{j,k,l}) & \text{if } 0 \le a_{j,k,l} \le \pi/2 \\ 1 & \text{if } \pi/2 < a_{j,k,l} \le \pi \end{cases} \tag{2.41}$$

The angle $a_{j,k,l}$ is given by

$$\cos a_{j,k,l} = \vec{n}_{j,k,l} \cdot (\vec{r}_{j+1,k,l} - \vec{r}_{j,k,l}) = \vec{n}_{j,k,l} \cdot (\vec{r}_{j,k,l} - \vec{r}_{j-1,k,l}) \tag{2.42}$$

where $\vec{n}$ is the local unit normal.

The approach to generate the meshes and advance them in the "time"-like direction (η for 2D/surface, ζ for volume) is to take the linearized formulations and supply a perturbation $\delta\vec{r} = \vec{r}_{l+1} - \vec{r}_l$. This results in a "Delta"-like formulation similar to the implicit formulation for hyperbolic conservation laws obtained by Beam and Warming [84]. In the case of the volume mesh generation system, the system linearized about $\vec{r}_l$ is given by,

$$\left[I + C^{-1}(\vec{r}_l)A(\vec{r}_l)\partial_\xi + C^{-1}(\vec{r}_l)B(\vec{r}_l)\partial_\eta - \epsilon_\xi(\Delta\nabla)_\xi - \epsilon_\eta(\Delta\nabla)_\eta\right](\vec{r}_{l+1} - \vec{r}_l) = C^{-1}(\vec{r}_l)\vec{f}(\vec{r}_l) \tag{2.43}$$

where $\partial_{\xi,\eta}$ are the central differencing operators. A further simplification can be made by performing an alternating-direction implicit (ADI) splitting, which is given by,

$$\left[I + C^{-1}(\vec{r}_l)A(\vec{r}_l)\partial_\xi - \epsilon_\xi(\Delta\nabla)_\xi\right]\left[I + C^{-1}(\vec{r}_l)B(\vec{r}_l)\partial_\eta - \epsilon_\eta(\Delta\nabla)_\eta\right](\vec{r}_{l+1} - \vec{r}_l) = C^{-1}(\vec{r}_l)\vec{f}(\vec{r}_l) \tag{2.44}$$

Each bracket is now a block-tridiagonal system that is less costly to solve.

2.2.3 Boundary Conditions

Chan and Steger [57] list several boundary conditions available for hyperbolic mesh generation. Typical boundary conditions include periodic and symmetry boundary conditions. A less conventional boundary condition is the constant Cartesian plane. This boundary condition enforces coincidence with a constant x, y, or z plane, allowing the boundary to only move along this specified plane.

A more general-purpose boundary condition when the other boundaries cannot be used is the "splay" boundary. This boundary condition is based on an interior extrapolation, which, at the $j = 1$ boundary, is given by the following,

$$\Delta\vec{r}_{j=1} = \Delta\vec{r}_{j=2} + \varepsilon_{splay}\left[\Delta\vec{r}_{j=2} - \Delta\vec{r}_{j=3}\right] \tag{2.45}$$

The variable ε_{splay} is a user-specified quantity of the range $0 \leq \varepsilon_{splay} \leq 1$. The extreme quantities recover the zeroth and first-order extrapolations, respectively. As the value is increased beyond the zeroth-order extrapolation, the boundaries have a tendency to splay outward away from the interior of the mesh. This can be helpful in overset mesh generation, since it facilitates overlap between grids. However, the boundary must be empirically set by the user, and there is no general formulation to describe the relationship between the quantity of ε_{splay} and the amount of splay. Figure 2.1 shows an example of a volume mesh generated from an initial surface (blue) with splay boundaries (magenta, salmon, olive, and yellow). Note how the boundary surfaces bend away from the center of the mesh.

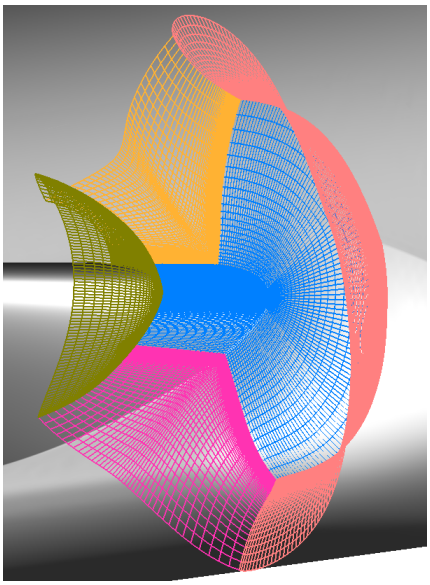


Figure 2.1: Volume mesh with initial surface (blue) generated with splay boundaries (non-blue surfaces).

Grid generation using hyperbolic schemes also require an initial condition to be provided, since it is an initial value problem. In the case of the field or surface mesh system, this requires an initial curve. The surface mesh system also requires a separate reference surface on which the grid is generated, and the initial curve must also lie on the reference surface. The volume mesh system, on the other hand, requires an initial surface. This data is generally user-provided.

2.3 Review of Manual Overset Mesh Generation

A review of the manual approach for overset mesh generation is given here as a reference point to demonstrate the automation scheme presented in the subsequent section. In most situations, the entire process can be completed using the CGT script library tools. Following the approach outlined by Chan et al [26, 85], computing a flow solution from a CAD geometry with overset meshes generally consists of five main steps: (1) surface domain decomposition, (2) surface mesh generation, (3) volume mesh

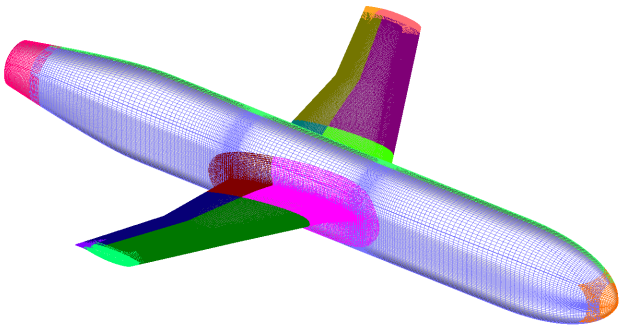


Figure 2.2: Manually-generated overset meshes of the Juncture Flow geometry [87].

generation, (4) domain connectivity computation, and (5) flow solver input generation . The surface domain decomposition step divides the wetted wall surfaces of a geometry into smaller regions that can be meshed. Typically, this involves extracting the geometry information from a CAD model, which can be done by CAD processing software (e.g. ANSA [86]). If a solid BRep CAD is available (STEP, IGES), the EGADS2SRF software [80] can be used to extract the original geometry information into the PLOT3D format, which can be used to generate meshes using the scripting libraries within CGT. In other cases, only a collection of panel surfaces or a surface triangulation is available. The geometry may require further processing, such as merging or breaking up various surfaces such that it is amenable to a structured mesh parameterization.

The surface meshing step usually involves further processing of the outputs generated from the surface decomposition step. This can involve generation of or redistribution of the surface parameterizations from the previous step. Seam meshes (e.g. collar meshes, leading and trailing edge meshes, etc.) are generated to connect the various surfaces together into a closed surface from an overset perspective. Users may also choose to construct meshes such that multiple different geometric features are represented within a single grid. Care must be taken to ensure that sufficient overlap is present in the surface mesh. In all cases, the requirement is at least $2 \times n_f$ overlap, where n_f is the number of fringe (interpolation) point layers in order to support the differencing stencil in each dimension needed for the desired numerical scheme. This will minimize the issue of orphan points, or fringe points without viable donor cells in neighboring grids, in the final volume mesh system. Figure 2.2 shows an example of a manually-generated overset surface mesh. Note that the user is able to minimize the number of meshes needed to represent each feature in the geometry. For example, only four meshes are needed on the wing to represent important features such as the leading and trailing edges, wing crank, and wing tip. The fuselage also only contains four meshes with a single main indigo mesh representing most of the fuselage, while smaller orange and red cap meshes (and an additional tail mesh obscured by the perspective) cover the axisymmetric ends. Each wing-fuselage junction is covered by two collar meshes (magenta and brown on the right, lime green and dark green on the left) to connect the two features.

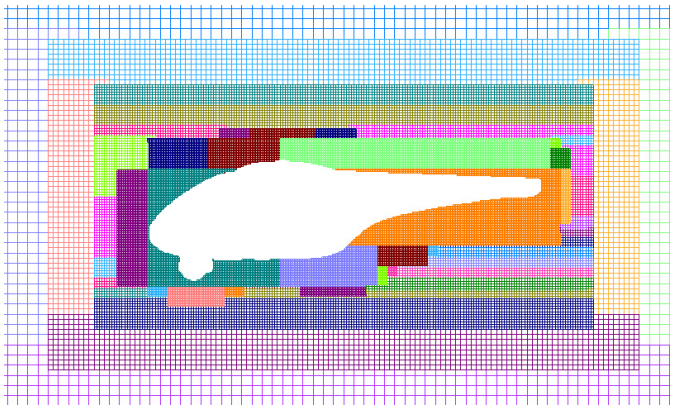


Figure 2.3: Hole-cutting performed using Overflow DCF and XRAYS on the off-body Cartesian meshes.

Once the surface meshes are finished, the volume mesh can be generated. There are two types of volume meshes: (1) near-body meshes, and (2) off-body meshes. The near-body meshes are body-conforming and surround the wetted wall surfaces in the domain. These meshes are usually generated using hyperbolic methods (see Section 2.2). Depending on the turbulent physics desired to be captured, a fine spacing based on some proportion of the $y+$ is applied near the wall. The normal grid spacing stretches as the meshes are grown away from the wall to avoid overly-fine meshes. Splay boundaries may also need to be adjusted to ensure complete coverage of the flow domain. In the surrounding region to the far field, a background Cartesian mesh can be generated in which the near-body meshes are embedded. Refinement of the region surrounding the near-body meshes is typically applied to ensure proper resolution of gradients and communication between grids. Depending on the geometries, this may require several Cartesian grids embedded within each other. In external flow applications, the off-body grid generation step in Overflow [65] can provide a computationally efficient approach.

Once the entire flow domain is meshed, the domain connectivity steps can be performed, which include hole-cutting and donor-stencil search. Hole-cutting must be performed to ensure that grid points not within the flow domain are removed or *blanked*, while donor-stencil search is performed to find interpolation data for fringe points to effectively form a complete flow domain. This can be performed automatically using PEGASUS [67], or manually through the XRAY [28] and Domain Connectivity Function (DCF) [88] routines from within Overflow. Figure 2.3 shows the hole-cutting performed on the off-body mesh for a rotorcraft fuselage near-body grid. With the completion of the mesh generation process, the user can then create inputs for controlling the flow solver and for performing load integration. At this point, the pre-processing is complete and the user can run the flow computation.

2.4 Automatic Structured Overset Mesh Generation

The following section briefly details the process for automatic structured overset mesh generation. It consists of two software tools called *EGADS2SRF* [80] and the Pre-processor for Overset Grid Simulations

(POGS) [83], which are currently under active development within the Chimera Grid Tools software suite. The two automation tools are collectively known as EPOGS. While the end goal is to have complete automation of the process between having a CAD model to starting a flow computation, the current state of automation is evolving constantly as development is underway. In some of the work presented in this dissertation, a *hybrid* automatic approach is used. Here, the surface and volume mesh generation steps are automatically performed by EPOGS, while the domain connectivity step is manually performed. This is due to the progress of development in EPOGS at the time of writing.

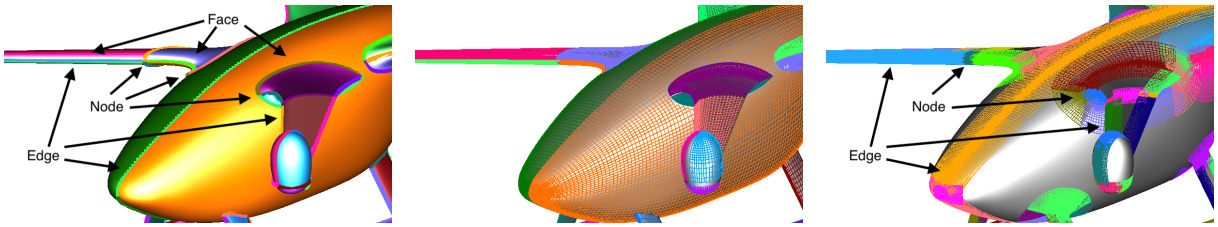
2.4.1 Surface Domain Decomposition and Mesh Generation

Similar to the manual approach, EGADS2SRF is used to perform surface domain decomposition, which extracts all the relevant geometric and topological features. This step involves decomposing a BRep geometry into discrete components of face surface meshes, edge curves, and node points that can then be later used by POGS. Here, EGADS2SRF queries information from the CAD via routines from the Engineering Sketch Pad (ESP) library [81]. The user simply has to supply several meshing parameters that prescribe limits on the local resolution, which is detailed in Table 2.1. The software factors this information into account when discretizing the various geometric elements. This step requires only a few minutes to complete for a geometry with moderate complexity.

Proceeding into the surface mesh generation step, POGS generates surface meshes using input files created by EGADS2SRF. The edge curves and node points are used to generate edge and node surface meshes that bound and trim the various face meshes, resulting in an overset mesh representation of the geometry surface. The choice of local spacing and point distribution for the newly generated edge and node

Table 2.1: Mesh parameter inputs for EPOGS.

Mesh Parameter Input	Description
Max edge length (Δs_{max})	Sets max edge length of cells on the surface mesh and the outer normal spacing of the volume mesh. Also sets the edge length of off-body meshes surrounding the near-body meshes.
Max turning angle ($\Delta\theta_{max}$)	Sets max dihedral angles between edges of adjacent cells.
Minimum point count (np_{min})	Sets minimum size of each dimension for a single grid block. Also controls number of points across a finite-thickness trailing edge.
Max stretching ratio (SR_{max})	Sets max allowed stretching between adjacent cell edges.
Number of fringe points (n_{fringe})	Sets minimum required overlap to ensure sufficient connectivity between grids.
Initial wall spacing (Δs_{wall})	Initial spacing off the surface mesh during volume mesh generation in step (3).



(a) (b) (c)

Figure 2.4: Surface mesh decomposition from BRep geometry. (a) BRep geometry entities. (b) Meshes generated from BRep faces. (c) Meshes generated from BRep nodes and edges.

surface meshes are guided by the parameters defined in the input file. Compared to manually generated meshes, there are generally more near-body meshes when generated automatically due to the fact that a surface mesh is generated for nearly every node, edge, and face. Domain connectivity on the surface mesh, including surface hole-cutting, is also automatically performed. The purpose of surface hole-cutting in this step is to trim surface mesh points that do not sit on an external or wetted surface of a geometry. A retraction of face meshes is also performed so that the overlap between all surface meshes are reduced such that only sufficient overlap is maintained to ensure proper donor stencil quality for interpolated boundaries. This also assists to reduce overhead of face mesh points that may not participate in the flow computation. Figure 2.4 demonstrates how EGADS2SRF decomposes the geometry based on the geometric entities, followed by the surface meshes that are then generated using POGS.

2.4.2 Volume Mesh Generation

In the subsequent step, POGS creates a set of near-body volume meshes using the generated surface meshes as initial conditions. The software automatically determines various meshing parameters based on the input file and/or the initial surface. Chan et al [83] provides greater detail in how these parameters are determined. Generally, splay boundaries are typically applied unless the surface mesh boundary sits on a symmetry boundary or external boundary of the flow domain. The near-body volume meshes are generated using hyperbolic methods and grown until a specific distance is reached such that the normal spacing grows from the initial wall spacing Δs_{wall} to the outer maximum spacing Δs_{max}.

In external flow situations, POGS can also generate an off-body volume Cartesian mesh surrounding the geometry of interest. This grid will have a special refinement region surrounding the geometry with uniform spacing of Δs_{max}, and then it stretches away to the far-field to avoid significant clustering in the far-field regions where flow gradients are small. The refinement region also allows for ideal overlap with the outer layers of the near-body volume meshes.

2.4.3 Hole-Cutting and Domain Connectivity

POGS also has the capability to perform automatic domain connectivity and hole-cutting for the final volume mesh configuration [83]. The domain connectivity is a two-step approach. Hole-cuts are

first performed for the near-body mesh, which are based on propagating the surface mesh hole-cuts, or IBLANKS, to the other L levels generated in the volume mesh generation step. Intersections of grid rays in the marching direction are also tagged for blanking if they intersect the geometry itself. The hole-cut for the geometry is then performed on the off-body mesh. An initial domain-connectivity step is performed, allowing for identification of orphan points among the fringe points. Additional Cartesian boxes are added to the off-body mesh to provide coverage for the orphan points. The hole-cutting and connectivity steps are then repeated for these smaller Cartesian boxes, which should result in a complete mesh with sufficient overlap.

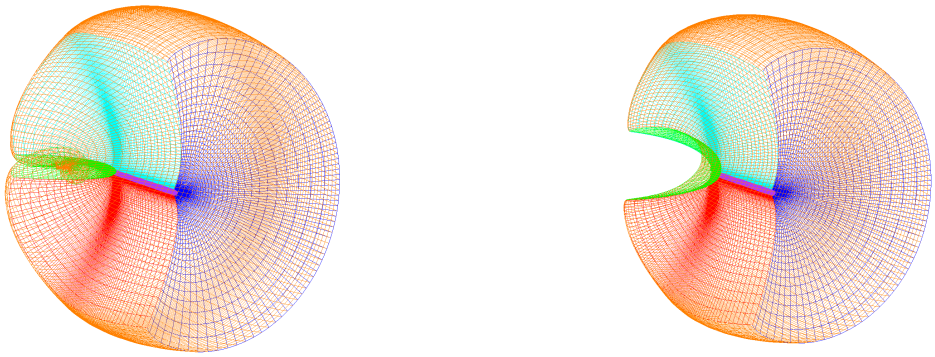
An alternative manual approach utilizing the Domain Connectivity Function (DCF) [88] in Overflow and X-Rays [28] is also possible. Here, POGS can assist with creating the X-Ray hole-cutters, as well as an initial set of hole-cutting instructions. The user must then pass these inputs along with the grid into Overflow to perform the hole-cutting and connectivity in DCF. The instructions must then be iterated until the desired connectivity quality is obtained. This is largely a "hybrid" approach, since it still requires user intervention to produce the desired output. In most cases in this dissertation, this is the main approach utilized for connectivity when using the automatic mesh generation tools.

2.4.4 Flow Solver Input Generation

POGS can also automate the input deck necessary to pass into the desired solver. These include input files for controlling solver options, run configurations, and boundary conditions. A separate input deck for configuring the force and moment computation setup are also automatically generated. Currently, however, these files will typically require minor user modification before they can be directly used.

2.4.5 Manual Repairs

At the time of writing, the grid configurations generated by EPOGS may require manual repairs. Essentially, the various mesh generation steps are still not perfectly automated, since they can be reliant on



(a) (b)

Figure 2.5: Volume mesh needing repair. (a) Mesh with negative cell volumes from grid boundary twisting. (b) Manually repaired mesh.

input parameters that are difficult to control (e.g. mesh splay boundaries, grid smoothing parameters, etc.) As such, some manual intervention may be required. This can include extending surface meshes such that there is sufficient overlap between all surfaces, or regenerating volume meshes to eliminate negative Jacobians or negative cell volumes. Occasionally, volume meshes may need to be generated with different parameters to ensure sufficient overlap between grids. In practice, only a small handful of meshes ever experience such problems for many tested configurations. Figure 2.5 shows an example of such meshes that require manual regeneration. A boundary containing poor mesh quality can be observed in the green bounding face in Figure 2.5a, resulting in the fixed mesh after the repairs are applied in Figure 2.5b. Currently, the success rate for automation with highly-quality grids is around $\approx 99 - 100\%$ for surface meshes, and $\approx 95 - 100\%$ for volume meshes [83].

Chapter 3

Overflow Flow Solver

Overflow [21] is a three-dimensional, implicit finite-difference/finite-volume flow solver that solves the compressible Navier-Stokes equations on overset grids. The flow solver also contains routines for performing multi-layered off-body Cartesian mesh generation, X-RAY hole-cutting and domain-connectivity through DCF. It can also handle relative motion and applications with multi-species and/or variable specific heats. It supports a variety of numerical discretizations for the inviscid terms, and various turbulence models are also available. The code is also amenable to parallel computation through MPI and OpenMP. It currently is a widely-used flow solver within the overset mesh community in government laboratories, industry, and in academia. This flow solver is used to explore the aerodynamic applications demonstrated in Chapter 5. Since the focus is not on the CFD solver itself, only a concise summary of the various flow solver options and numerical methods that are used in this work are given in this chapter.

3.1 Reynolds-Averaged Navier-Stokes

The Reynolds-Averaged Navier-Stokes (RANS) is the main formulation of the Navier-Stokes equations that is studied in this work. This is mainly due to the abundance of RANS solutions available for comparison and verification, and the fact that RANS is still the main workhorse for many CFD applications. Investigating the viability of the automatically-generated meshes with higher fidelity formulations, such as Large Eddy Simulation (LES) or hybrid RANS-LES, is the subject of future work.

Reynolds averaging for any quantity $A(\mathbf{x}, t)$ is given by the following,

$$A(\mathbf{x}, t) = \bar{A}(\mathbf{x}, t) + A'(\mathbf{x}, t) \tag{3.1}$$

where

$$\bar{A}(\mathbf{x}, t) = \frac{1}{T} \int_{-T/2}^{T/2} A(\mathbf{x}, t + \tau) d\tau \tag{3.2}$$

is the turbulent-averaged quantity, and A' is the turbulent fluctuation. However, in Overflow, the compressible Navier-Stokes equations are solved, and as such, Favre averaging is used instead. Favre averaging

is a density-weighted averaging, which is defined by the following,

$$A(\mathbf{x},t) = \widetilde{A}(\mathbf{x},t) + A''(\mathbf{x},t) \tag{3.3}$$

where

$$\widetilde{A}(\mathbf{x},t) = \frac{\overline{\rho A}}{\bar{\rho}} \tag{3.4}$$

and

$$\overline{\rho A''} = 0 \tag{3.5}$$

When applied to Equation (1.1) expressed in Cartesian coordinates, the averaging thus results in the following formulation,

$$\begin{aligned}
\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\widetilde{u}_i\right) &= 0 \\
\frac{\partial \bar{\rho}\widetilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{\rho}\widetilde{u}_i\widetilde{u}_j\right) + \frac{\partial \bar{p}}{\partial x_i} &= \frac{\partial \widetilde{\tau}_{ij}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(-\overline{\rho u_i'' u_j''}\right) \\
\frac{\partial \bar{\rho}\widetilde{E}}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{\rho}\widetilde{u}_j\widetilde{H}\right) &= \frac{\partial}{\partial x_j}\left(\widetilde{u}_i\widetilde{\tau}_{ij} + \overline{u_i''\tau_{ij}}\right) + \frac{\partial}{\partial x_j}\left(\frac{c_p\widetilde{\mu}}{Pr}\frac{\partial \widetilde{T}}{\partial x_j}\right) \\
&\quad - \frac{\partial}{\partial x_j}\left(\widetilde{u}\overline{\rho u_i'' u_j''} + c_p\overline{\rho u_j'' T''} + \frac{1}{2}\overline{\rho u_j'' u_i'' u_i''}\right)
\end{aligned} \tag{3.6}$$

The equation of state is given by,

$$\bar{p} = (\gamma - 1)\left[\bar{\rho}\widetilde{E} - \frac{1}{2}\bar{\rho}\widetilde{u}_i\widetilde{u}_i - \bar{\rho}k\right] \tag{3.7}$$

where $k = (1/2)u_i''u_i''$ is the turbulent kinetic energy. Four terms typically must be modeled, which are the

- Reynolds stress: $-\overline{\rho u_i'' u_j''}$
- Turbulent heat flux: $c_p\overline{\rho u_j'' T''}$
- Molecular diffusion: $\overline{u_i''\tau_{ij}}$
- Turbulent transport: $(1/2)\overline{\rho u_j'' u_i'' u_i''}$

The Reynolds stress is usually evaluated through various turbulence models. A common approach is to use the Boussinesq approximation, which assumes the turbulent stress is proportional to the mean rate of strain by the turbulent, or eddy, viscosity. Thus, the Reynolds stress can be analogously treated as the viscous stress terms, which is given by,

$$-\overline{\rho u_i'' u_j''} = \mu_t\left(\frac{\partial \widetilde{u}_j}{\partial x_i} + \frac{\partial \widetilde{u}_i}{\partial x_j} - \frac{2}{3}\frac{\partial \widetilde{u}_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\bar{\rho}k\delta_{ij} \tag{3.8}$$

where μ_t is the eddy viscosity. As a result, most eddy-viscosity turbulence models are incorporated into the equations by adding this extra viscosity term to the viscous stresses. The turbulent heat flux can be modeled through a Reynolds analogy, which is given by,

$$c_p\overline{\rho u_j'' T''} \approx -\frac{c_p \mu_t}{Pr_t}\frac{\partial \widetilde{T}}{\partial x_j} \tag{3.9}$$

where Pr_T is the turbulent Prandtl number, which is usually 0.9 for air. The molecular diffusion and turbulent transport terms, on the other hand, can be neglected. A further reduced form of the RANS equations [89] is given by,

$$\begin{aligned}
\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\widetilde{u}_i\right) &= 0 \\
\frac{\partial \bar{\rho}\widetilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{\rho}\widetilde{u}_i\widetilde{u}_j\right) + \frac{\partial \bar{P}}{\partial x_i} &= \frac{\partial \widetilde{\tau}_{ij}^T}{\partial x_j} \\
\frac{\partial \bar{\rho}\widetilde{E}}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{\rho}\widetilde{u}_j\widetilde{H}\right) &= \frac{\partial}{\partial x_j}\left(\widetilde{u}_i\widetilde{\tau}_{ij}^T\right) + \frac{\partial}{\partial x_j}\left(c_p\left[\frac{\widetilde{\mu}}{Pr} + \frac{\widetilde{\mu}_T}{Pr_T}\right]\frac{\partial \widetilde{T}}{\partial x_j}\right)
\end{aligned} \tag{3.10}$$

where

$$\begin{aligned}
\widetilde{\tau}_{ij}^T &= \widetilde{\tau}_{ij} + \left(-\overline{\rho u_i'' u_j''}\right) = \left(\mu + \mu_t\right)\left(\frac{\partial \widetilde{u}_j}{\partial x_i} + \frac{\partial \widetilde{u}_i}{\partial x_j} - \frac{2}{3}\frac{\partial \widetilde{u}_k}{\partial x_k}\delta_{ij}\right) \\
\bar{P} &= \bar{p} + \frac{2}{3}\bar{\rho}k
\end{aligned}$$

Here, $\bar{P}$ is a modified pressure term to account for the missing turbulent kinetic energy term.

3.2 Spalart-Allmaras Turbulence Model

The Spalart-Allmaras (SA) turbulence model [90] is a one-equation model that closes the system in Equation (3.6) through modeling the transport equation for the kinematic eddy turbulent viscosity under the Boussinesq approximation. This is given by,

$$\frac{\partial \widehat{\nu}}{\partial t} + \widetilde{u}_j\frac{\partial \widehat{\nu}}{\partial x_j} = c_{b1}\left(1 - f_{t2}\right)\widehat{S}\widehat{\nu} - \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\widehat{\nu}}{d}\right)^2 + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left(\left(\nu + \widehat{\nu}\right)\frac{\partial \widehat{\nu}}{\partial x_j}\right) + c_{b2}\frac{\partial \widehat{\nu}}{\partial x_i}\frac{\partial \widehat{\nu}}{\partial x_i}\right] \tag{3.11}$$

and the eddy viscosity is evaluated via

$$\mu_t = \bar{\rho}\widehat{\nu}f_{v1} \tag{3.12}$$

The coefficient f_{v1} is given by,

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \qquad \chi = \frac{\widehat{\nu}}{\nu} \tag{3.13}$$

Here, $\nu = \mu/\rho$ is the molecular kinematic viscosity, and μ is the molecular dynamic viscosity. The term $\widehat{S}$ is given by,

$$\widehat{S} = \Omega + \frac{\widehat{\nu}}{\kappa^2 d^2}f_{v2} \tag{3.14}$$

where $\Omega = \sqrt{2W_{ij}W_{ij}}$ is the vorticity magnitude and d is the distance to the nearest wall. Additional terms are given by the following:

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, \quad f_w = g\left[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right]^{1/6}, \quad g = r + c_{w2}(r^6 - r)$$
$$r = \min\left[\frac{\widehat{\nu}}{\widehat{S}\kappa^2 d^2}, 10\right], \quad f_{t2} = c_{t3}\exp(-c_{t4}\chi^2), \quad W_{ij} = \frac{1}{2}\left(\frac{\partial \widetilde{u}_i}{\partial x_j} - \frac{\partial \widetilde{u}_j}{\partial x_i}\right)$$

The constants are given by,

$$c_{b1} = 0.1355, \quad \sigma = 2/3, \quad c_{b2} = 0.622, \quad \kappa = 0.41$$
$$c_{w2} = 0.3, \quad c_{w3} = 2, \quad c_{w1} = 7.1, \quad c_{t3} = 1.2$$
$$c_{t4} = 0.5 \quad c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}$$

3.2.1 Compressibility Correction

In Overflow, a compressibility correction [91] is included by default to improve behavior in compressible mixing layers. This correction is achieved through incorporating the following term on the right-hand-side of Equation (3.11),

$$-C_5 \frac{\widehat{\nu}^2}{a^2}\frac{\partial \widetilde{u}_i}{\partial x_j}\frac{\partial \widetilde{u}_i}{\partial x_j} \tag{3.15}$$

where a is the local speed of sound and $C_5 = 3.5$.

3.2.2 Rotation/Curvature Correction

A curvature correction [92] can be applied to the SA model (indicated by SA-RC) to account for rotation and curvature effects. This is incorporated by replacing the $c_{b1}(1-f_{t2})\widehat{S}\widehat{\nu}$ with $c_{b1}(f_{r1}-f_{t2})\widehat{S}\widehat{\nu}$. The term f_{r1} is given by,

$$f_{r1} = (1+c_{r1})\frac{2r*}{1+r*}\left[1 - c_{r3}\tan^{-1}(c_{r2}\hat{r})\right] - c_{r1} \tag{3.16}$$

The variables $r*$ and $\widehat{r}$ are given by

$$\begin{aligned} r* &= S/\omega \\ \widehat{r} &= \frac{2\omega_{ik}S_{jk}}{D^4}\left(\frac{DS_{ij}}{Dt} + (\varepsilon_{imn}S_{jn} + \varepsilon_{jmn}S_{in})\Omega'_m\right) \end{aligned} \tag{3.17}$$

Here, S_{ij} is the strain rate tensor. The additional variables introduced are given by

$$\begin{aligned} &S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \quad \omega_{ij} = \frac{1}{2}\left[\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) + 2\varepsilon_{mji}\Omega'_m\right] \\ &S^2 = 2S_{ij}S_{ij}, \qquad \omega^2 = 2\omega_{ij}\omega_{ij} \quad D^2 = \frac{1}{2}\left(S^2 + \omega^2\right) \\ &c_{r1} = 1.0 \quad c_{r2} = 12 \quad c_{r3} = 1.0 \end{aligned} \tag{3.18}$$

The term ε_{ijk} is the Levi-Civita symbol, or alternating tensor. The term DS_{ij}/Dt is the material derivative of the strain rate tensor, which is given by,

$$\frac{DS_{ij}}{Dt} = \frac{\partial S_{ij}}{\partial t} + u_K\frac{\partial S_{ij}}{\partial x_k} \tag{3.19}$$

Additionally, Ω' is the term representing the rotation of the reference frame.

3.2.3 Quadratic Constitutive Relations

3.2.3.1 QCR2000

Spalart [93] introduced several terms to the Reynolds stresses called the Quadratic Constitutive Relations (QCR) to introduce nonlinearities in the turbulence, allowing for improved modeling of corner-type flows. The additional terms added are incorporated in the following manner,

$$\tau_{ij,QCR2000} = \tau_{ij} - C_{cr1}\Big[O_{ik}\tau_{jk} + O_{jk}\tau_{ik}\Big] \tag{3.20}$$

Here, τ_{ij} is the original turbulent stress tensor from the Boussinesq approximation evaluated using the original SA turbulence model, and O_{ik} is the skew-symmetric normalized rotation tensor, which is given by,

$$O_{ik} = \frac{2W_{ik}}{\sqrt{\dfrac{\partial u_m}{\partial x_n}\dfrac{\partial u_m}{\partial x_n}}}, \qquad W_{ik} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_k} - \frac{\partial u_k}{\partial x_i}\right) \tag{3.21}$$

The coefficient C_{cr1} in the formulation is 0.3. Though not originally named as such, the modification is known as QCR2000. The addition of QCR2000 is indicated in the turbulence modeling name convention, e.g. SA-QCR2000, SA-RC-QCR2000, etc.

3.2.3.2 QCR2013-V

Mani et al [94] introduced a modified version of the QCR2000 model, known as QCR2013. The modification improves prediction of secondary vortical flows developed form corners by eddy-viscosity turbulence models. The new Reynolds stress is given by,

$$\tau_{ij,QCR2013} = \tau_{ij,QCR2000} - C_{cr2}\mu_t\sqrt{2S*_{mn}\,S*_{mn}}\delta_{ij} \tag{3.22}$$

where

$$S*_{ij} = S_{ij} - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{3.23}$$

Here, S_{ij} is the strain rate tensor, and the coefficient $C_{cr2} = 2.5$. However, issues with the added term have been noted in wake regions when μ_t is not small [95], and a fix for this was proposed by Rumsey et al [96], which is called QCR2013-V. This approach uses vorticity instead of the strain, which is given by,

$$\tau_{ij,QCR2013-V} = \tau_{ij,QCR2000} - C_{cr2}\mu_t\sqrt{2W_{mn}W_{mn}}\delta_{ij} \tag{3.24}$$

3.3 Inviscid Flux Discretization

This section describes the numerical schemes used to discretize the inviscid terms in the Navier-Stokes equations, i.e. $\partial\vec{E}/\partial\xi$, $\partial\vec{F}/\partial\eta$, $\partial\vec{G}/\partial\zeta$ in Equation (1.4). Each scheme presented describes the discretization in one-dimension, and in practice the discretization is applied separately in each dimension. For the ease of explanation, each scheme is considered on the Euler equations in one-dimension, which represents a system of hyperbolic conservation laws given by

$$\frac{\partial\vec{q}}{\partial t} + \frac{\partial\vec{f}(\vec{q},x,t)}{\partial x} = 0 \tag{3.25}$$

where $\vec{q}$ is the vector of conserved variables $[\rho, \rho u, \rho E]^T$ and $\vec{f}(\vec{q}, x, t)$ is the vector flux $[\rho u, \rho u^2 + P, \rho u H]^T$.

3.3.1 Second-Order Central-Difference

The second-order central-difference scheme (IRHS=0,FSO=2.0,SMOO=1) is based on the "JST" scheme originally developed by Jameson et al [97], which adds second and fourth-order dissipation terms to dissipate oscillations and prevent even-odd decoupling associated with the plain second-order central finite difference scheme. Thus, the flux derivative can be approximated in the following way,

$$\frac{\partial \vec{f}}{\partial x} \approx \frac{\widetilde{\vec{f}}_{i+1/2} - \widetilde{\vec{f}}_{i-1/2}}{\Delta x} \tag{3.26}$$

where

$$\widetilde{\vec{f}}_{i+1/2} = \frac{1}{2}\left(\vec{f}_{i+1} + \vec{f}_i\right) - \frac{1}{2}\left(a_{i+1} + a_i\right)\left(\vec{D}^{(2)}_{i+1/2} - \vec{D}^{(4)}_{i+1/2}\right) \tag{3.27}$$

Here, a_i is the spectral radius of the flux Jacobian $[\partial \vec{f}/\partial \vec{q}]$. From taking only the non-dissipative terms, the differencing produces a 2nd-order centered difference. The terms $D^{(2)}_{i+1/2}$ and $D^{(4)}_{i+1/2}$ are the second and fourth-order dissipation terms respectively. They are given by,

$$\begin{aligned} \vec{D}^{(2)}_{i+1/2} &= \epsilon^{(2)}_{i+1/2}\left(\vec{q}_{i+1} - \vec{q}_i\right) \\ \vec{D}^{(4)}_{i+1/2} &= \epsilon^{(4)}_{i+1/2}\left(\vec{q}_{i+2} - 3\vec{q}_{i+1} + 3\vec{q}_i - \vec{q}_{i-1}\right) \end{aligned} \tag{3.28}$$

The variables $\epsilon^{(2)}_{i+1/2}$ and $\epsilon^{(4)}_{i+1/2}$ are given by,

$$\epsilon^{(2)}_{i+1/2} = (1/2)\left(\epsilon^{(2)}_{i+1} + \epsilon^{(2)}_i\right), \quad \epsilon^{(2)}_i = \epsilon_2 \max\left(\gamma_{i+1}, \gamma_i, \gamma_{i-1}\right), \quad \epsilon^{(4)}_{i+1/2} = \max\left(0, \epsilon_4 - \epsilon^{(2)}_{i+1/2}\right) \tag{3.29}$$

The coefficients ϵ_2 (DIS2) and ϵ_4 (DIS4) are generally set to 0.5 and 0.01, respectively, and the term γ_i is a pressure sensor defined by

$$\gamma_i = \left|\frac{p_{i+1} - 2p_i + p_{i-1}}{p_{i+1} + 2p_i + p_{i-1}}\right| \tag{3.30}$$

The second-order dissipation is only active in regions with shocks or sharp gradients, which is detected by γ_i. This reduces the scheme locally to first order accuracy to capture the shock. On the other hand, the fourth-order dissipation introduces a third-order error term scaled by Δx^3. This provides a lower dissipation that both stabilizes the plain centered difference scheme and ensures second-order accuracy in smooth flow regions under grid-refinement.

3.3.2 Third-Order Roe Scheme with Koren Limiter

This scheme is a high-resolution finite-volume scheme that uses Roe's approximate Riemann solver [98] (IRHS=4) and Koren third-order flux limiter [99] (FSO=3.0, DELTA=1). Using the Godunov approach to discretize Equation (3.25), we obtain the following,

$$\vec{q}_i^{n+1} - \vec{q}_i^n = -\frac{\Delta t}{\Delta x}\left(\vec{f}^{Roe}_{i+1/2} - \vec{f}^{Roe}_{i-1/2}\right) \tag{3.31}$$

where

$$\vec{f}^{Roe}_{i+1/2} = \vec{f}^{Roe}\left(\vec{q}^{L}_{i+1/2}, \vec{q}^{R}_{i+1/2}\right), \tag{3.32}$$

where $\vec{f}^{Roe}_{i+1/2}$ is the Roe-flux, and $\vec{q}^{L}_{i+1/2}$ and $\vec{q}^{R}_{i+1/2}$ are the left and right states at the right cell interface. The Roe approximate solver is given by

$$\vec{f}^{Roe}\left(\vec{q}^{L}, \vec{q}^{R}\right) = \frac{1}{2}\left(\vec{f}^{L} + \vec{f}^{R}\right) - \frac{1}{2}\sum_{i=1} \widetilde{\alpha}_i |\widetilde{\lambda}_i| \vec{\widetilde{K}}^{(i)} \tag{3.33}$$

where $\widetilde{\lambda}_i$ and $\vec{\widetilde{K}}^{(i)}$ are the eigenvalues and right eigenvectors of the flux Jacobian $A = \frac{\partial \vec{f}}{\partial \vec{q}}$ constructed using the *Roe-averaged* variables. Roe-averaging for a variable a is given by

$$\widetilde{a} = \frac{\sqrt{\rho_L} a_L + \sqrt{\rho_R} a_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{3.34}$$

where ρ_L and ρ_R are the densities at the left and right states, respectively. The scalar $\widetilde{\alpha}_i$ are the wave strengths from mapping the interface differences onto the right eigenvectors, which are given by

$$\Delta\vec{q} = \vec{q}_R - \vec{q}_L = \sum_{i=1}^{m} \widetilde{\alpha}_i \vec{\widetilde{K}}^{(i)} \tag{3.35}$$

The left and right states can be computed using the Koren limiter. The Koren limiter achieves a third-order accurate scheme by selecting $\kappa = 1/3$ in the following discretization:

$$\begin{aligned} q^{L}_{i+1/2} &= q_i + \frac{1+\kappa}{4}(q_{i+1} - q_i) + \frac{1-\kappa}{4}(q_i - q_{i-1}) \\ q^{R}_{i+1/2} &= q_{i+1} + \frac{1+\kappa}{4}(q_i - q_{i+1}) + \frac{1-\kappa}{4}(q_{i+1} - q_{i+2}) \end{aligned} \tag{3.36}$$

The scheme can be reformulated into a limiter formulation given by

$$\begin{aligned} q^{L}_{i+1/2} &= q_i + \frac{1}{2}\phi(R_i)(q_i - q_{i-1}) \\ q^{R}_{i+1/2} &= q_{i+1} + \frac{1}{2}\phi(1/R_{i+1})(q_{i+1} - q_{i+2}) \end{aligned} \tag{3.37}$$

where the limiter function $\phi(R)$ is given by

$$\phi(R) = \frac{2R^2 + R}{2R^2 - R + 2} \tag{3.38}$$

The variable R_i is given by

$$R_i = \frac{q_{i+1} - q_i}{q_i - q_{i-1}} \tag{3.39}$$

3.4 Time-Advancement

In this dissertation, the focus of the flow computations are on steady state solutions. The most common approach to compute these solutions is to advance the flow solutions in time to drive the time-derivatives to either zero or below a specified tolerance. Overflow, in particular, mainly uses implicit schemes to avoid the Courant-Friedrichs-Lewy time-step restrictions, or CFL condition, that are common with explicit

schemes. The scalar penta-diagonal scheme of Pulliam and Chaussee [100] (ILHS=0) and the Symmetric Successive Over-Relaxation (SSOR) scheme [101] (ILHS=6-8,26-28) are presented here and used later for computation. Low-Mach Preconditioning [102, 103] (BIMIN=-1.0) is also presented, which is used in cases with low-speed and are nearly incompressible ($0.0 < M_\infty \leq 0.3$). To present these schemes, the two-dimensional Euler system is considered, which is given by,

$$\begin{gathered} \frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{f}}{\partial x} + \frac{\partial \vec{g}}{\partial y} = 0 \\ \vec{q} = \left[\rho, \rho u, \rho v, \rho E\right]^T, \quad \vec{f} = \left[\rho u, \rho u^2 + p, \rho uv, \rho uH\right]^T, \quad \vec{g} = \left[\rho v, \rho uv, \rho v^2 + p, \rho vH\right]^T \end{gathered} \tag{3.40}$$

A semi-discretized form based on the backward Euler scheme is given by

$$\Delta \vec{q}^n + \Delta t \left(\partial_x \vec{f}^{n+1} + \partial_y \vec{g}^{n+1}\right) = 0 \tag{3.41}$$

Here, $\Delta \vec{q}^n = \vec{q}^{n+1} - \vec{q}^n$, and Beam and Warming [84] introduced the following linearization for the flux terms at $(n+1)$, which are given by

$$\begin{aligned} \vec{f}^{n+1} &= \vec{f}^n + \Delta t \left(\frac{\partial \vec{f}}{\partial t}\right)^n + O(\Delta t^2) \\ &= \vec{f}^n + \Delta t \left(A \frac{\partial \vec{q}}{\partial t}\right)^n + O(\Delta t^2) \\ &= \vec{f}^n + A^n \left(\vec{q}^{n+1} - \vec{q}^n\right) + O(\Delta t^2) \end{aligned} \tag{3.42}$$

where A is the flux Jacobian $\partial \vec{f} / \partial \vec{q}$. A similar linearization can be done for the y direction, with the flux Jacobian denoted by B. This results in the following formulation,

$$\left[I + \Delta t \left(\delta_x A^n + \delta_y B^n\right)\right] \Delta \vec{q}^n = -\Delta t \left(\delta_x \vec{f}^n + \delta_y \vec{g}^n\right) \tag{3.43}$$

where δ_x and δ_y are the spatial discretization operators approximating the spatial first-derivative in each dimension. In Overflow, the operators are generally based on the central-differencing scheme presented in Section 3.3.1.

3.4.1 Pulliam-Chaussee ARC3D Scalar Pentadiagonal Scheme

This scheme uses an approximate factorization of Equation (3.43), resulting in the following form,

$$\left[I + \Delta t \delta_x A^n\right] \left[I + \Delta t \delta_y B^n\right] \Delta \vec{q}^n = -\Delta t \left(\delta_x \vec{f}^n + \delta_y \vec{g}^n\right) \tag{3.44}$$

Pulliam and Chaussee [100] substituted the flux Jacobians with their respective eigendecompositions, which are given by

$$A = T_x \Lambda_x T_x^{-1}, \qquad B = T_y \Lambda_y T_y^{-1}. \tag{3.45}$$

This results in the following linear system,

$$T_x \left[I + \Delta t \delta_x \Lambda_x\right] \widehat{N} \left[I + \Delta t \delta_y \Lambda_y\right] T_y^{-1} \Delta \vec{q}^n = \Delta t \left(\delta_x \vec{f}^n + \delta_y \vec{g}^n\right) \tag{3.46}$$

where $\widehat{N} = T_x^{-1} T_y$. Here, the implicit operators $\left[I + \Delta t \delta_x \Lambda_x\right]$ and $\left[I + \Delta t \delta_y \Lambda_y\right]$ are scalar pentadiagonal, since the δ_x and δ_y terms are the centered-differencing operators originally presented in Section 3.3.1. Because the blocks are now diagonalized, the implicit operators can be solved using a scalar pentadiagonal solver instead of using a block pentadiagonal system.

3.4.2 Successive Symmetric Over-Relaxation

The SSOR approach was originally implemented into Overflow by Nichols et al [101]. Unlike the aforementioned approach, SSOR is directly applied to the time-linearized form without alternating direction factorization given in Equation (3.43), which is given by

$$\Delta \vec{q}_{j,k}^{mm+1} = (1-\omega)\,\Delta \vec{q}_{j,k}^{mm} + \omega \left(-\overline{\delta_x \vec{f}^n + \delta_y \vec{g}^n} - \overline{\delta_x A} \Delta \vec{q}_{j,k}^{mm-1} - \overline{\delta_y B_L} \Delta \vec{q}_{j,k-1}^{mm+1} - \overline{\delta_y B_R} \Delta \vec{q}_{j,k+1}^{mm} \right) \quad (3.47)$$

where mm indicates the iteration of the SSOR, and $\omega = 0.9$ is the relaxation factor. The overbar indicates the off-diagonal blocks that have been pre-multiplied by the diagonal block. A left and right matrix coefficients for B are also denoted to indicate Jacobian terms operating on the "left" and "right" support in the y derivatives.

3.4.3 Low-Mach Preconditioning

The Low-Mach Preconditioning approach can be presented through the two-dimensional conservative system given by Equation (3.40). The primitive variables $\vec{q}^p = \left[\rho, u, v, p\right]^T$ are also considered. The transformation matrix M is given by

$$M = \frac{\partial \vec{q}}{\partial \vec{q}^p}, \qquad M^{-1} = \frac{\partial \vec{q}^p}{\partial \vec{q}} \quad (3.48)$$

Expressing the quasilinear form of the conservative equations in terms of M and the primitive variables results in the following,

$$M \partial_t \vec{q}^p + AM \partial_x \vec{q}^p + BM \partial_y \vec{q}^p = 0 \quad (3.49)$$

Preconditioning seeks to replace the matrix M in front of the time derivative with another matrix Γ, which results in the following,

$$\Gamma \partial_t \vec{q}^p + AM \partial_x \vec{q}^p + BM \partial_y \vec{q}^p = 0 \quad (3.50)$$

The matrix Γ^{-1} can be multiplied through the system,

$$\partial_t \vec{q}^p + \Gamma^{-1} AM \partial_x \vec{q}^p + \Gamma^{-1} BM \partial_y \vec{q}^p = 0 \quad (3.51)$$

The system can then be re-expressed in terms of the conservative variables, resulting in

$$\begin{aligned} M^{-1}\partial_t \vec{q} + \Gamma^{-1} AMM^{-1} \partial_x \vec{q} + \Gamma^{-1} BMM^{-1} \partial_y \vec{q} &= 0 \\ \partial_t \vec{q} + M\Gamma^{-1} A \partial_x \vec{q} + M\Gamma^{-1} B \partial_y \vec{q} &= 0 \\ \partial_t \vec{q} + M\Gamma^{-1} \left(\partial_x \vec{f} + \partial_y \vec{g} \right) &= 0 \end{aligned} \quad (3.52)$$

This can be applied to the implicit formulation in Equation (3.43), which is given by

$$\left[I + \Delta t \left(\delta_x M\Gamma^{-1}A^n + \delta_y M\Gamma^{-1}B^n\right)\right]\Delta\vec{q}^n = -\Delta t M\Gamma^{-1}\left(\delta_x \vec{f}^n + \delta_y \vec{g}^n\right) \tag{3.53}$$

The transformation matrices M and M^{-1} are given by,

$$M = \begin{bmatrix} \rho/p & 0 & 0 & -\rho^2/p \\ \rho u/p & \rho & 0 & -\rho^2 u/p \\ \rho v/p & 0 & \rho & -\rho^2 v/p \\ e/p & \rho u & \rho v & \rho(1/(\gamma-1) - e/p) \end{bmatrix}, \tag{3.54}$$

$$M^{-1} = \begin{bmatrix} (\gamma-1)(u^2+v^2)/2 & -(\gamma-1)u & -(\gamma-1)v & (\gamma-1) \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ (-p/\rho + (\gamma-1)(u^2+v^2)/2)/\rho & -(\gamma-1)u/\rho & -(\gamma-1)v/\rho & (\gamma-1)/\rho \end{bmatrix} \tag{3.55}$$

Weiss and Smith [103] define the preconditioning matrix Γ as

$$\Gamma = \begin{bmatrix} \rho/\beta'\gamma p & 0 & 0 & -\rho^2/p \\ \rho u/\beta'\gamma p & \rho & 0 & -\rho^2 u/p \\ \rho v/\beta'\gamma p & 0 & \rho & -\rho^2 v/p \\ (e/p+1)/\beta'\gamma - 1 & \rho u & \rho v & \rho(1/(\gamma-1) - e/p) \end{bmatrix} \tag{3.56}$$

where $\beta' = \beta/[1 + (\gamma-1)\beta]$. The parameter β is typically set to

$$\beta = \max(\min(M_\infty^2, 1.0), \beta_{min}) \tag{3.57}$$

where $\beta_{min} = 3M_\infty^2$, and M_∞ is a freestream or reference Mach number.

Chapter 4

Boundary Coupling for Hyperbolic Overset Mesh Generation

A critical aspect of overset meshing is the need for meshes to maintain sufficient overlap such that the outer fringe points of a given mesh have sufficient support from the interior points of the neighboring mesh to perform interpolation. When manually constructing such a mesh, a user has a few strategies to ensure sufficient overlap that namely involve controlling the outer extents of each mesh.

This chapter explores the application of overset boundaries for hyperbolic meshes with overlapping initial conditions. Motivations for the overlap problem and mathematical foundations are discussed, which is then followed by the details of the implementation. Two possible implementations are discussed, where one approach involves the re-purposing of a pre-existing boundary condition, while the other implementation is effectively similar to the approach more commonly used in overset boundaries for other PDE or flow solver applications. Other implementation considerations, such as domain connectivity, are also discussed. Finally, examples of some applications are demonstrated in both surface and volume meshing.

It should also be noted that the following discussion is under the premise that the mesh is generated from a BRep solid using the automatic meshing procedure described in Section 2.4. While this facilitates certain aspects of the setup and implementation, it is not strictly necessary, and the approach can be utilized in manual mesh generation.

The chapter is structured as follows. Section 4.1 presents the motivation behind the development of the coupled boundary conditions. Section 4.2 provides the mathematical analysis for the feasibility of the boundary conditions for the various hyperbolic mesh generation equations. Section 4.3 provides an overview of the numerical implementation, as well as two approaches for coupling that have been found to be feasible. Finally, Section 4.4 demonstrates the applications of the boundary conditions for complex, three-dimensional geometries with aeronautical applications.

4.1 Motivation

Throughout the history of hyperbolic mesh generation development [27,50,57], only a handful of boundary conditions were ever considered. The purpose of these boundary conditions was simply to ensure the equations were well-behaved and, to a lesser extent, well-posed. The simplest (or rather, most intuitive) boundary is the periodic boundary, which requires that the initial data is a "closed" curve or surface. Other boundary conditions are empirical and rely on one of the directions to be prescribed based on (axi-) symmetry or constant planes in the underlying geometry. Barring these options, the most general boundary condition is the splay boundary, which is an ad-hoc extrapolation from the interior of the mesh. In manual meshing, these approaches are sufficient since the initial data can be constructed to take advantage of these boundaries. If one is constructing a mesh for a simple fuselage or a wing, periodic boundaries can be applied annularly or in the chord-wise direction, respectively.

More recently, with the introduction of automatic overset boundaries for BRep geometries [82, 83], the initial data appears significantly more "fractured", meaning overset boundaries are now present where previously a manual user would not place them. This is due to the fact that the automation approach effectively generates a mesh for nearly every node, edge, and face in the geometry, whereas the conventional approach is to attempt to represent as many of these geometric features as possible within a single mesh. Thus, the meshes in the automatic approach are not constructed in a way such that the empirically-prescribed boundaries are applicable, resulting in the use of splay as the most feasible boundary condition. Because of the meshing approach, the fracturing and use of splay boundaries can appear in both surface meshes and the resulting volume meshes. Figure 4.1 shows a "fractured" set of collar meshes at the intersection between a fuselage and a wing-like component. Here, four collar meshes (magenta, navy, blue, green) are generated hyperbolically by marching outward onto the fuselage from the junction curve. These meshes are generated independently of each other with splay boundary conditions, which

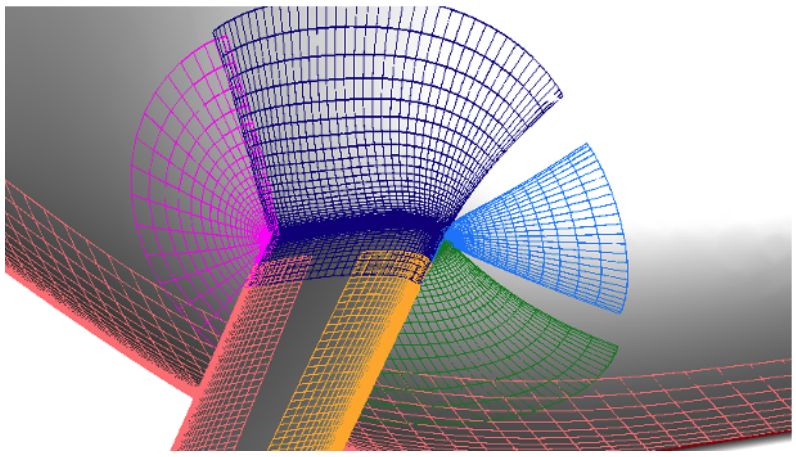


Figure 4.1: Fractured collar surface meshes generated hyperbolically (magenta, navy, blue, green) with poor overlap due to uncontrolled splay.

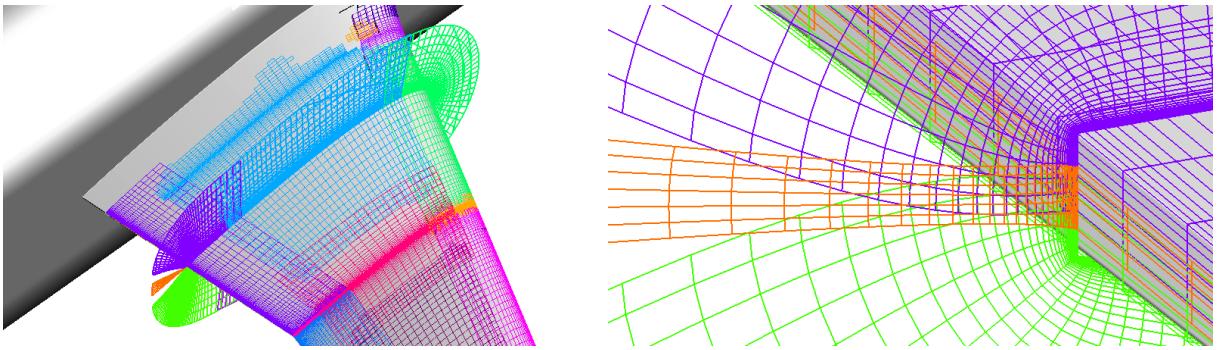


Figure 4.2: Fractured volume meshes generated on the Juncture Flow Model 0015 wing trailing edge with poor overlap due to insufficient splay.

do not guarantee any overlap and can result in gaps between grids. User iteration is required to ensure overlap is maintained in the annular direction.

Figure 4.2 shows how the fracturing can occur in the volume meshing. In this case, because there are multiple meshes in the trailing edge regions, the mesh boundaries diverge away from each other due to insufficient splay. This can create further problems downstream in the mesh generation process, where the off-body meshes need to be present to cover the resulting gap to ensure there are no orphan points or unintended holes in the mesh. The off-body mesh may not be properly sized to cover this gap, which may require excessive refinement in the off-body mesh.

The splay boundaries have two idiosyncrasies that make it non-ideal for wide-spread use in an automatic meshing context. The first is that the splay boundaries require a "splay parameter" to properly ensure the side boundaries bend sufficiently outward such that the minimum overlap between neighboring meshes are achieved. Often, the parameter may require some iteration to satisfy overlap requirements. This iterative procedure, however, is not amenable to automation since it is heavily case dependent, and the behavior is not easily characterized mathematically.

The second issue is that the splay boundary does not always result in sufficient splay across the entire "marching history" of the boundary. In some cases, a high splay may allow the outer grid layers to expand outwards and satisfy the overlap requirements but can still lack overlap at grid layers closer to the initial data. In the case of volume meshing, this may result in situations where there is insufficient overlap in the boundary layer region, while sufficient overlap may be retained everywhere else.

To tackle the overlap problem between hyperbolically-generated meshes, a "coupling" approach is proposed where meshes with overlapping *initial* data have coupled boundary conditions applied in the overlap region. In essence, the outer boundaries along the parameterizations making up the initial data will inherit their marching behavior from the corresponding point sitting on an overlapping neighboring curve or surface. From the perspective of solving the underlying hyperbolic system, this approach is effectively a form of the overset technique applied to the hyperbolic mesh generation equations.

4.2 Mathematical Foundations

In this section, key considerations are presented that enable the coupling approach. While it may seem intuitive to simply "interpolate" from a neighboring grid as it is done when solving other systems of PDEs [9,10], it is not immediately clear that such an approach is viable for the hyperbolic mesh generation system. Other more common PDEs exchange field data that typically do not influence the evolution of the local grid. The following provides the mathematical analysis to make the argument that the coupling strategy is viable for all three hyperbolically-generated mesh types: 2D field, surface, and volume. This analysis underpins the approach with some essential theoretical insights. However, it stops short of full mathematical rigor, which is beyond the scope of this work.

In all mesh types, it is necessary that the overlapping regions of the initial conditions are *isometric*. That is, the overlapping initial data simply only needs to represent the same underlying shape or geometry, but it is not necessary that the parameterization of the region is the same.

4.2.1 2D Field Meshes

We first consider the two-dimensional (2D) field mesh generation equations [50], where an initial curve is evolved to form an orthogonal mesh in $\mathbb{R}^2$. The field mesh equations are given by,

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \tag{4.1}$$

$$|\vec{r}_\xi \times \vec{r}_\eta| = \Delta A \tag{4.2}$$

where $\vec{r}\,(\xi,\eta) = \langle x\,(\xi,\eta)\,, y\,(\xi,\eta)\rangle$ describe the coordinates of the generated mesh. Here we observe that the first equation is the orthogonality condition, and the second equation enforces the local elemental surface area of the mesh determined by ΔA. Typically, the equations are re-arranged into a different form that more closely resembles a hyperbolic system. This form is given by,

$$\begin{aligned} &\vec{r}_\eta + \frac{\Delta A}{r_\xi^2}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\vec{r}_\xi = 0 \\ &\vec{r}\,(\xi,0) = \vec{r}_0\,(\xi) \end{aligned} \tag{4.3}$$

where $\vec{r}\,(\xi,\eta)$ is advanced in η in a "time"-like direction, and the initial data is described by $\vec{r}_0\,(\xi)$. Here we can observe that the system evaluates and advances a mesh by its local unit normal along the η direction scaled by the magnitude $\sigma = \Delta A / r_\xi$. This normal is the "principal" normal, which is invariant to the curve parameterization. Thus, the unit normal vectors in the overlapping region must have the same orientation provided the overlap region is isometric. It can also be shown that the magnitude term σ will also be the same between different parameterizations as well. First, we return to eq. (4.2) and rewrite the form using a cross-product identity,

$$\|\mathbf{a}\times\mathbf{b}\|^2 = \|\mathbf{a}\|^2\|\mathbf{b}\|^2 - (\mathbf{a}\cdot\mathbf{b})^2 \tag{4.4}$$

Thus eq. (4.2) can be reformulated into the following,

$$\Delta A = [r_\xi^2 r_\eta^2 - (\vec{r}_\xi \cdot \vec{r}_\eta)^2]^{1/2} \tag{4.5}$$

Using eq. (4.1) further simplifies the local area parameter into the following,

$$\Delta A = r_\xi \cdot r_\eta \tag{4.6}$$

which then allows us to then further simplify the magnitude,

$$\sigma = \Delta A / r_\xi = r_\eta \tag{4.7}$$

When the equations are solved, r_η is typically prescribed based on the desired growth behavior in the mesh. If this is consistently defined between the overlapping meshes, then the advanced magnitude in the overlapping region should be equivalent, along with the direction itself. Therefore, the solution transfer between overlapping meshes via interpolation is valid under isometry.

4.2.2 Surface Meshes

A similar analysis can be done for the more general problem in $\mathbb{R}^3$ of hyperbolic surface mesh generation using the modified system used by Chan and Buning [57]. The equations are given by

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \tag{4.8}$$

$$\vec{n} \cdot (\vec{r}_\xi \times \vec{r}_\eta) = \Delta S \tag{4.9}$$

$$\vec{n} \cdot \vec{r}_\eta = 0 \tag{4.10}$$

where $\vec{r}(\xi, \eta) = \langle x(\xi, \eta), y(\xi, \eta), z(\xi, \eta) \rangle$ describe the coordinates of a generated surface mesh in 3D space, $\vec{n} = \langle n_x, n_y, n_z \rangle$ is the local normal of the reference surface on which the mesh is generated, and ΔS is a similar area parameter to ΔA in the field mesh equations. The first two equations enforce similar conditions as eqs. (4.1) and (4.2). A third condition is added here to close the system by enforcing orthogonality of the marching direction with the local surface normal.

Like the field mesh system, we can re-write the system to appear more similarly to a hyperbolic system with ξ representing space and η representing time. We first start with a modification to the system using a triple-product identity on eq. (4.9), producing the following,

$$\begin{aligned} \vec{r}_\xi \cdot \vec{r}_\eta &= 0 \\ \vec{r}_\eta \cdot (\vec{n} \times \vec{r}_\xi) &= \Delta S \\ \vec{n} \cdot \vec{r}_\eta &= 0 \end{aligned} \tag{4.11}$$

The system can then be re-written into a matrix vector product,

$$\begin{aligned}\Big[A\Big]\,\vec{r}_\eta &= \vec{f} \\ \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ n_y z_\xi - n_z y_\xi & n_z x_\xi - n_x z_\xi & n_x y_\xi - n_y x_\xi \\ n_x & n_y & n_z \end{bmatrix} \vec{r}_\eta &= \begin{bmatrix} 0 \\ \Delta S \\ 0 \end{bmatrix}\end{aligned} \tag{4.12}$$

The hyperbolic system is given by re-writing the system in the following form,

$$\vec{r}_\eta - \Big[A\Big]^{-1}\,\vec{f} = 0 \tag{4.13}$$

Evaluating matrix inverse and vector multiply gives the following,

$$\vec{r}_\eta + \frac{\Delta S}{\det A}\begin{bmatrix} n_z y_\xi - n_y z_\xi \\ n_x z_\xi - x_\xi n_z \\ n_y x_\xi - n_x y_\xi \end{bmatrix} = 0 \tag{4.14}$$

To evaluate the determinant of A, we can rewrite the operator into a triple product,

$$\begin{aligned}\det A &= \vec{r}_\xi \cdot [(\vec{n} \times \vec{r}_\xi) \times \vec{n}] \\ &= \vec{r}_\xi \cdot [\vec{r}_\xi\,(\vec{n} \cdot \vec{n}) - \vec{n}\,(\vec{r}_\xi \cdot \vec{n})]\end{aligned} \tag{4.15}$$

Here, we observe that the local tangent of the curve is always orthogonal to the local surface normal, i.e. $\vec{r}_\xi \cdot \vec{n} = 0$. Thus, the determinant gives a similar result to that of the field meshes,

$$\det A = \vec{r}_\xi \cdot [\vec{r}_\xi\,(\vec{n} \cdot \vec{n})] = \vec{r}_\xi \cdot \vec{r}_\xi = r_\xi^2. \tag{4.16}$$

We can then factor out the surface normals as a matrix operator to rewrite the system in a "quasi-continuity" form. This is given by the following,

$$\vec{r}_\eta + \frac{\Delta S}{r_\xi^2}\begin{bmatrix} 0 & n_z & -n_y \\ -n_z & 0 & n_x \\ n_y & -n_x & 0 \end{bmatrix}\vec{r}_\xi = 0. \tag{4.17}$$

Note that if the reference surface lies entirely on the xy-plane (i.e. $\vec{n} = \langle 0, 0, 1\rangle$), we recover the 2D field mesh equations in eq. (4.3).

The matrix operator on the tangent vector in ξ here is effectively a cross-product operator of the local surface normal and the local unit tangent. Effectively, this results in the "front" or curve being propagated along the direction normal to both the unit tangent vector of the curve and the local surface normal. Providing isometry of the initial data and the reference surface are maintained in the overlapping regions, the generated normal vector should also be identical between overlapping grids and enable the solution exchange. The isometry of the reference surfaces implies the surface normals are consistent between grids, and the unit tangent is also consistent under isometry due to invariance with parameterization.

4.2.3 Volume Meshes

The final system to be considered is the volume mesh generation system in $\mathbb{R}^3$. Here, we consider the system developed by Steger and Rizk [53], which is given by,

$$\vec{r}_\xi \cdot \vec{r}_\zeta = 0 \tag{4.18}$$

$$\vec{r}_\eta \cdot \vec{r}_\zeta = 0 \tag{4.19}$$

$$\vec{r}_\zeta \cdot (\vec{r}_\xi \times \vec{r}_\eta) = \Delta V, \tag{4.20}$$

where $\vec{r}(\xi, \eta, \zeta) = \langle x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta) \rangle$ describes the coordinates of a volume, and ΔV is the elemental volume. The first two equations enforce the orthogonality condition with the marching direction ($\vec{r}_\zeta$), and the last equation enforces the local elemental volume. The system can be re-written in a "quasi-continuity" form, which begins by reformulating the system as a matrix-vector product,

$$\begin{aligned} \Big[A\Big]\,\vec{r}_\zeta &= \vec{f} \\ \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ y_\xi z_\eta - z_\xi y_\eta & z_\xi x_\eta - z_\eta x_\xi & x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \vec{r}_\zeta &= \begin{bmatrix} 0 \\ 0 \\ \Delta V \end{bmatrix} \end{aligned} \tag{4.21}$$

Inverting the matrix and putting all terms on the left-hand side gives the following,

$$\vec{r}_\zeta + \frac{\Delta V}{\det A} \begin{bmatrix} z_\xi y_\eta - y_\xi z_\eta \\ z_\eta x_\xi - z_\xi x_\eta \\ x_\eta y_\xi - x_\xi y_\eta \end{bmatrix} = 0 \tag{4.22}$$

The evaluate the matrix determinant, we can observe the following operation,

$$\det A = \vec{r}_\xi \cdot [\vec{r}_\eta \times (\vec{r}_\xi \times \vec{r}_\eta)] \tag{4.23}$$

Using a triple product identity, we find that the determinant is the square of the local elemental area in ξ and η,

$$\begin{aligned} \det A &= \vec{r}_\xi \cdot [\vec{r}_\eta \times (\vec{r}_\xi \times \vec{r}_\eta)] \\ &= (\vec{r}_\xi \times \vec{r}_\eta) \cdot (\vec{r}_\xi \times \vec{r}_\eta) \\ &= \|\vec{r}_\xi \times \vec{r}_\eta\|^2 \end{aligned} \tag{4.24}$$

Substituting back into eq. (4.22), we obtain the "quasi-continuity" form,

$$\vec{r}_\zeta + \frac{\Delta V}{\|\vec{r}_\xi \times \vec{r}_\eta\|^2} \begin{bmatrix} z_\xi y_\eta - y_\xi z_\eta \\ z_\eta x_\xi - z_\xi x_\eta \\ x_\eta y_\xi - x_\xi y_\eta \end{bmatrix} = 0 \tag{4.25}$$

Much like the original field equations, we find that the "quasi-continuity" form reveals that the mesh generation system is the evolution of a surface along its local unit surface normal scaled by the magnitude $\sigma = \Delta V / \|\vec{r}_\xi \times \vec{r}_\eta\|$. From looking at eq. (4.20), the ΔV contains an area magnitude term, resulting in $\sigma = r_\zeta$, which is a user-defined term based on the desired local mesh generation characteristics. So as long as this is consistent between overlapping grids, this guarantees there is consistency in the magnitude of the grid marching between the grids. The direction is held consistent between overlapping grids due to the invariance of the local surface normals to parameterization.

4.3 Numerical Approach

This section details the numerical implementation of the coupling between grids. Two coupling strategies are presented here: (1) loose coupling and (2) tight coupling. In concept, both implementations are similar to the original concept of overset boundaries applied when solving other PDEs, such as the Inviscid Euler Equations or Navier-Stokes equations. Domain connectivity between the overlapping initial data must be set up. Once the hyperbolic solutions are started, solution data must be transferred between grids through an interpolation procedure.

An overview of the domain connectivity procedure is first presented. Special considerations are also discussed when handling complex geometries, which were originally detailed in the work of Chuen and Chan [104]. Following that, the two coupling strategies are outlined. Like the domain connectivity, the discussion of the loose-coupling approach follows that of the work initially detailed in [104].

4.3.1 Domain Connectivity

Before any coupling can be applied, domain connectivity must be set up to ensure that all active points with insufficient stencil support to apply the numerical differencing scheme are able to interpolate their solution values from a neighboring overlapped grid. Such points are known as "fringe" points, and interpolating from their neighbors provides the notion that the grids provide a complete coverage of the initial data. For the purposes of boundary coupling for hyperbolic grid generation, fringe points are usually the external boundaries of the initial data. Hole-boundary fringe points are not considered, since there does not exist an equivalent or analogue when specifically solving the hyperbolic mesh generation equations. In the case of field or surface mesh generation, the fringe points are found on the initial curves at the $J = 1$ or $J = NJ$ nodes. In volume mesh, the fringe points are found on the boundary edges of $J = 1$, $J = NJ$, $K = 1$, and $K = NK$ in the initial surfaces. For points at external boundaries with no viable overlap to be found, other standard boundary conditions are then applied where appropriate [27,57]. The hyperbolic mesh generation equations have a three-point stencil in each dimension. As such, the minimum depth of overlap between grids to just perform the coupling is generally two cells of overlap. Higher-depth of overlap can be enforced to ensure sufficient support for the eventual numerical scheme that will be applied by the flow-solver.

To perform the domain connectivity, a donor-stencil search is performed in neighboring grids for each fringe point. The neighboring initial data are represented using piece-wise linear cells for 2D field or surface meshing, or piece-wise bilinear cells for volume meshing. The interpolation parameters are found for the corresponding fringe point, which then allow for extraction of fringe point data from neighboring grids as the grids are generated.

Several checks are typically applied to determine which interpolating stencil is appropriate, as there may be multiple viable stencils if there are multiple grids in the same vicinity. These checks include

- alignment of normals to the initial data
- distance between fringe point and projected point
- donor stencil quality

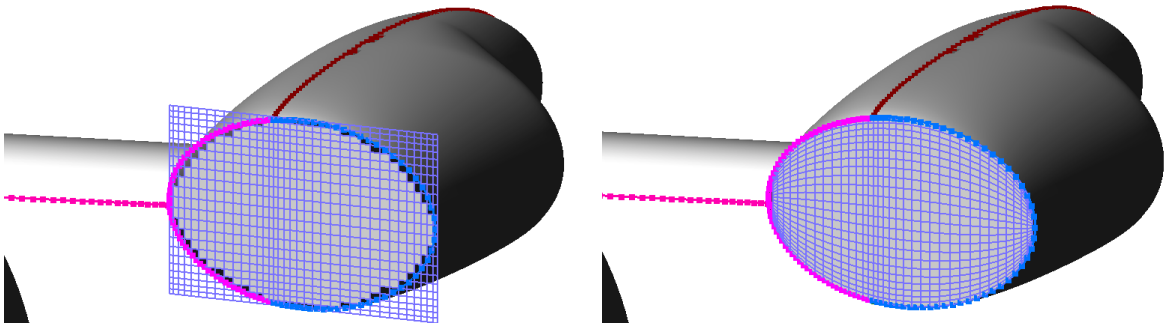
Occasionally, cell-size compatibility is also considered if there are multiple possible donor cells. Here, donor stencil quality is a measure computed from the weighted average based on proximity of a given fringe point to the stencil points of the donor cell and whether the stencil points are active interior points or are blanked or fringe points. A quality of 1.0 means the donor stencil is entirely constructed of non-fringe, non-blanked points, and is the preferred stencil if available. Occasionally, multiple donor stencils may still satisfy these criteria, in which case they are then further down-selected based on a first-in, first-out order. It is not clear if other additional criteria are necessary for maintaining grid quality in the coupling. Currently, only one stencil coupling is applied at each fringe point, although enabling interpolation from multiple viable donor stencils when available may be an interesting strategy to explore in future efforts.

4.3.2 Additional Volume Meshing Connectivity Considerations

Several additional considerations are presented here that ensure the robustness of volume meshing with coupling for complex geometries or even production-level applications. Some considerations are due to the "peculiarities" of using BRep geometries, while others are related to non-ideal situations for the current mesh generation procedures.

4.3.2.1 Face Mesh Reconstruction

A common face representation that appears frequently in BRep solids is a face with implicit boundary edges. This type of face is constructed such that it also contains an in/out function that is also mapped to its underlying parameterization. The regions lying inside the bounding curves are marked as "in" to indicate it is on the true geometry, while the regions outside the bounding curves are marked as "out". This allows for the underlying face surface definition to be constructed independently of its bounding curves, as not all regions within a given set of bounding curves can be cleanly mapped to a logically-rectangular u, v space.



(a) (b)

Figure 4.3: Rear face (indigo) of a motor fairing with bounding curves indicated in magenta and blue. (a) Parameterization of the face using a "trimmed" Cartesian mesh. (b) Reconstructed parameterization where outer boundaries lie completely within bounding curves.

However, this type of face mesh can present a problem for performing the coupling. In the automatic meshing approach, the hyperbolic mesh generation equations can only use the surface definition directly through a discrete parameterization. As was shown previously in Section 4.2, the behavior of these equations is highly dependent on the fact that the overall "shape" of the parameterization is isometric between overlapping meshes. The existence of the "out" points will change the local surface normals for the interior points adjacent to the implicit edges, thus making the coupling invalid due to the violation of isometry. In more practical situations where the coupling is either applied at the outer-most points of the interior regions of such faces or only a one-side coupling occurs and the face meshes are not coupled, the resulting meshes can exhibit poor mesh quality and contain negative Jacobians, negative cell volumes, etc. While generally these may occur in the "out" or blanked regions of the mesh, this is generally not allowed as a starting point for most CFD solvers, e.g. Overflow [21].

To deal with this issue in the automatic meshing framework, one can perform "mesh reconstruction", where a new parameterization is generated such that the outer boundaries (i.e. $J = 1, NJ$ and $K = 1, NK$) lies either on or within the outer bounding curves. This allows for a parameterization that is completely interior and avoids the issue of exterior points that can affect the local normal evaluation. Outside of the automatic meshing framework, however, it is incumbent on the mesh generator to avoid implicit geometry definitions to utilize the coupling. Figure 4.3 shows an example of a face mesh that is reconstructed. Prior to reconstruction, the face is represented implicitly through "trimming", where points lying outside of the bounding curves are blanked.

4.3.2.2 Decoupling of Collar Meshes and Filleted Faces

A common difficulty for the hyperbolic mesh generation equations in producing high-quality grids are grids with concave initial data. This typically includes "collar" meshes [64] and fillet-type faces with large turning angles. These meshes are typically difficult due to the fact that the surface normals converge towards each other, and typically require high splay amongst other input modifications to assist the

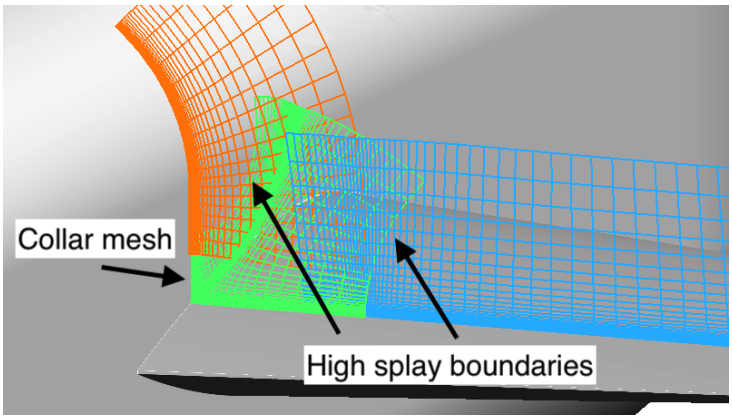


Figure 4.4: Collar mesh (green) decoupled due to the need for high-splay boundaries for good grid quality and already sufficient overlap with neighboring grids.

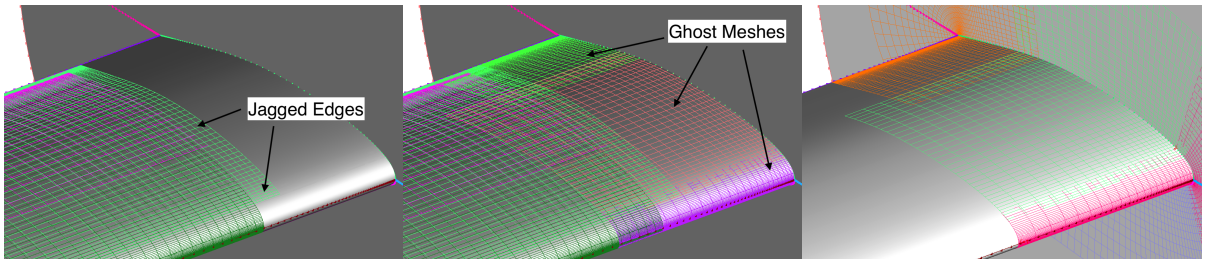
equations in growing out of the concavity. The high splay assists in diverting or "pulling" the normals away from the concavity. Incorporating the coupling may actually be detrimental to this process, in addition to adversely affecting the neighboring non-concave grids. Additionally, the meshes generally already exhibit sufficient overlap, and coupling may not be necessary. Thus, coupling of such meshes are avoided. Figure 4.4 shows an example of decoupling occurring for a collar mesh. Note the high concavity of the initial surface as it sits on a junction region.

4.3.2.3 Ghost Meshes

One issue with decoupling of certain meshes described in the previous section is the possibility of "jagged" boundaries. These boundaries arise from the lack of alignment of boundary edges for adjacent meshes after concave meshes are filtered out from the coupling procedure. These boundaries exhibit a mix of splay and coupled boundary conditions along a single boundary edge or corner. The mixed conditions can result in poor convergence with the iterative coupling process or poor grid quality. Through empirical investigation, it was discovered that it is generally good practice to ensure that, in collection of a set of coupled meshes:

- open (non-coupled) edges should collectively form or originate from a common curve
- the same type of boundary condition is applied to each edge boundary that make up a particular common curve

One way to incorporate these observations is to include "ghost" meshes. These meshes are sub-patches of collar meshes on the relevant side of the junction curve. In the case of the automeshing framework, the edge meshes surrounding filleted faces are also divided into sub-patches, where the patch lying not on the filleted side are only included. These sub-patches are usually included because the original geometric curve representing the intersection curve lies on a particular grid line, and the curve is also likely shared

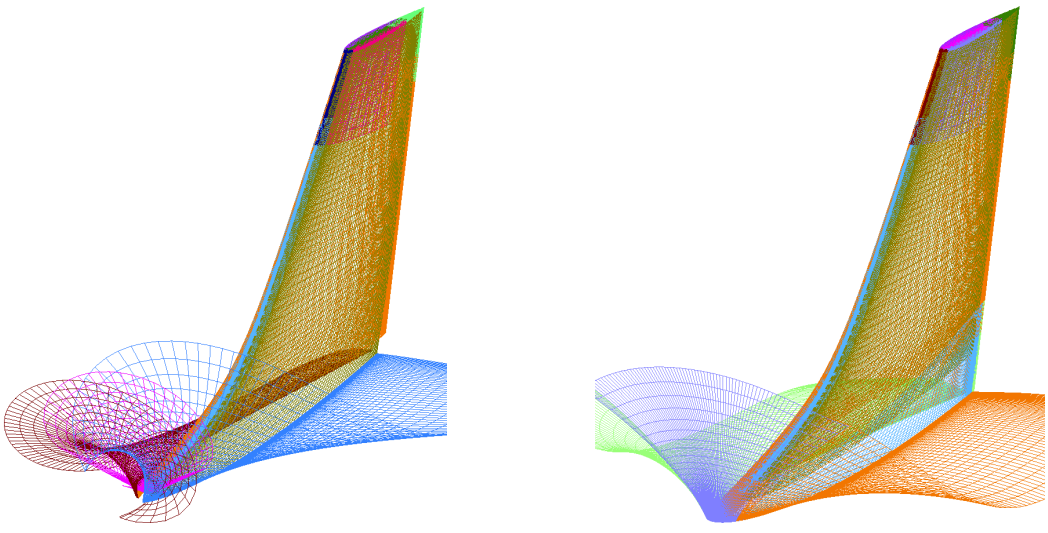


(a) (b) (c)

Figure 4.5: Initial surface geometry with and without "ghost" meshes near a fuselage junction region. (a) Original coupled surfaces without ghost meshes containing jagged boundaries. (b) Modified set of initial surfaces with incorporated ghost meshes. (c) Collar meshes from which ghost meshes are extracted.

with other similar meshes in the vicinity. Thus, the grids with open boundaries have edges aligned to a common curve. Additionally, boundary conditions can be consistently set for such edges.

These meshes are introduced into the overall connectivity for the sole purpose of improving overall grid quality and avoiding the mathematically ambiguous behavior at jagged boundaries. Once the coupling procedure is complete, these meshes are discarded, whereas the original, uncoupled node or edge meshes that the ghost meshes are extracted from are kept. Figure 4.5 compares how the incorporation of the ghost meshes can change the geometry of the coupled initial surface. The ghost meshes in Figure 4.5b now allow for a common geometrical edge to be shared with all initial surfaces that contain an uncoupled edge. This allows for the application of splay boundaries along the free edge of each mesh without introducing potential poor mesh quality. Figure 4.6 shows how the introduction of ghost meshes can improve the



(a) (b)

Figure 4.6: Vertical tail coupled volume mesh splay boundaries at root generated (a) without ghost meshes (note blue, magenta, and raspberry boundary surfaces) and (b) with ghost meshes (note orange, purple and light green boundary surfaces).

behavior of the splay boundaries for a vertical tail. Without ghost meshes, the volume meshes exhibit mixed coupling and splay boundaries, which can result in crossing of normal gridlines as the meshes are generated. On the other hand, incorporating the ghost meshes allow for splay boundaries to be generated in a similar fashion to how a splay boundary would appear if the tail initial surface were represented by a single mesh.

4.3.3 Loose-Coupling Approach

In this section, the implementation with the loose-coupling approach, or loosely-coupled boundary conditions (LCBC), is considered. This work was originally outlined in Chuen et al [104]. This approach initially came from an observation of the "float-along" boundary condition that is applied in hyperbolic surface meshing [57]. This particular boundary condition allows a mesh to follow a particular trajectory described by a prescribed curve as the meshes are generated. This approach is generally applied to hyperbolic surface meshing due to the fact that the surface meshes in the auto-meshing framework are not always generated with the same grid-spacing in the normal direction. A more-tightly coupled approach may require meshes to be marched out simultaneously with a common step size, which can result in all meshes needing to compute additional sub-steps that may not coincide with the prescribed normal grid spacing. The loose-coupling can also be applied to volume meshes, although a "tighter" coupling described in the subsequent section would be a more appropriate strategy.

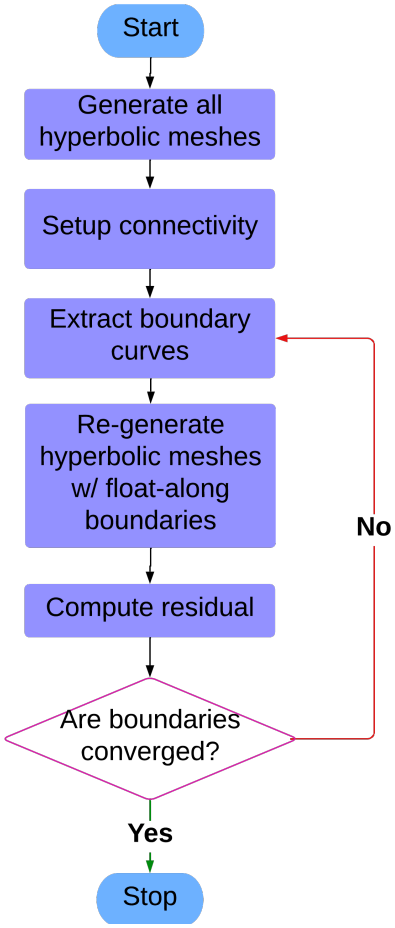
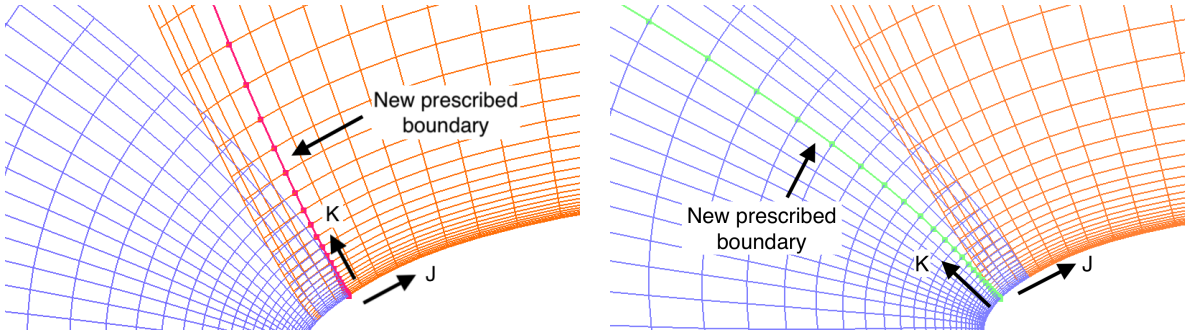


Figure 4.7: Flow chart for loose-coupling algorithm.

To apply the coupling approach, all meshes are initially generated using splay boundaries at the coupled boundaries to produce the "initial guess". Using the connectivity information, curves along the marching direction (K index for field/surface meshes, L for volumes) are extracted from neighboring grids. The extracted curves are then applied as "float-along" boundaries as the meshes are re-generated. The extraction and boundary forcing steps are then repeated until convergence is achieved, where the coupled boundaries are no longer varying. Figure 4.7 shows the key steps and structure of the loose coupling algorithm. Figure 4.8 shows an example of the boundary exchange that occurs between two coupled meshes. For each boundary of a given mesh, an interior curve from a neighboring mesh is extracted and applied as a new forced boundary condition before the mesh is re-generated.



(a) (b)

Figure 4.8: New prescribed boundaries for overlapping coupled meshes. (a) Boundary curve for blue mesh (magenta) extracted from orange mesh. (b) Boundary curve for orange mesh (green) extracted from blue mesh.

To apply the float-along boundary conditions, the magnitude of the step size is extrapolated from the interior, while the marching direction is generally prescribed by the boundary curve. This is done to respect the wave propagation from the eigenvalue analysis of the "flux" terms of surface meshing equations. In the case of volume meshes, a similar approach is taken through a one-dimensional extrapolation in the direction normal to the particular boundary edge.

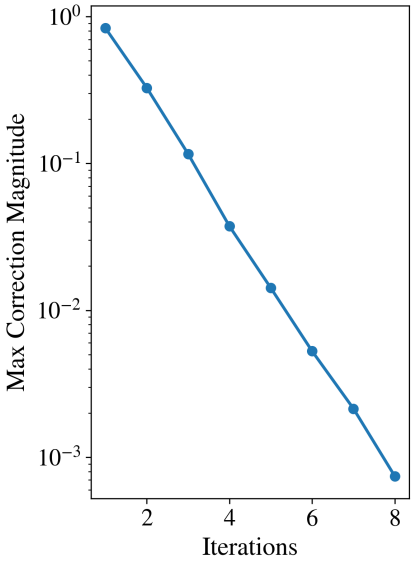
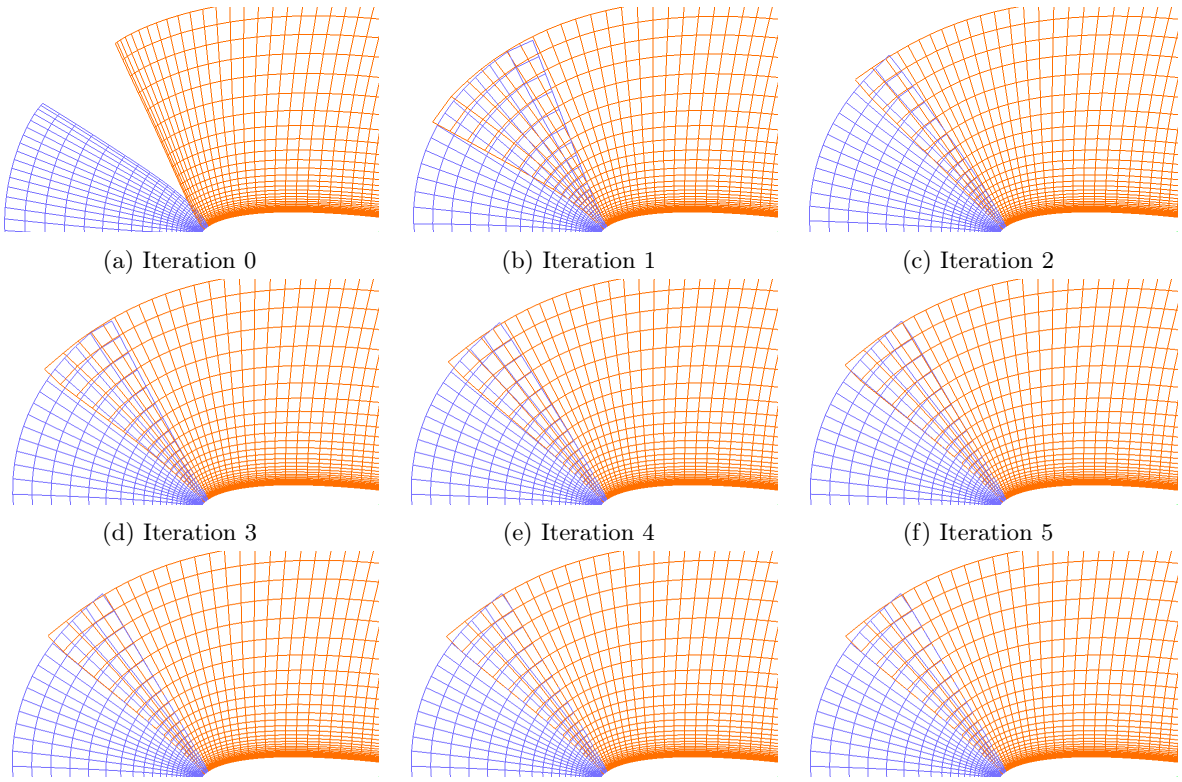


Figure 4.9: Convergence history of coupled blue and orange grids in Figure 4.10.

To illustrate the approach, we will consider a pair of overlapping initial curves for generating field meshes that start with sufficient overlap. In this case, these two grids have a four-cell overlap with two fringe points. Figure 4.9 shows the convergence history of the max displacement at each iteration for all boundary points. The convergence appears linear, which shows the convergence rate is similar to that of fixed-point iteration. Figure 4.10 shows the grids generated at each iteration until convergence is reached at the eighth iteration. The initial curves lie in the lower right corner of each frame where there is a tightly-clustered spacing in the normal direction, and the grids are grown outward to the left and upward.

In the initial iteration (fig. 4.10a), the grids are simply generated with simple extrapolation boundaries, immediately demonstrating the grid divergence problem and the lack of maintained overlap between the meshes. The grids are then iteratively re-generated, with boundary curves from the interior of neighboring grids exchanged. Eventually, convergence is achieved (Figure 4.10i), resulting in overlap consistently maintained across the entire marching direction.



(a) Iteration 0 (b) Iteration 1 (c) Iteration 2

(d) Iteration 3 (e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (h) Iteration 7 (i) Iteration 8

Figure 4.10: Evolution of boundaries for overlapping grids in the loose-coupling approach.

4.3.4 Tight-Coupling Approach

A straight-forward implementation that resembles the approach commonly used in CFD solvers is to apply the coupling at the solve-step at each subsequent grid level or step. In this approach, all coupled grids are advanced simultaneously, and the boundary data exchange occurs at the linear-solve step instead of after generating the meshes completely. The boundary exchange is iterated several times before the current step is completed and the next step can be evaluated. This is effectively a lagging approach to exchange the boundary data between all grid blocks occurring at each grid step. Furthermore, this approach is considered a "tighter" coupling than the aforementioned loose-coupling approach due to the convergence of the boundaries occurring simultaneously and globally at each grid step.

To apply this particular coupling approach, all meshes first set up the data to solve for the $L+1$ grid level. The meshes here are typically solved implicitly using a Beam and Warming linearization of the system [27], so here the solve is considered in terms of assembling a linear system. The coupling is incorporated into the system simply through Dirichlet boundary conditions. The right-hand side is the displacement vector of the coupled boundary point to advance to the next grid level at $L+1$. An initial solve is performed, where the right-hand side of the coupled boundaries are defined by the surface-normal approximation derived from a one-sided differencing of the local parameterization in the parent mesh. The

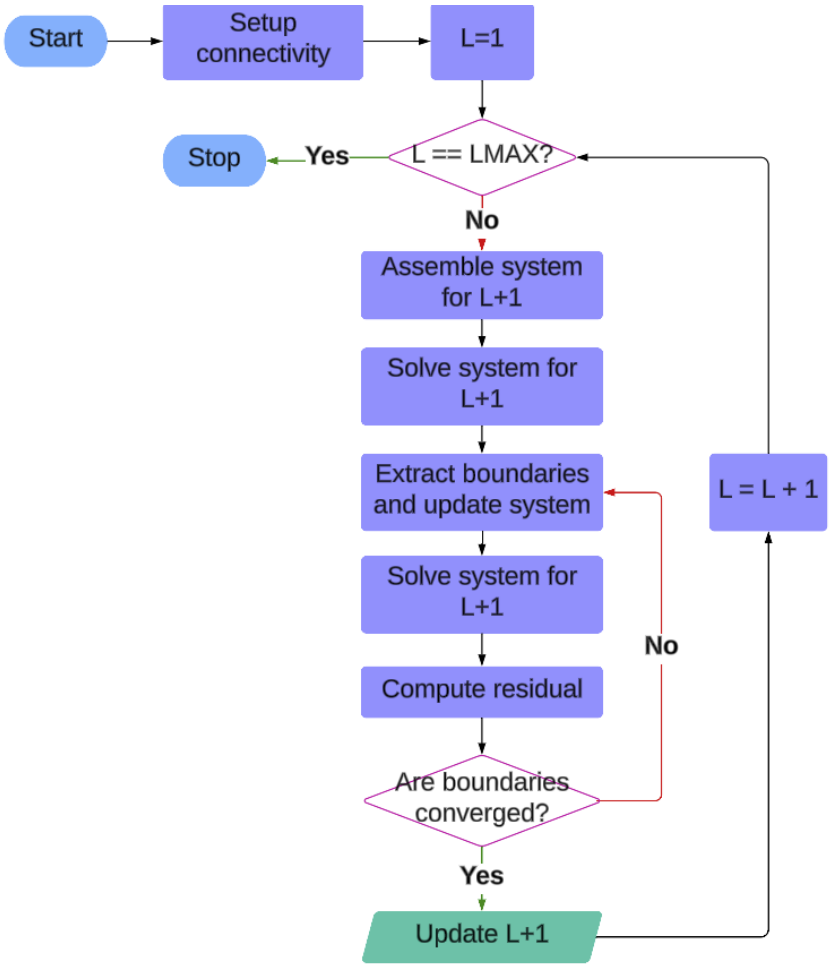


Figure 4.11: Flow chart for tight-coupling algorithm.

magnitude is also scaled by the prescribed step-size to ensure consistency with the interior points. With an initial guess generated for $L+1$, the boundary exchange can occur. Using the domain connectivity information, a displacement vector from L to $L+1$ is interpolated from neighboring grids and replaces the right-hand side for each coupled point. The system is then solved again to update the $L+1$ grid. The boundary extraction and update of the $L+1$ level is iterated until the displacement of the $L+1$ point at each coupled boundary point falls below some specified tolerance. The meshes then continue onto generating $L+2$, where the process is repeated and the coupling is applied. The coupling is performed at all grid levels until the meshes are generated. Figure 4.11 illustrates the coupling implementation through a flow chart.

In the automatic meshing approach, this coupling scheme is more amenable with volume meshing than LCBC. A common issue with LCBC in volume meshing is poor mesh quality in the intermediate iterations. While this is not immediately disqualifying for LCBC, it can result in more iterations needed to eliminate the poor mesh quality in addition to the residual "error" in the local grid geometry. Additionally,

the poor mesh quality can produce spurious interior grid behavior, which can further delay or prevent convergence. On the other hand, the convergence should be faster for a tight-coupling approach due to decreased dependency on the geometry of the exchanged information.

4.4 Applications

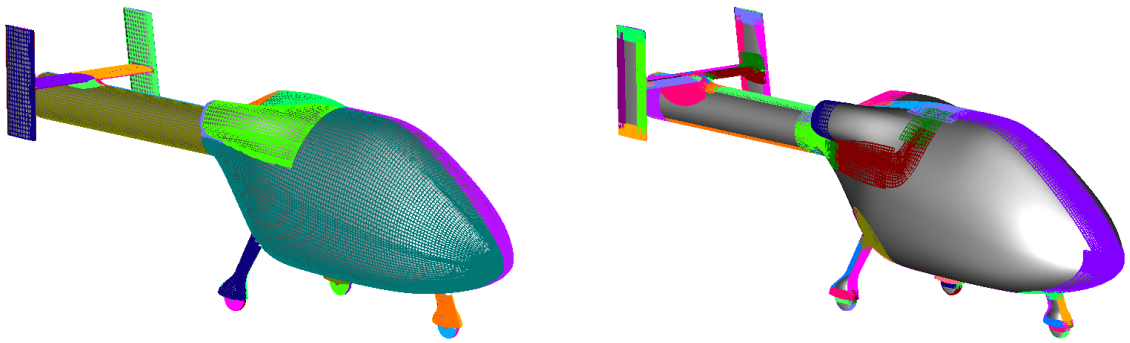
Various applications of the boundary coupling schemes for hyperbolic meshing on complex geometries are presented in this section. All examples are shown here are in the context of the automatic overset mesh generation framework. However, it should be emphasized again that the boundary conditions can function in other contexts where hyperbolic mesh generation is used for structured overset mesh generation. The first set of examples demonstrates the LCBC on surface meshes. The second set illustrates cases for volume meshing, where both loose and tight coupling applications are demonstrated.

4.4.1 Surface Meshing

4.4.1.1 Quiet Single Main Rotor

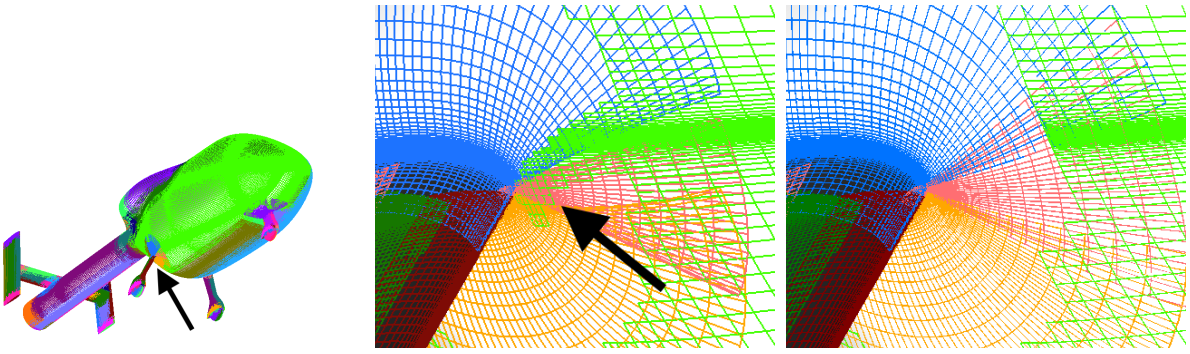
The first example is the Quiet Single Main-Rotor Helicopter (QSMR) [105], which is a concept vehicle from NASA's Revolutionary Vertical Lift Technology (RVLT) project. The original BRep geometry consists of 42 faces, 91 edges, and 51 nodes, and the aerodynamic components have zero-thickness trailing edges. A total of 179 meshes are generated, with 42 face meshes (see Figure 4.12a), 137 edge/node meshes (see Figure 4.12b), and 394012 points in total on the surface. The face mesh generation takes approximately 130 seconds, while the node/edge mesh generation and surface mesh connectivity steps take approximately 44 seconds. This surface mesh was generated on a 2019 MacBook Pro with a 2.6 GHz 6-Core i7 processor.

Coupling the boundaries for the surface meshes is a relatively inexpensive process. The wall clock time for generating the surface meshes and performing connectivity does not differ significantly with the use of LCBC. Without using the node and edge mesh coupling, a complete surface mesh with surface connectivity can be generated in 37 seconds. Incorporating the loose coupling only takes 18 iterations



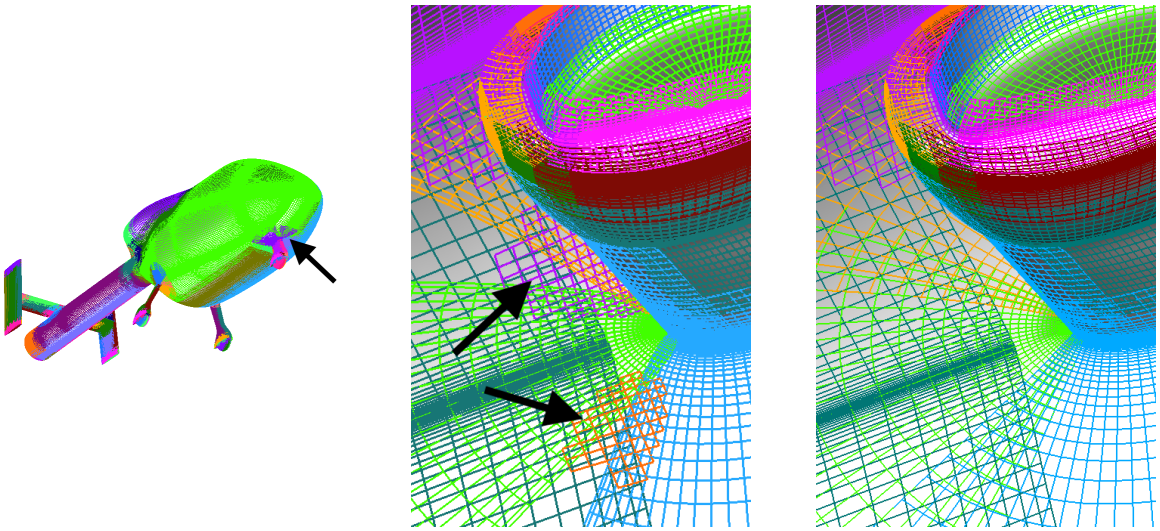
(a) (b)

Figure 4.12: Surface meshes generated for QSMR decomposed into (a) face meshes and (b) edge and node meshes.



(a) (b) (c)

Figure 4.13: Surface meshes generated at fuselage and rear right landing gear fairing junction for QSMR. (a) Full surface mesh system. (b) Collar meshes (blue, red, gold) generated without LCBC. (c) Collar meshes generated with LCBC. Note the face mesh (light green) attempting to fill in gaps between hyperbolic collar meshes when generated without LCBC.



(a) (b) (c)

Figure 4.14: Surface mesh system generated at front landing gear fairing and fuselage junction. (a) Full surface mesh system. (b) Without LCBC. (c) With LCBC. Unnecessary face mesh cells (orange, purple) are removed after coupling eliminates gaps.

and increases the wall clock time by approximately 7 seconds.

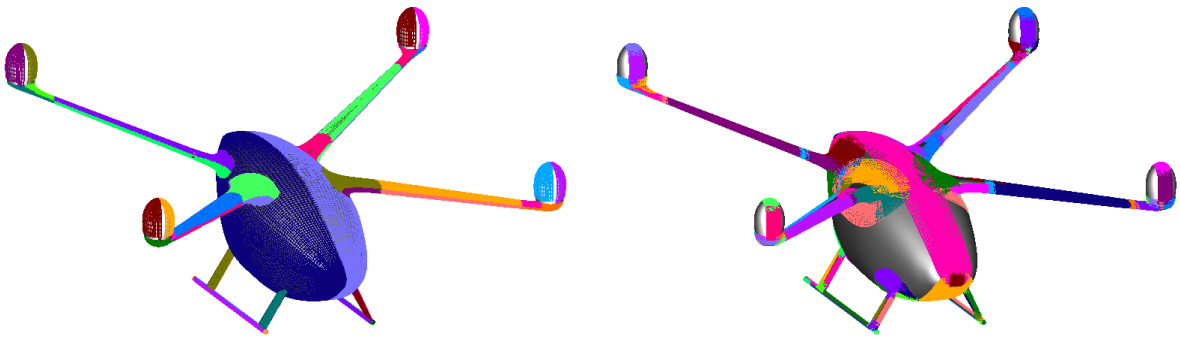
Figure 4.13 shows the effect of introducing the LCBC to the junction between the fuselage and rear right fairing of the landing gear. Before incorporating the coupling, the hyperbolic meshes (shown in blue and red) develop a gap between them as the meshes are marched outwards from the junction curve (see Figure 4.13b). As a result, the face mesh cells (shown in green) must remain unblanked in the gaps to maintain overlap between the meshes. While no orphan points are generated from the attempted fill-in, the grid resolution consistency is negatively affected near the junction edge. The hyperbolic node and edge meshes exhibit a finer spacing near the edge of the junction, while the face mesh places much coarser

spacing in the region. Once the LCBC is applied, the overlap between the hyperbolic grids is improved, blanking out the face mesh points near the junction edge (see Figure 4.13b). The overlap between the node and edge meshes remains in the far boundaries of the meshes.

Other regions also benefit from the improved resolution consistency due to the loose coupling. In Figure 4.14b, the lack of coupling results in a similar situation in the junction of the fuselage and front landing gear. Large gaps appear between the node and edge meshes (shown in orange, green, and blue) on the fuselage surface. Some face mesh cells need to remain unblanked in the region to cover the gaps, resulting in small, isolated patches of face meshes (purple, orange). Although there are no orphan points, the variations in size of overlapping cells are non-ideal for interpolation of flow field variables. In Figure 4.14c, applying the coupling gives the edge meshes optimal overlap, and is able to prevent the connectivity routines from leaving behind isolated face mesh cells.

4.4.1.2 Six-Passenger Quadrotor

The second test case is the six-passenger Quadrotor (6-Pax Quad) from the NASA RVLT project [106]. The BRep geometry consists of 56 faces, 139 edges, and 71 nodes. This geometry exhibits high curvature, convex features, which can demonstrate the effectiveness of the LCBC method for both surface and volume meshes. The automatic scheme generated an overset surface mesh system with a total of 224 surface meshes (see Figure 4.15) and 461619 points. The face mesh generation procedure takes approximately 157 seconds to generate 56 face meshes. The edge/node meshes and surface mesh connectivity steps are then performed, which takes approximately 61 seconds of wall clock time to create 168 edge/node meshes with 15 iterations for the LCBC. The total wall clock times for generating edge/node meshes and performing surface mesh connectivity with and without LCBC only differ by approximately 4 seconds. The total run time without LCBC takes approximately 56 seconds. The result of the extra effort is an improvement of interpolation quality in the overlap, as well as a reduction in the final orphan point count. Without LCBC, the surface mesh exhibits 52 orphan points, while generating the mesh with



(a) (b)

Figure 4.15: Surface meshes generated for 6-Pax Quad. The surface is decomposed into (a) face meshes and (b) edge and node meshes.

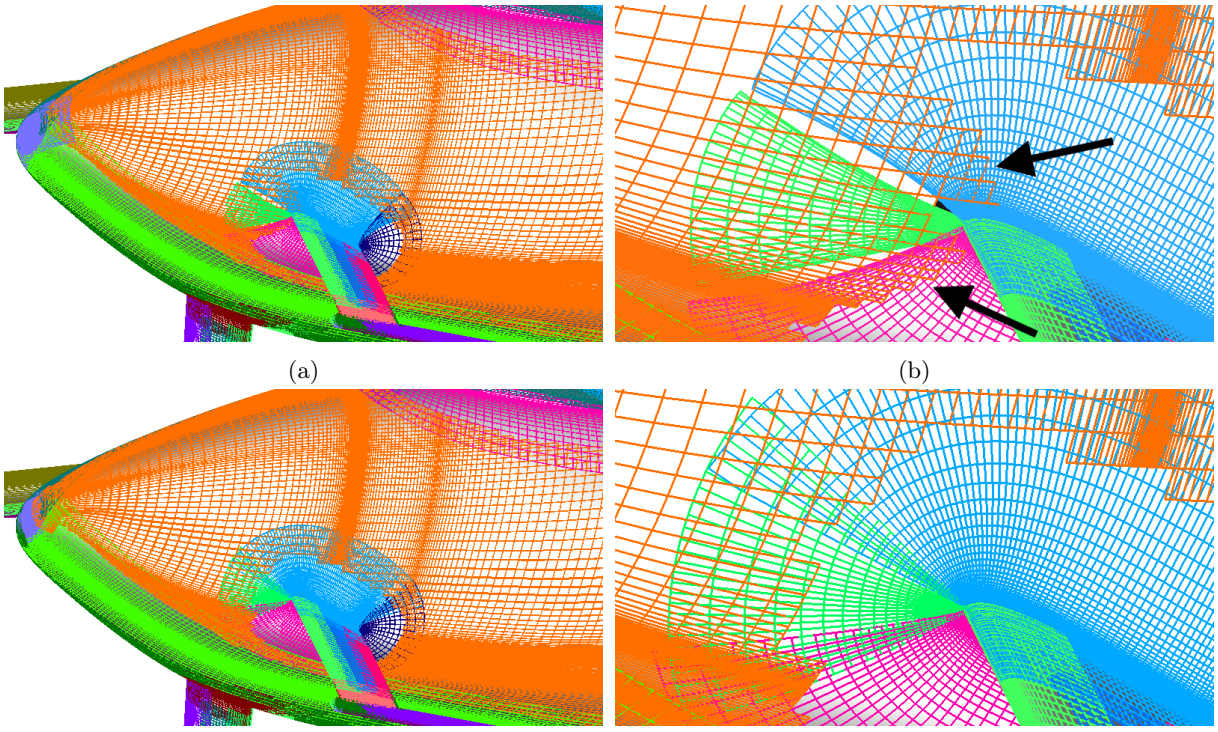
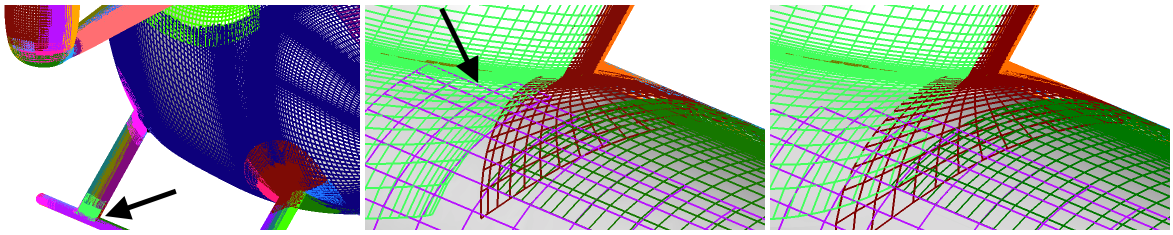


Figure 4.16: Comparison of meshes generated at strut-fuselage junction for 6-Pax Quadrotor. (a) Full view without LCBC. (b) Close-up strut-fuselage junction leading edge region without LCBC. (c) Full view with LCBC. (d) Close-up of strut-fuselage junction leading edge region with LCBC.

LCBC results in 43 orphan points. While this is only a modest reduction of orphan points, improvements to the inter-grid communication can be observed upon closer examination of the surface mesh. It is also possibly the case that a lower-quality inter-grid communication may not necessarily result in a significant increase in orphan points.

One region that demonstrates how the LCBC improves overlap is the fuselage-landing strut junction (Figure 4.16). In Figure 4.16a, large gaps develop between the edge meshes at the leading edge due to the use of extrapolation boundary conditions. A closer look in the region in Figure 4.16b reveals that, in an attempt to fill the gaps, the automatic approach left the face mesh (orange) unblanked. The unfortunate consequence is that the coarse face mesh extends into regions where the edge/node meshes (red, green, blue) have tight spacing in order to maintain sufficient overlap between the face and edge/node meshes. When computing the flow solution, this is not ideal, since the coarse face mesh is likely not suited for the proper resolution of the fine gradients in that region. Additionally, the coarser spacing introduces additional difficulty in maintaining the overlap when gaps occur in very tight regions. Figure 4.16b shows that the coarse spacing of the face mesh near the leading edge of the strut is unable to fully patch over the gaps and provide suitable donor stencils for the resulting fringe points in the region. The situation shows marked improvements when the LCBC is applied, as shown in Figure 4.16c and Figure 4.16d. The



(a) (b) (c)

Figure 4.17: Edge meshes generated at junction of right rear landing strut and skid for 6-Pax Quadrotor. (a) Full view of skid and strut region. (b) Zoomed-in view without LCBC, where purple face mesh cells remain close to strut/skid intersection region. (c) Zoomed-in view with LCBC, where purple face mesh cells are retracted from strut/skid intersection region.

edge meshes now have sufficient overlap between the meshes, resulting in the removal of the gaps between them. Because the gaps are closed, the hole-cutting routines do not need to leave the face mesh unblanked in the overlap regions to maintain connectivity. As a result, the face meshes only overlap with the edge meshes at the far boundary, where the grid spacing between the meshes are closer in size. Furthermore, overlap is able to be maintained between cells of similar size and orientation.

Figure 4.17 shows another similar situation occurring at the junctions between the skids and strut fairings. While orphan points do not appear from the lack of overlap, the face mesh (purple) can be seen attempting to fill the gaps in Figure 4.17b by extending deeper into the edge/node meshes (lime-green, brown). Some fringe points of the face mesh extend all the way to the initial curve shared by the edge/node meshes, where the local cell spacing is much smaller than that of the face mesh. When proper overlap between the edge/node meshes is maintained (shown in Figure 4.17c), the overlap with the face mesh remains largely in the outer portions of the edge meshes, where the grid spacing between the overlapping meshes are closer in scale.

4.4.2 Volume Meshing with Loose-Coupling: Juncture Flow Model with F6 Wing

This example applies volume meshing with LCBC to the Juncture Flow Model with F6 wing (JFM-F6) [107]. These meshes originate from a flow solution comparison and replication study [108] of the original work performed by Lee and Pulliam [87] to demonstrate the feasibility of using the EPOGs approach for CFD analysis of complex geometries. A grid sensitivity study [108] was performed where meshes at four levels of grid refinement (coarse, medium, fine, extra-fine) were generated using the auto-meshing approach. The input parameters at each grid

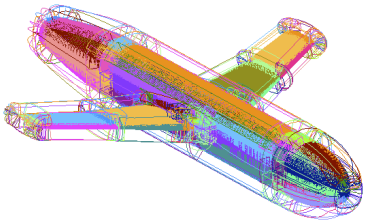


Figure 4.18: Near-body volume meshes generated by EPOGs on the JFM-F6.

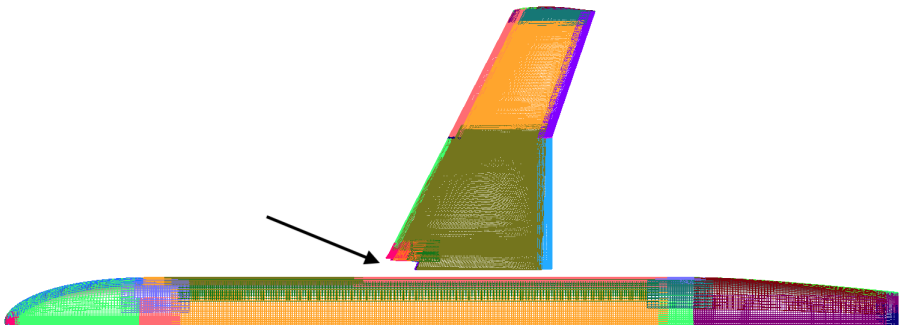


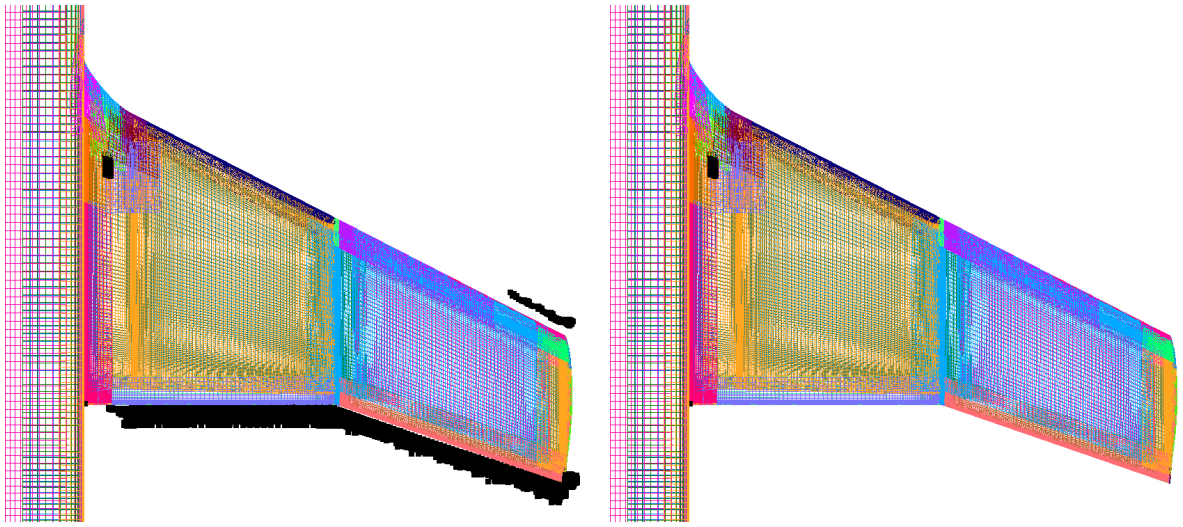
Figure 4.19: Initial surfaces coupled in volume-mesh LCBC procedure, with indicated gap due to collar meshes not included in volume mesh coupling.

level were selected such that the local surface resolution and total grid-size correlated with the resolution and grid-sizing used in the original study by Lee and Pulliam [87]. In all cases, the volume meshes were generated with LCBC to ensure sufficient overlap between near-body meshes. Figure 4.18 shows the geometry and the resulting near-body volume meshes generated by EPOGs. Figure 4.19 shows the starboard side of the initial surfaces that participated in the LCBC procedure, with a mirrored configuration of initial surfaces across the center $Y = 0$ plane. A gap is present between the main wing and the fuselage, which is a result of not including the collar meshes that represent the wing-root junction. These meshes are concave and thus are not amenable to the coupling procedure.

Table 4.1 illustrates the performance of the loose-coupling approach. These grids were generated on a 2019 MacBook Pro with a 2.6 GHz 6-Core i7 processor using 10 OpenMP threads. Here, the grid quality is measured through orphan point count, while the computational overhead here is measured in the wall-clock time, iterations, and final convergence residuals. It should be noted that these grids were generated before it was found that incorporating ghost meshes was good practice, so the final residuals of the boundaries and computational overhead may be higher than what can currently be achieved.

Table 4.1: Grid quality and mesh generation performance for JFM-F6 at four levels of refinement using EPOGs.

	Coarse	**Medium**	**Fine**	**Extra-Fine**
No. Near-Body Grids	174	181	187	185
N_p surface, Near-Body	337K	570K	1.40M	2.43M
N_p volume, Near-Body	17.2M	37.6M	130M	282M
Orphan Pts, No LCBC	40887	50416	59665	1016
Orphan Pts w/ LCBC	3740	8172	2120	670
Wall Time	22.3 min	39.3 min	2 hr 48 min	18 hr 24 min
Residual	8.17E-04	7.70E-04	8.99E-04	7.33E-01
Iterations	146	124	150	250



(a) (b)

Figure 4.20: Orphan points (black) detected by Overflow DCF [28] for JFM-F6 near-body volume meshes generated with EPOGs using (a) only splay boundaries and (b) LCBC.

Additionally, grid quality checks were also performed at each iteration, which also adds additional wall-clock time that scales with the mesh size.

From the table, it can be observed that applying the LCBC approach overwhelmingly results in reductions in orphan point count. Across all grid levels, the reduction in orphan points from the application of LCBC range from 34 to 90%. Additionally, the total run time appears to scale roughly with the mesh size with exception to the Extra-Fine case. In the case of the Extra-Fine, the wall time appears to also scale with both the grid size and with the increased quantity of iterations. The Extra-Fine case also appears to have poorer convergence behavior in comparison to the coarser meshes, which satisfy the minimum criteria of the residual dropping three orders of magnitude. This turned out to be caused by a single grid that stalled the convergence behavior, whereas other grids within the same case were able to satisfy the convergence criteria for loose-coupling. In most situations, this mesh may still be acceptable for use in CFD, but in worst-case scenarios the stalled grid may fall under the list of meshes that require manual repair before use.

A closer look at the actual meshes reveals where most of the orphan points are eliminated. Most of the orphan points are clustered along the finite-thickness trailing edge, outboard leading edge, and tip edge of the main wing as shown in Figure 4.20. These regions are highly convex, resulting in significantly poor overlap from the splay boundaries. The leading edge does not exhibit any orphan points because there is only a single CAD feature edge, meaning there can only be one edge in this region. The finite-thickness trailing edge, however, has a feature edge for where the upper and lower surfaces meet the trailing-edge face. Thus, the region is split into potentially two to three separate meshes (one for each feature edge, and

one for the face)[1]. Conversely, the meshes generated with loose-coupling (fig. 4.20b) show significantly fewer orphan points in similar regions due to the preservation of the existing overlap in the surface mesh. Examination of the trailing edge mesh geometry reveals the cause for the large presence of orphan points. Figure 4.21 shows slices of the volume mesh generated with splay boundaries along the span of the wing, as well as close-ups of the trailing edge. In the trailing edge region, the meshes are able to maintain overlap some distance away from the wall before the mesh boundaries pull away from each other, leaving behind a large gap just aft of the trailing edge region. Where this divergence occurs is too close to the geometry surface for the off-body mesh to cover the region while simultaneously maintaining the proper overlap. Thus, the fringe points are then marked as orphan points due to no available grid to interpolate from.

However, once the loose-coupling is applied, the resulting gap is closed and the overlap in J and K along the wall-normal direction is improved. Figure 4.22 shows the effect of the loose-coupling on the volume meshes at trailing edge along the span of the wing. A zoomed view of the trailing edge region shows that the overlap at the geometry surface is able to be consistently maintained in the wall-normal direction (see Figures 4.22c and 4.22d). The boundaries no longer diverge away from each other, but instead are

[1]EPOGs can now detect and build a single mesh similar to a leading-edge mesh that captures the entire finite-thickness trailing edge, but this capability was not yet available at the time of the study.

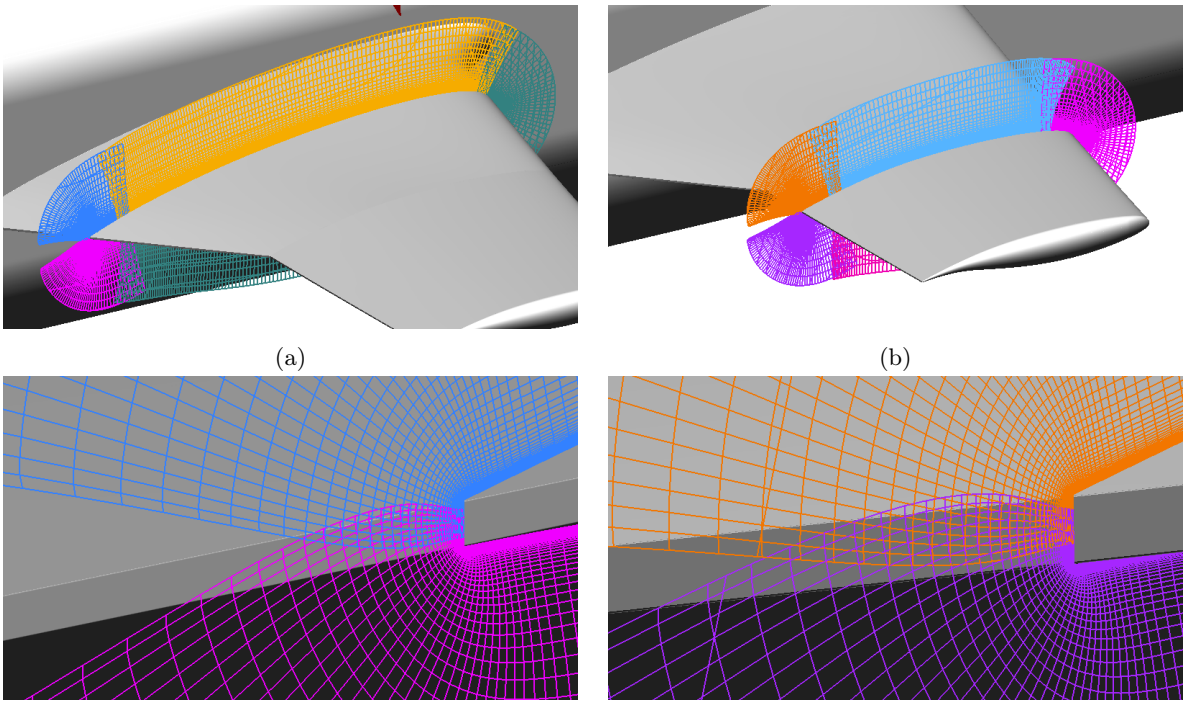


Figure 4.21: Constant Y-plane cuts of volume meshes generated with splay boundaries on JFM-F6 wing. (a) Cut at $y = 550$. (b) Cut at $y = 1150$. (c) Zoomed view of trailing edge at $y = 550$. (d) Zoomed view of trailing edge at $y = 1150$.

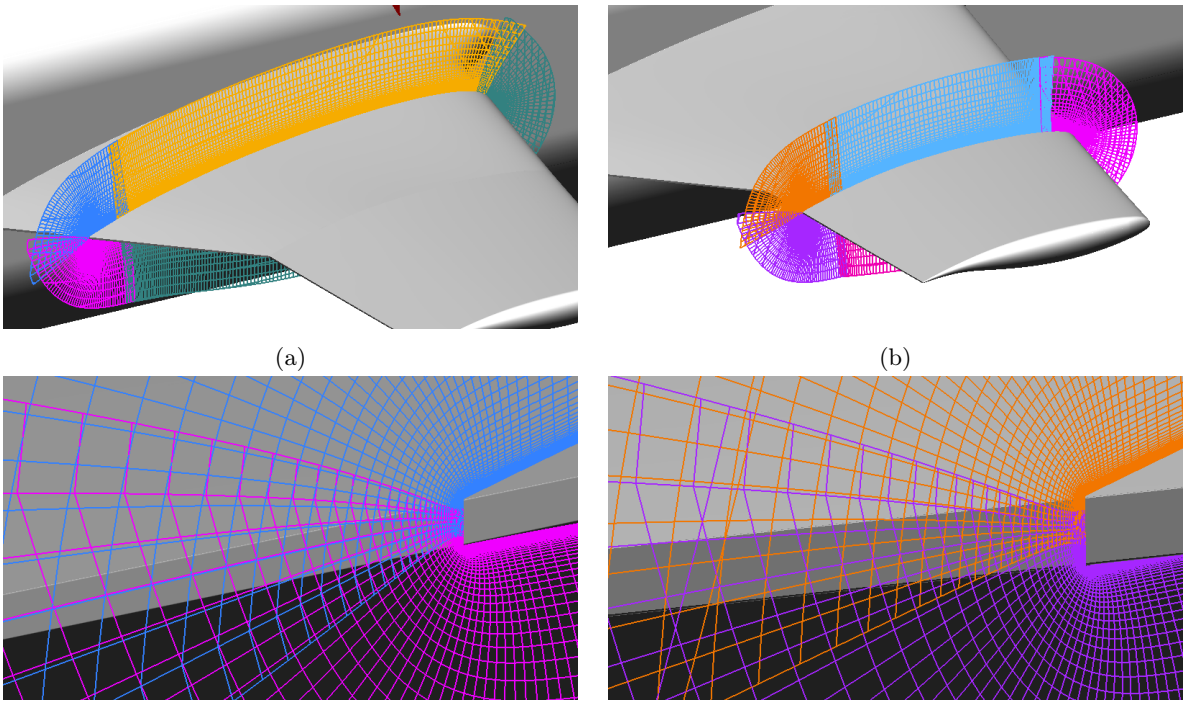
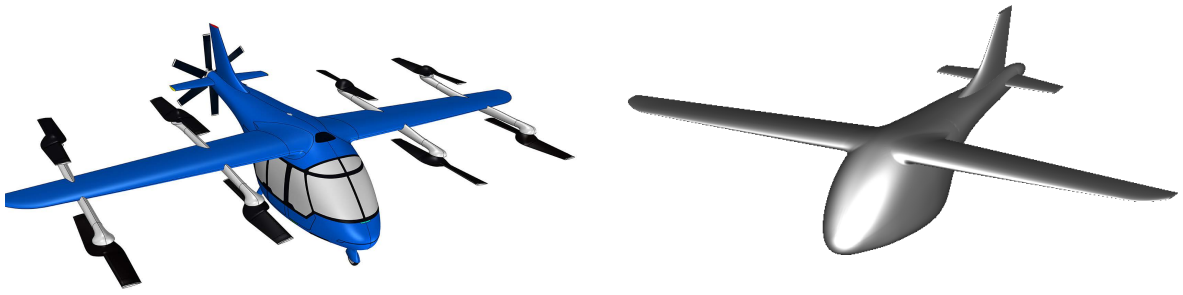


Figure 4.22: Constant Y-plane cuts of volume meshes generated with LCBC on JFM-F6 wing. (a) Cut at $y = 550$. (b) Cut at $y = 1150$. (c) Zoomed view of trailing edge at $y = 550$. (d) Zoomed view of trailing edge at $y = 1150$.

able to be marched away from the interior of the grids and into neighboring grids. An expanded view (see Figures 4.22a and 4.22b) shows that the overlap is also maintained out to the outer boundary of the volume mesh. Thus, the fringe points for the meshes at the trailing edge are able to find corresponding donor mesh interpolation stencils, which eliminate them from the list of orphan points. This results in the significant orphan-point reduction shown in Table 4.1 once the coupling is applied. This also alleviates the burden on off-body meshes in maintaining overlap and connectivity on all meshes. In the case of the splay boundaries, a user or algorithm would potentially have to tune the hole-cutting and off-body mesh generation to cover the gap shown in Figures 4.21c and 4.21d. This ultimately increases the complexity of both the hole-cutting and off-body mesh generation process in order to obtain high-quality connectivity with matching resolutions. With the coupling, such processes may not require as detailed and precise attention.

Other grids on the wing surface also show a slight reduction in overlap in comparison to the splay boundaries. However, because of the way the coupling approach operates, the overlap at each grid level is reduced to exhibit the same overlap as that of the surface. Thus, as long as the surface mesh connectivity is of sufficient quality, the volume mesh too will exhibit connectivity of similar quality when the coupling is applied.



(a) (b)

Figure 4.23: Lift+Cruise concept vehicle. (a) Full geometry with rotors and propeller. (b) Wing-body geometry.

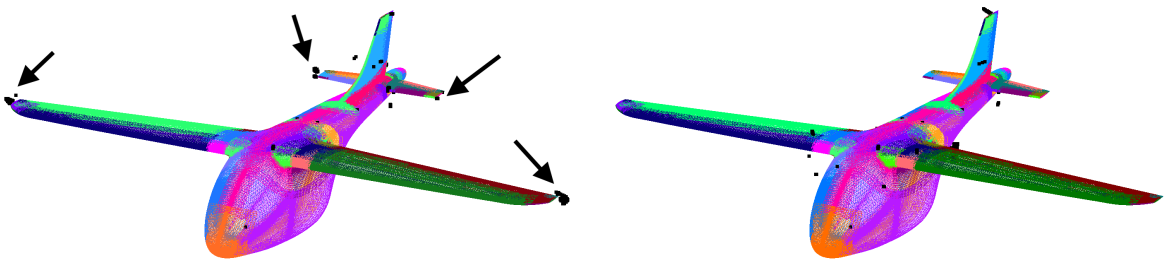
4.4.3 Volume Meshing with Tight-Coupling: Lift+Cruise Concept Vehicle

The following example involves volume meshing with the tight-coupling approach applied to the RVLT Lift+Cruise (L+C) concept vehicle [106] using the automatic meshing tools. This vehicle is a representative geometry of a certain class of Urban Air Mobility (UAM) vehicles currently under development with separate propulsion components for the different stages of flight. Pylon-mounted rotors on the wings provide thrust (or "lift") for vertical take-off and landing (VTOL), and a rear-mounted propeller provides thrust for forward flight, or "cruise". However, for demonstrating the tight-coupling approach, only the simplified wing-body-tail geometry is considered without other moving or static components. Figure 4.23 shows the full geometry of the vehicle and the actual meshed geometry. Note the lack of pylons and landing gears in Figure 4.23b.

Table 4.2 shows the performance statistics of the mesh generated with tight-coupling. Here, the meshes are generated with a max stretching ratio of 1.13, max turning angle of 3.0°, a max edge spacing

Table 4.2: Grid quality and mesh generation performance for Lift+Cruise Wing-Body-Tail using EPOGS.

Parameter	**Quantity**
No. Near-Body Grids	96
N_p surface, Near-Body	335K
N_p volume, Near-Body	33.8M
Orphan Pts, No Tight-Coupling	2002
Orphan Pts w/ Tight-Coupling	1860
Total Wall Time (sec)	188
Coupling Wall Time (sec)	110
Residual	1.382E-12
Iterations per Grid Level	15



(a) (b)

Figure 4.25: Orphan points (indicated in black) found on near-body meshes. (a) Volume mesh generated with splay. (b) Volume mesh generated with tight-coupling.

of 0.33 ft, and viscous wall spacing of 4.10×10^{-6} ft. From the table, it can be observed that the total wall time for performing surface and volume mesh generation takes approximately 3 minutes. For comparison, a similar run on a 2019 MacBook with Intel i7 processor using the loose coupling on the same geometry [109] required approximately 1 hour and 150 iterations to perform the loose-coupling. Several factors are likely contributing to the speed-up here. First, the use of the tight-coupling also enables convergence rates similar to that of fixed-point iteration. Because the convergence is applied at the linear solve level and is not as dependent on the geometric or grid quality of the exchanged information, a linear convergence behavior can be achieved. Figure 4.24 shows the convergence behavior at the final grid level computation. The linear convergence also enables less iterations to be taken, and here we can observe that the tight-coupling allows for a factor of ten reduction in iterations. Furthermore, grid quality checks are not occurring intermittently through the iterations as it is done in the loose-coupling procedure, which also further reduces the additional overhead of each iteration. Finally, this example was performed on an Apple Silicon M3 chip, which is a more powerful chip than the 2019 Intel i7 chip. This likely factored into additional, albeit minor, additional speed-up in the run.

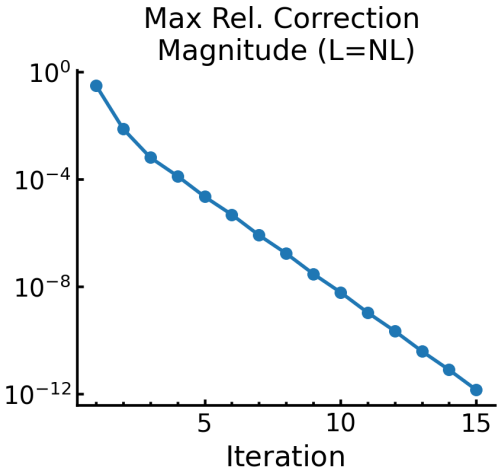


Figure 4.24: Residual history of tightly-coupled boundaries for Lift+Cruise volume meshes.

The table also shows a reduction in orphan points after the tight-coupling is applied. Similar to the loose-coupling approach, the tight-coupling also eliminates orphan points that can appear from poor connectivity from splay boundaries. Figure 4.25 shows the orphan points that appear from the domain connectivity procedure performed by the EPOGS software for volume meshes with splay and tight-coupling boundaries. For the meshes with splay boundaries (see fig. 4.25a), small pockets of orphan

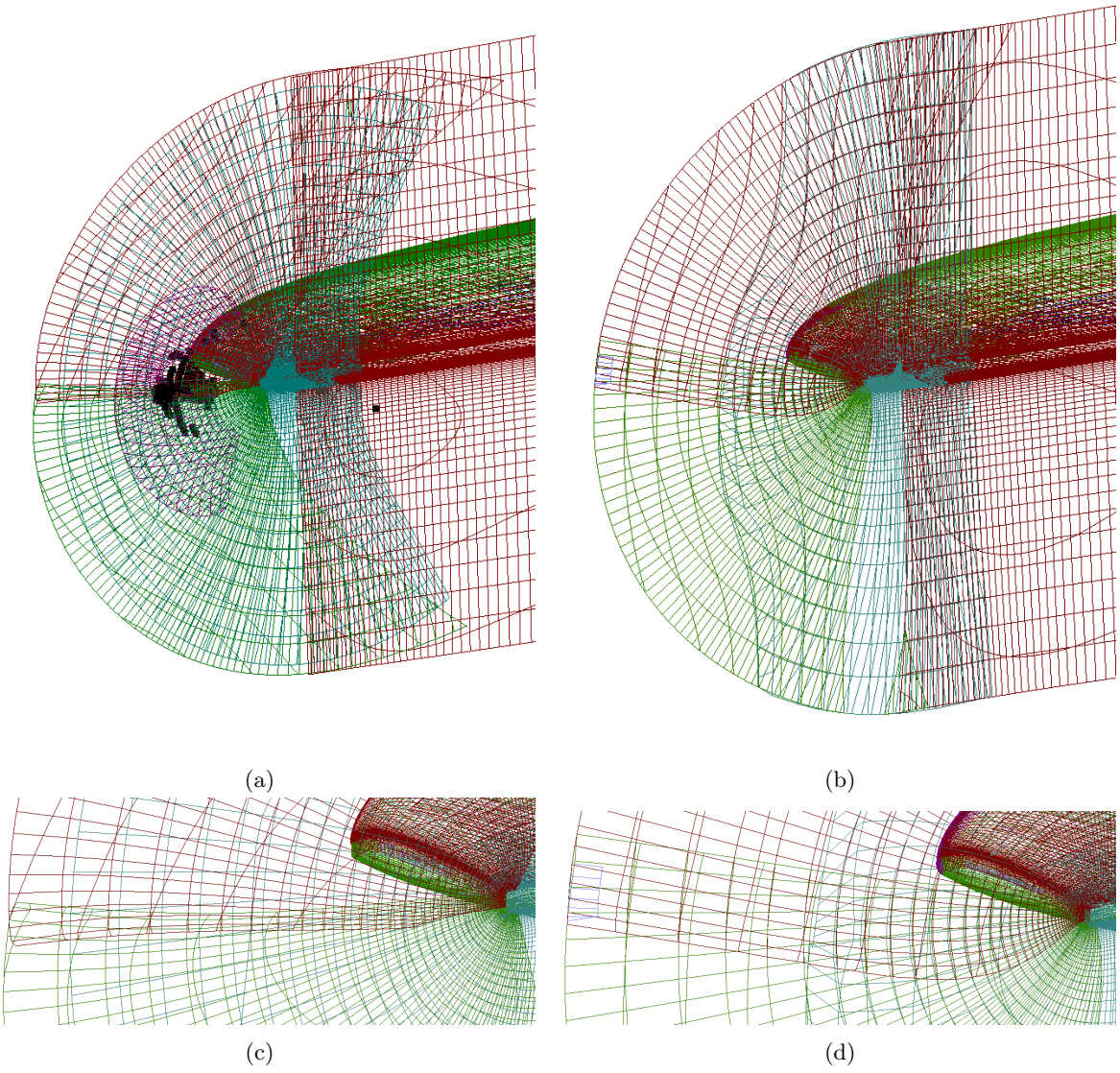


Figure 4.26: Orphan points (indicated in black) found on near-body meshes. (a) Volume mesh generated with splay. (b) Volume mesh generated with tight-coupling.

points can be observed at convex regions, like at the tips of the main wing or tail surfaces. Because this geometry only has sharp trailing edges, orphan points only appear in small clusters instead of in large clusters in the wake aft of the trailing edges. When the tight-coupling is applied, such pockets no longer appear. This is demonstrated by the lack of orphan-point clusters that appear at the wing tips (see fig. 4.25b).

Figure 4.26 provides a closer look at the orphan point cluster that occurs in the left tip of the main wing as shown in Figure 4.25a. Along the span of the wing, a constant X plane cut of the volume mesh at the aft quarter-chord is also shown, displaying the behavior of the volume mesh normal to the surface. The slice is also color-coded to match the corresponding surface mesh color. Figures 4.26a and 4.26b

show the larger view of the meshes surrounding the convex wing-tip region. Overall, the meshes with splay boundaries exhibits slices that overlap each other inconsistently, where some regions exhibit more than sufficient overlap while others exhibit too little overlap. The region with insufficient overlap results in the orphan points appearing at the tip. The tight-coupling meshes, on the other hand, exhibit overlap that appears to correspond to the overlap already present in the surface mesh. A zoomed view of the tip region in Figures 4.26c and 4.26d provides conclusive evidence of how the orphan points occur from the overlap. The splay boundaries in Figure 4.26c shows some reasonable overlap before the brown and green meshes mildly diverge away from each other. The resulting overlap away from the wall is now only two to three cells deep, which is insufficient for a second-order CFD scheme with a five-point support in each dimension. The tight-coupling in Figure 4.26d shows a much wider band of overlap occurring along the entire marching direction of the volume mesh, providing more than sufficient support for domain connectivity in a CFD computation.

Chapter 5

Feasibility Studies of Automatic Overset Mesh Generation for Aerodynamic Applications

The development of the automatic overset mesh generation procedure EPOGS [83] introduces a novel and systematic approach for domain decomposition of the wetted surfaces of the analyzed geometry. As previously discussed in Chapter 2, the automation scheme utilizes the topological information contained in a BRep solid. The scheme effectively produces a mesh for each topological entity (face, edge, node), where the meshes generated for edges and nodes serve to bound all the face surface meshes, resulting in an overset representation of the wetted geometry. While the resulting meshes are viable for use in most overset flow solvers, it is critical to demonstrate that the approach can function effectively as a "drop-in" replacement for conventional, manually-generated overset meshes for CFD while simultaneously reduce the manual effort required to generate such a mesh.

Such automatic meshes are typically much more "fractured" from the construction of a mesh over each face, edge, and node, resulting in increased overset boundaries compared to a typical manual meshing approach. As such, this fracturing introduces a potential unknown factor or uncertainty in computing CFD solutions. This chapter seeks to characterize the effects, if any, of the mesh fracturing in solutions of flow simulations. In particular, three aspects are considered: (i) numerical accuracy, (ii) numerical efficiency, and (iii) computational efficiency.

In the context of this effort, computational accuracy here refers to the numerical accuracy of the CFD solution itself. Of course, in the case of mesh generation and numerical simulation, the main form of error relevant to mesh generation is the discretization error. The numerical schemes in all flow solvers are intimately tied with how the flow domain is meshed or discretely represented. In addition to discretization error, "code-to-code" comparisons, as well as a "mesh-to-mesh" comparisons, are also considered in our accuracy assessment. Flow solutions generated from other solvers with differing meshing paradigms, or even with the same solver using manually-generated grids, are used to ground the flow solutions. Comparison with experimental data where available is also considered, but is only contained within the

code-to-code comparison. Since the focus is mainly on the mesh itself, only the numerical accuracy can be interrogated. Assessing the physical accuracy of the CFD solver (e.g. turbulence modeling, compressible flow phenomena, etc.), however, is largely beyond the scope of this work.

In addition to numerical accuracy, focus is also given to "numerical efficiency", or the rate at which the numerical methods can arrive at the desired solution with the respect to the size of the problem. For the most part, certain aspects of the solver may be less sensitive to the fracturing, such as the more critical lower-level computations like the setup and evaluation of the linear system. These algorithms have theoretical and practical limits based on the specific schemes' time complexity and are usually dependent on certain characteristics of the sparse linear systems rather than the purely on the grid itself. On the other hand, the higher-level numerics, such as the nonlinear pseudo-time advancement of the solution (i.e. the residual), are more likely to exhibit potential sensitivity to the actual grid itself, and thus the fracturing, due to usage of the grid for its evaluation. On a related note, convergence behavior of the integrated loads, which are typically the main quantities of interest in most engineering applications, are also dependent on the residual convergence and thus are also potentially sensitive to the fracturing. These particular aspects are investigated to evaluate whether the fracturing has a negative impact on the convergence.

Finally, a closely related factor to the "numerical efficiency" is the computational efficiency. This aspect examines the effect that fracturing may have on the actual implementation of the numerical algorithms within the flow solver. Given the increase of the overset boundaries, it is possible that there may be increased overhead from the increased quantity of communication between grids. Additionally, the smaller, more numerous grids may also affect the parallelization, or other algorithms that are intended to accelerate the flow computation. These variables are also considered in the assessment of the grid fracturing.

In addition to understanding the implications of fracturing on CFD solutions, it is also critical to characterize the time and effort savings that the automation tool provides. It is already well-known that manual overset mesh generation is a cumbersome and onerous process for real engineering applications with complex geometries. The automation scheme should demonstrate a notable reduction in effort in the critical steps of mesh generation (see Ch. 2).

This chapter is divided into three sections. The first section examines two three-dimensional, external flow benchmark cases from the NASA Langley Turbulence Modeling Resource (TMR). The main focus of this section is on the computational accuracy, since the cases are relatively small and have reference solutions from other solvers for the purpose of verification. The second section examines various wing-body and electric Vertical Take-Off and Landing (eVTOL) concepts configurations. These geometries are more complex and representative of actual aircraft. In addition to computational accuracy, the residual convergence and computational efficiency are also examined for these cases. The final section is a high-

fidelity analysis exercise of a component build-up study of a Lift+Cruise concept vehicle to demonstrate the time and effort savings that the auto-meshing tools afford for engineering workflows utilizing overset meshes.

In every flow computation presented in this chapter, care is taken to ensure that solutions are "converged" unless specified. This essentially means that solutions are iterated until the residual drops at least five orders of magnitude, the lift and pitching moment coefficients exhibit variations smaller than 0.001, and the drag coefficient varies less than a single count. This is a common practice to ensure the numerical errors are sufficiently and efficiently eliminated, since it may be computationally prohibitive to completely drive the residual numerical error to the order of machine precision.

5.1 Benchmark Cases

Two benchmark cases from the NASA Turbulence Modeling Resource (TMR) [110] are computed here to evaluate the effect of fracturing on the computational accuracy. In each case, a solid CAD model is generated using Engineering Sketch Pad (ESP) [81], from which an automatic structured overset grid is generated using EPOGS. The flow solutions are computed using Overflow and denoted as OF-EPOGS. The results are then compared with the original computations performed by Diskin et al [111], who performed verification computations of RANS solutions using several unstructured solvers (FUN3D, USM3D) and a structured multi-block solver (CFL3D). These set of results are henceforth referred to as the *reference cohort*, and their respective analysis data are available on the TMR website. In both cases, the Overflow solutions are computed using the same inviscid discretization scheme (Roe flux difference splitting [112] and third-order MUSCL scheme with Koren limiter [113]) and turbulence model (SA [90]) that was also used in the CFL3D solutions. An overview of the refinement procedure to generate a mesh family for automatically-generated grids is also discussed.

In addition to comparing the integrated loads and local flow solution profiles, uncertainty analysis [114] is also performed to fully interrogate the grid convergence behavior, and thus the numerical accuracy of the solutions. This analysis procedure is outlined in Appendix A. The convergence rates and whether the solutions are in the asymptotic range are also discussed.

5.1.1 Grid Family Generation Procedure

The typical approach when generating a mesh family is to generate the finest-level mesh, followed by successively eliminating every other grid index in each mapping direction. This effectively corresponds to coarsening by a factor of two in each dimension (factor of 8 total) between each successive grid level. However, this naive approach can be difficult to use for generating a mesh family from a fine set of automatically-generated overset meshes. Several factors complicate the process for these types of meshes. The first is the difficulty in maintaining sufficient overlap for the overset boundaries. Because the meshes are generated with optimal overlap, the first level of coarsening will not exhibit sufficient support for

fringe points. The second factor is the preservation of feature points and edges from the original CAD model. If a feature edge sits entirely on a constant grid line, it is difficult to ensure that the grid line is preserved through the coarsening without ensuring that the grid line sits on a specific index through careful setup of the grid dimensions. The feature point will need to sit on grid point that is also preserved through the levels of coarsening as well. Finally, there is currently no mechanism within the automatic meshing framework to build meshes such that it can support the necessary levels of coarsening. For example, to support three levels of coarsening, each grid dimension or grid-cell count must be divisible by four and still have sufficient resolution at the coarsest grid level. This level of control of the grid dimensions is not currently exposed to the user. As such, not all grids can be coarsened to the desired level.

An alternative approach is to successively refine a set of coarse, automatically-generated overset meshes. This approach eliminates two of the previously discussed issues. As the meshes are refined, the actual overlap region is maintained. The density of points, and thus the fringe point support, actually increases between the meshes. Additionally, it is no longer necessary to build meshes of minimum size in each dimension to support coarsening.

However, several issues remain in refining these meshes. The first issue is how to refine meshes such that the stretching is preserved. A tri-linear approach is insufficient, as it will result in divisions at the mid-points of each cell edge. Thus, internally, the stretching ratio will drop to 1 for refined cells generated from the same cells. Instead, a tri-cubic Hermite interpolation approach should be taken instead. Since the interpolation polynomial in one dimension is degree 3, or fourth-order, it can preserve the stretching behavior along the grid-indices since it will be captured in the higher derivative representation [115]. Thus, to perform the refinement, a tricubic Hermite interpolation (see Appendix B) must be constructed from the grid. Generating the interpolant only needs to be done once for the coarsest level grid. The subsequent grid levels simply only need to query the interpolant to generate meshes. Additionally, the usage of a single mapping also enables consistency in the refinement procedure since all quantities can be mapped to a common computational domain.

An additional benefit of using the spline interpolation is that the refinement is not limited to a factor of 2 in each direction. Other, smaller factors can be used, such as 1.5 or 1.26, which allow for a total increase by a factor of 3.375 and 2 between each grid level, respectively. To generate the next level of refinement, the parametric inputs simply need to linearly refined. For example, consider the original parametric ξ_0 corresponding to the grid indices in j, which is given by

$$\xi_0 = [\quad 1.0, \quad 2.0, \quad 3.0, \quad 4.0, \quad 5.0 \quad] \tag{5.1}$$

Refining by a factor of 2 can be achieved by interpolating the spline at the values in $\xi_{ref=2.0}$, which is given by,

$$\xi_{ref=2.0} = [\quad 1.0, \quad 1.5, \quad 2.0, \quad 2.5, \quad 3.0, \quad 3.5, \quad 4.0, \quad 4.5, \quad 5.0 \quad] \tag{5.2}$$

Likewise, refining by a factor of 1.5 results in parametric evaluations in $\xi_{ref=1.5}$, which are given by,

$$\xi_{ref=1.5} = [\quad 1.0, \quad 1.667, \quad 2.333, \quad 3.0, \quad 3.667, \quad 4.333, \quad 5.0 \quad] \tag{5.3}$$

However, the issue of preserving feature edges and nodes that sit explicitly on grid lines and points still remain. One approach is to utilize the domain decomposition generated by the automatic meshes. Since nearly every edge and node from the CAD is represented by a mesh, the meshes can be split into sub-blocks that are divided by the corresponding j and/or k grid indices that explicitly represent the edges and nodes. The sub-blocks can be refined individually before they are recombined into a single mesh. This helps to ensure that the feature edges and nodes are preserved explicitly on grid lines or points. Additionally, if a feature edge is a sharp edge, it avoids evaluating large jumps in the mesh gradients that can occur near the wall. This is effectively similar to applying a multi-block refinement approach for each grid that requires subdivisions. Figure 5.1 shows an edge mesh with the trailing edge explicitly represented as a grid line. To perform the refinement, the grid will be split into sub-blocks corresponding to the parts of the grid lying on the upper and lower surfaces of the wing. The refinement is applied to each sub-block separately before being joined back together into a single mesh. This helps to preserve the trailing edge grid line and avoids evaluating derivatives across the sharp trailing edge in the chord-wise direction.

Figure 5.1: Edge mesh sitting on trailing edge of wing with trailing edge indicated by purple markers.

Finally, to ensure that the interpolated values at the surface are sitting on the true surface, the newly-generated surface vertices need to be projected back onto the original geometry surface. If the geometry is analytically defined, the projection can be performed using exact derivatives. Alternatively, an extremely fine bicubic spline interpolation mapping can be utilized to project the points to the surface if only the CAD model is available.

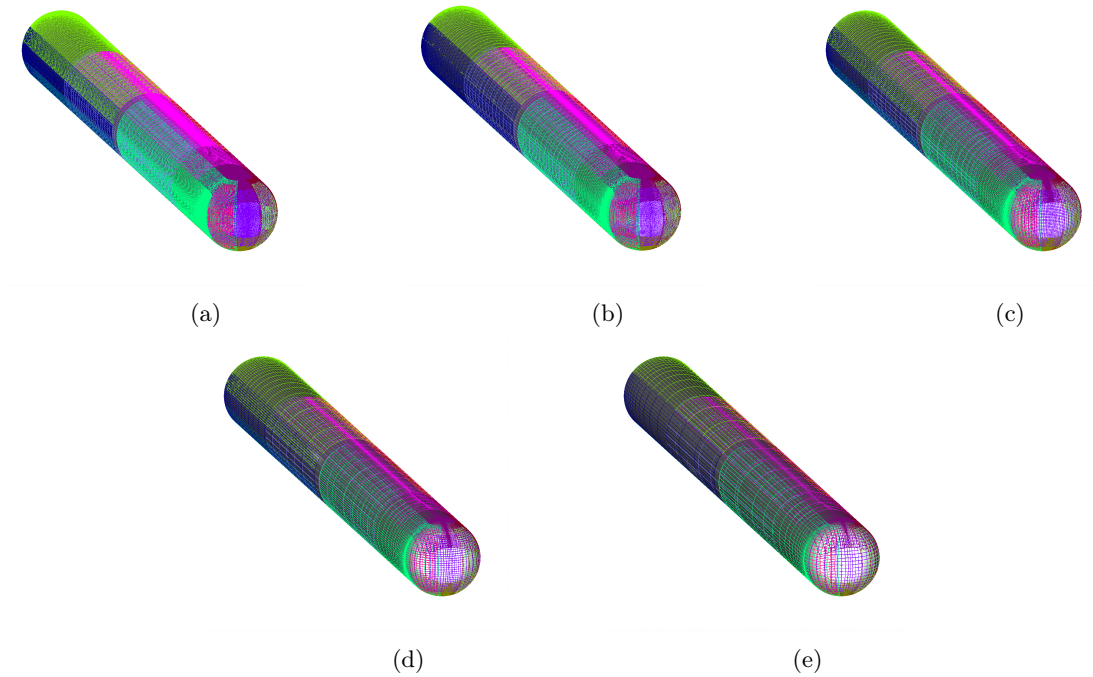
5.1.2 Hemispherical Cylinder

The first case is a hemispherical cylinder in free-air with subsonic flow at ($M_\infty = 0.6$) at a steep angle of attack ($\alpha = 19°$). This test case originates from the wind tunnel experiments performed by Hsieh [116] in 1977. The geometry consists of a cylinder of radius $r = 0.5\,in$ and length $l = 9.5\,in$ of which the forward end is capped by a hemisphere with the same radius. Note that the exit plane is coincident with the non-capped face of the cylinder geometry. The flow domain extends away from the geometry everywhere else. The exit plane also has outflow conditions applied with constant back-pressure $P/P_{ref} = 1.0$, and other external flow domain boundaries have far-field conditions applied. At the cylinder surface, standard adiabatic viscous wall conditions are applied.

Table 5.1: EPOGS input parameters used to generate the coarsest grid level for hemispherical cylinder.

Mesh Parameter	Δs_{max}	$\Delta\theta_{max}$	SR_{max}	np_{min}	Δs_{wall}	n_{fringe}
Value	$1.0in$	5.0°	1.3	15	$6.23685 \times 10^{-5} in$	2

A grid family is constructed using the procedure outlined in the previous section. First, the coarsest level is generated using the automatic meshing software. Spline refinement is then applied to create the subsequent grid levels. Table 5.1 shows the input parameters used to generate the meshes. The wall-



(a) (b) (c) (d) (e)

Figure 5.2: Surface mesh on upper surface of hemispherical cylinder at different refinement levels of EPOGS-generated mesh (Level 1 is the finest, and Level 5 is the coarsest). (a) Level 1. (b) Level 2. (c) Level 3. (d) Level 4. (e) Level 5.

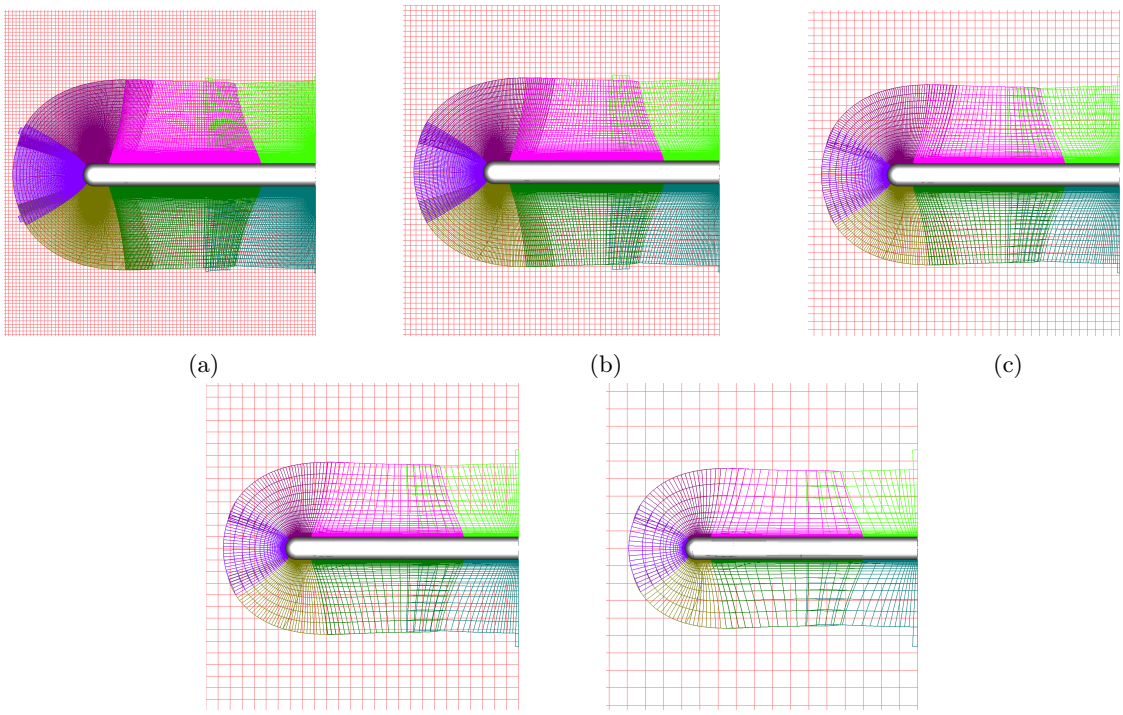


Figure 5.3: Volume mesh slice of hemispherical cylinder at $Y = 0.0$. (a) Level 1. (b) Level 2. (c) Level 3. (d) Level 4. (e) Level 5.

spacing selected here is based on a $y+ = 1$ flat plate estimate, and two layers of constant spacing is also used in the wall-normal direction. Unlike the original set of runs, a grid refinement factor of 1.5 in each dimension is used instead of the conventional factor of 2 increase used in the original set of runs. This allows for a total increase by a factor of 3.375 instead of 8 between grid levels. Figure 5.2 and Figure 5.3 shows the surface and volume mesh discretization generated by EPOGS. Both figures show a consistent refinement occurring in all grids along the surface and also in the volume regions, which is necessary for establishing grid convergence.

Figure 5.4 shows the grid convergence behavior of the integrated loads. From the "eye" perspective, most solvers appear to show generally decreasing load variations as the meshes are refined, and the loads appear to trend towards a similar solution as the mesh resolution increases. In the case of the global maximum turbulent viscosity, however, the automatic meshing results show similar behavior as other solvers in that further grid refinement may be required before the quantity is sufficiently grid-converged [111]. Table 5.2 shows the values obtained for each grid level for the automatic mesh solutions. The variations in loading coefficients between the finest three grid levels are below a single load count which, in most engineering applications, can be considered to be sufficiently grid-converged.

Table 5.3 show the final loads comparison at the finest grid levels for each solver's respective grid families. Overall, the OF-EPOGS loads generally fall within the spreads of each load coefficient. The

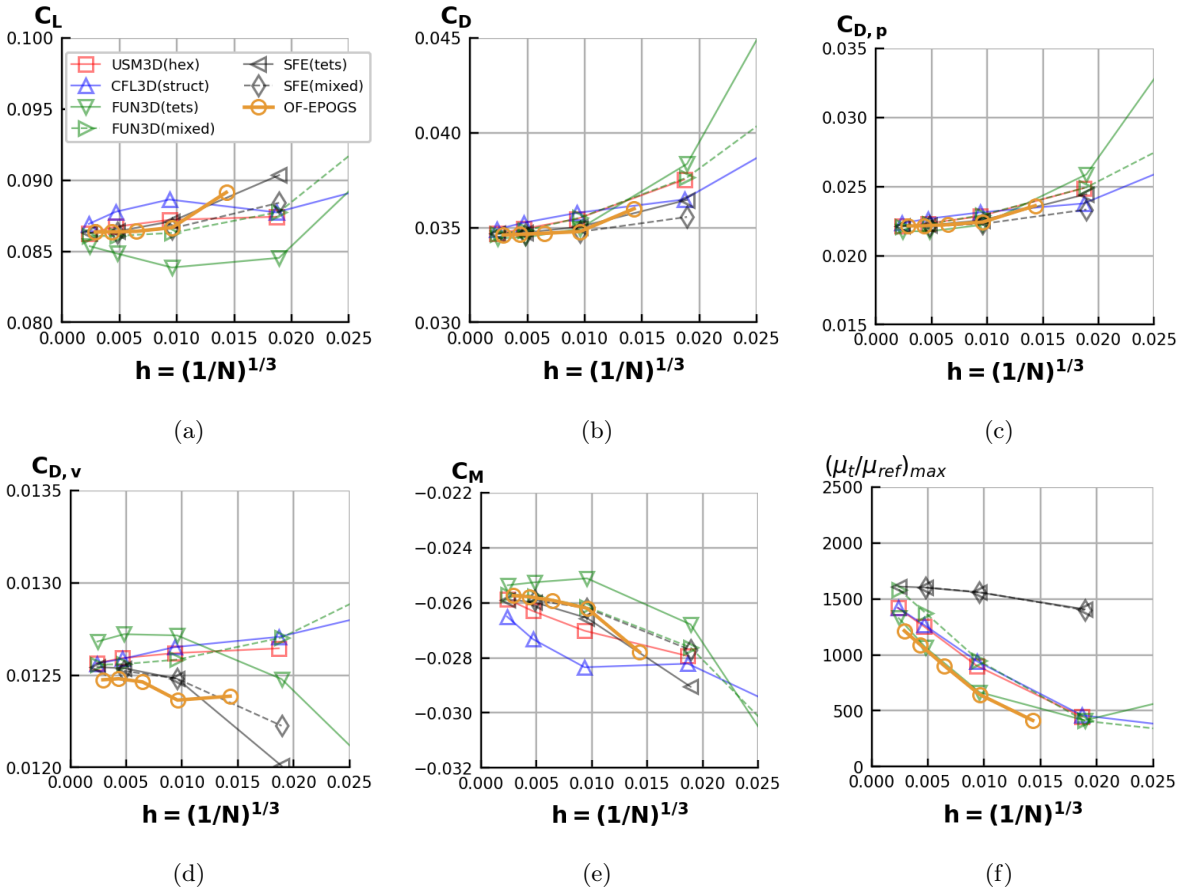


Figure 5.4: Grid convergence of aerodynamic load coefficients (lift, drag, pressure and viscous drag, and pitching moment) and the eddy viscosity for the hemispherical cylinder validation case.

Table 5.2: Loads coefficients obtained at all grid levels for the automatically-meshed flow solutions.

Level	N	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
1	41.0M	0.086352	0.034611	0.022136	0.012475	-0.025740	1215.51
2	12.4M	0.086366	0.034643	0.022163	0.012480	-0.025782	1089.63
3	3.72M	0.086401	0.034698	0.022234	0.012463	-0.025914	902.49
4	1.13M	0.086666	0.034791	0.022426	0.012365	-0.026203	645.62
5	341K	0.089137	0.035995	0.023609	0.012386	-0.027794	413.89

only load that falls outside the variation range of the other solvers is the viscous drag coefficient. However, the viscous drag differs just under a single count from the low end of the cohort of reference solutions, indicating the viscous drag is still sufficiently close to the rest of the solutions.

Table 5.4 shows the convergence rate using the uncertainty analysis of Celik et al [114], and Table 5.5 shows the Grid Convergence Index (GCI) ratios determining whether the analyzed quantities are within the asymptotic range. Across the various computed load coefficients, the EPOGS flow solutions show a

Table 5.3: Loads coefficients obtained at the finest grid level for the various flow solvers.

Solver	N	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
USM3D (hex)	78.6M	0.08627	0.03469	0.02212	0.01256	-0.02587	1418.31
CFL3D (struct)	78.6M	0.08692	0.03491	0.02234	0.01256	-0.02650	1420.97
FUN3D (tets)	71.4M	0.08535	0.03445	0.02177	0.01268	-0.02536	1327.41
FUN3D (mixed)	71.4M	0.08607	0.03466	0.02211	0.01256	-0.02569	1583.80
SFE (tets)	71.4M	0.08628	0.03468	0.02213	0.01254	-0.02589	1610.42
SFE (mixed)	9.0M	0.08629	0.03466	0.02214	0.01252	-0.02590	1601.79
OF-EPOGS	41.0M	0.08635	0.03461	0.02214	0.01248	-0.02574	1215.51

Table 5.4: Order of grid convergence p (see Appendix A) evaluated at finest grid levels.

Solver	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
USM3D (hex)	0.049021	1.142883	1.243329	-0.033327	0.723334	1.076906
CFL3D (struct)	0.023072	0.590622	0.515011	1.303487	0.322140	1.067104
FUN3D (tets)	0.935349	4.088136	5.377395	-2.591009	0.450870	0.601960
FUN3D (mixed)	4.024479	1.893320	1.868531	2.792185	2.915150	1.023879
SFE (tets)	2.613386	2.482824	2.573623	3.187678	2.631760	2.206888
SFE (mixed)	2.571892	2.829696	2.777642	2.627468	2.405666	1.973186
OF-EPOGS	2.179169	1.282647	2.437900	2.854807	2.880382	0.977971

Table 5.5: Asymptotic range ratio (see Appendix A) between finest and next finest grid levels. Values close to 1.0 indicate solutions are within asymptotic range.

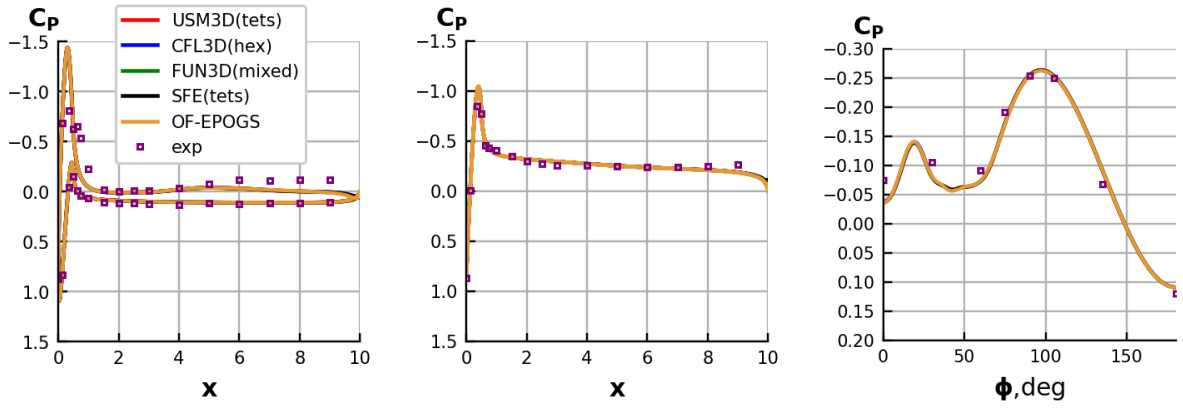
Solver	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
USM3D (hex)	1.040092	1.006938	1.009633	0.979309	1.016757	0.883014
CFL3D (struct)	1.026051	1.199679	1.391049	1.002043	1.288801	0.891532
FUN3D (tets)	0.993949	16.830938	40.933956	0.167683	1.359472	0.939855
FUN3D (mixed)	16.096586	1.005277	1.008114	1.000282	1.002117	0.866428
SFE (tets)	1.001357	1.001591	1.002780	9.027114	1.003637	0.994076
SFE (mixed)	1.003627	1.003295	1.007079	0.996605	1.011407	0.973937
OF-EPOGS	1.000164	1.000934	1.001212	1.004686	1.001611	0.896437

generally consistent above second-order convergence rate. The only exception is the total drag, which exhibits a convergence rate just above one. However, the pressure and viscous components of the drag coefficients individually show above second-order convergence rates. It is possible the reduced convergence rates of the total drag is due to the opposing concavities of the grid-convergence trends in the two components. In addition to this, all integrated loads exhibit an asymptotic ratio of 1, indicating that the

finest three grids are within the asymptotic range. The formal global order of the spatial discretization used is second order from the centered differencing of the viscous terms. The inviscid flux terms, on the other hand, are third order. The resulting convergence rates between second and third-order can likely be attributed to the dominance of inertial forces over viscous forces in high Reynolds-number flow. The maximum turbulent viscosity, on the other hand, exhibits a convergence rate just under one. Analysis of the grid convergence suggests that the quantities are not quite within the asymptotic range, which further indicates that finer meshes may be needed to achieve proper grid convergence behavior for this specific quantity.

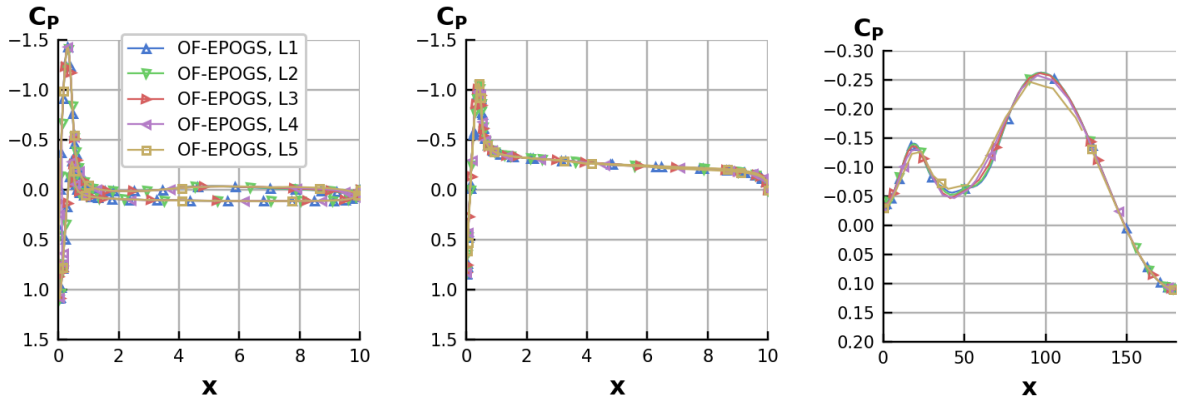
It is also interesting to compare the convergence rates of the obtained loads from the reference cohort. Some of the solvers are unable to demonstrate a consistent positive convergence rate, while other solvers exhibit positive convergence rates but the results may not be within the asymptotic range. Of the reference cohort, only the SFE solver with mixed tet/prism grid shows both consistent second to third-order convergence rates and also exhibits convergence within the asymptotic range. Overall, it appears that the OF-EPOGS meshes with fracturing are capable of performing well with regards to grid convergence behavior, providing evidence that the grid-fracturing appears to have little to no effect on the numerical accuracy of the solver with respect to evaluating the integrated loads.

Figure 5.5 shows various pressure cuts along the surface of the geometry at the finest grid level for each solver. Overall, it can be observed that the OF-EPOGS flow solutions show very good agreement with the other solvers due to the coverage of the results over the other results from the reference cohort. This provides stronger indication that the fracturing is not a concern with regards to the local flow solution in addition to the integrated loads. Figure 5.6 shows the pressure cuts for all grid levels of the automatic mesh solutions. Most levels (aside from the coarsest) show reasonably good agreement for the local pressure profiles.



(a) (b) (c)

Figure 5.5: Surface C_p cuts on the hemispherical cylinder at (a) $Y = 0.0$, (b) $Z = 0.0$, and (c) $X = 5.0$.



(a) (b) (c)

Figure 5.6: Surface C_p cuts on the hemispherical cylinder at (a) $Y = 0.0$, (b) $Z = 0.0$, and (c) $X = 5.0$ for OF-EPOGS at various grid levels.

5.1.3 ONERA-M6 Wing

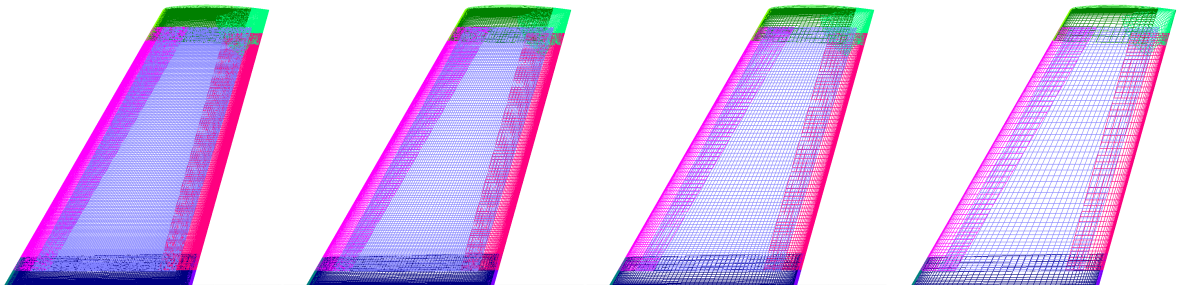
The next benchmark validation test case is the ONERA M6 Wing with sharp trailing edge at $M_\infty = 0.84$ and $\alpha = 3.06°$. This test case originates from the experiments originally conducted by Schmitt and Charpin [117]. The current verification case, however, more closely follows more recent investigations into the geometry with a sharp trailing edge [118]. The CAD geometry available on the TMR website is not amenable for use with EPOGS because of the abnormal parameterization and construction of faces. To alleviate this issue, a new BRep solid model is generated in ESP using a sweep operation applied to the original airfoil profile with added sharp trailing edge. The new geometry exhibits good agreement with the original CAD model with the exception of the rounded wing tip, of which there are slight deviations. This is due to underlying differences in how ESP and the original CAD software produced the closed tip region. Nevertheless, slight deviations in the wing tip should not result in significant changes in the overall wing loading at modest angles of attack.

Note that the $Y = 0$ boundary is a symmetry boundary, while the other outer boundaries away from the wing have far-field boundary treatment. An adiabatic wall boundary condition is applied to the wing surface.

Table 5.6: EPOGS input parameters used to generate the coarsest grid level for ONERA-M6 wing.

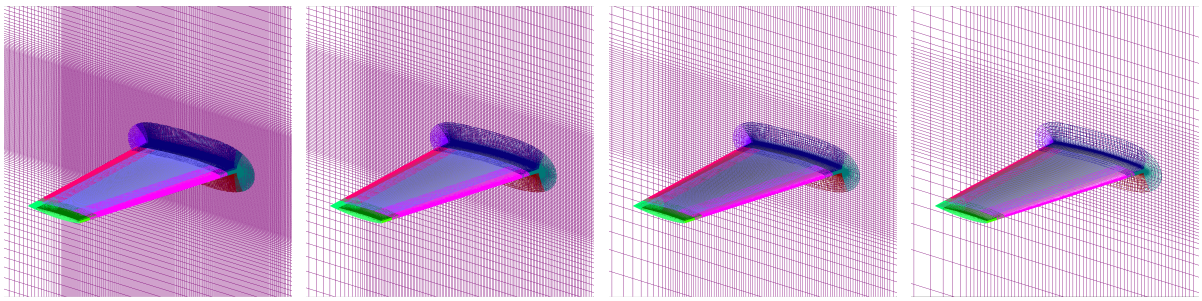
Mesh Parameter	Δs_{max}	$\Delta\theta_{max}$	SR_{max}	Δs_{wall}	n_{fringe}	np_{min}
Value	$0.03m$	$5.0°$	1.25	$1.2968 \times 10^{-6}m$	2	9

Like the previous case, a grid family is generated from a coarse-level mesh generated with EPOGS. Table 5.6 shows the input parameters used to generate this mesh. Again, the wall-spacing selected here is based on a $y+ = 1$ estimate with two layers of constant spacing in the wall-normal direction. A



(a) (b) (c) (d)

Figure 5.7: Surface mesh on upper surface of ONERA-M6 wing at different refinement levels of EPOGS-generated mesh (Level 1 is the finest, and Level 4 is the coarsest). (a) Level 1. (b) Level 2. (c) Level 3. (d) Level 4.



(a) (b) (c) (d)

Figure 5.8: Volume mesh slice at $Y = 0.0$ of ONERA-M6 mesh generated with EPOGS. (a) Level 1. (b) Level 2. (c) Level 3. (d) Level 4.

refinement factor of 1.5 is also used here instead of factor 2 used in the original study. This refinement factor was chosen to ensure consistent refinement without the mesh sizes growing too large, since the coarsest mesh generated by EPOGS is just slightly finer than the third finest meshes in the reference cohort. Unlike the cylinder case, however, only four grid levels are considered due to the relatively large size of the coarsest mesh at 1.65M points. A smaller mesh was not possible without modifying certain input parameters to ranges that are not recommended, such as increasing the maximum stretching ratio SR_{max} to 1.3 or beyond. Figure 5.7 shows the surface meshes of the upper surface at each grid level, and Figure 5.8 shows the a slice of the volume mesh at the symmetry plane. The original coarse level grid (Figure 5.7d) shows how the EPOGS tool automatically focuses the refinement. EPOGS automatically refines node and edge surface spacing near the feature curves from which they are extracted, resulting in the favorable refinement at the leading and trailing edge meshes (magenta and red, respectively). The refinement of meshes between grid levels is global, which can be observed by the consistent refinement occurring in both the geometry surface and volume meshes.

Figure 5.9 shows the grid convergence plots for the load coefficients and maximum eddy viscosity. Overall, the plots show that the automatic overset grid solutions trend towards similar loads as the

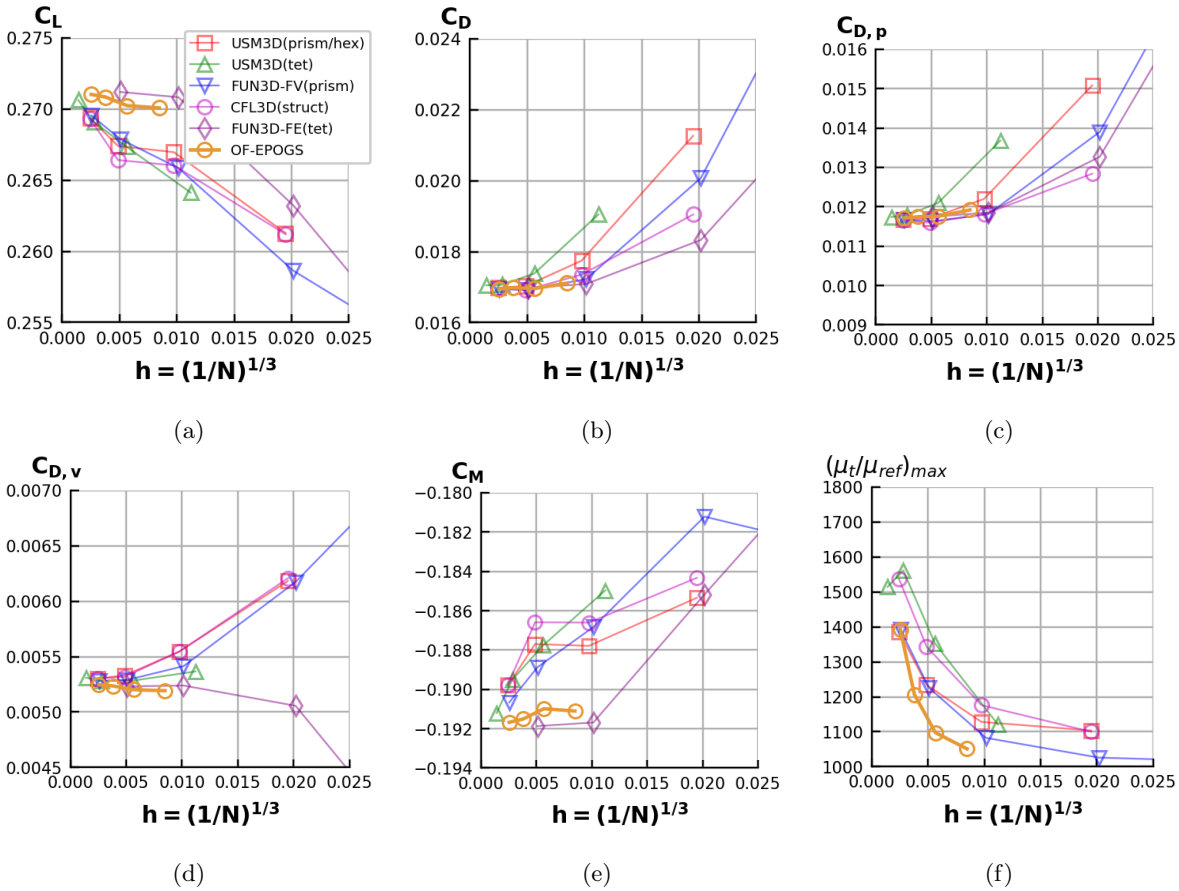


Figure 5.9: Grid convergence of aerodynamic load coefficients (lift, drag, pressure and viscous drag, and pitching moment) and the eddy viscosity for the ONERA M6 Wing benchmark case.

reference cohort solutions. Similar trends of lack of grid convergence is also exhibited in the maximum eddy viscosity, where all solvers show a large growth in eddy viscosity as the grid is refined. While the trends toward the infinite grid may be similar, it is interesting to observe that, with regard to the load coefficients, the automatic grid solutions exhibit parabolic trends more consistently in comparison to the other solutions from the other solvers, especially in the lift and pitching moment. Several other solvers in the reference cohort exhibit a relatively large divergence in the load "deltas" even at the finest grid levels.

Table 5.7: Load coefficients and maximum eddy viscosity obtained at all grid levels for the automatically-meshed flow solutions for the ONERA-M6 benchmark case.

Level	N	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
1	61.3M	0.271022	0.016953	0.011706	0.005247	-0.191679	1392.18
2	18.4M	0.270838	0.016983	0.011747	0.005236	-0.191512	1206.35
3	5.51M	0.270234	0.016961	0.011759	0.005202	-0.190990	1097.38
4	1.65M	0.270070	0.017111	0.011918	0.005193	-0.191103	1051.10

Table 5.8: Load coefficients obtained for the ONERA-M6 at the finest grid level for the various flow solvers.

Solver	N	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
USM3D (pris/hex)	69.2M	0.26935	0.01698	0.01168	0.00530	-0.18980	1386.25
USM3D (tets)	363M	0.27062	0.01705	0.01174	0.00531	-0.19120	1514.25
FUN3D-FV (pris)	60.8M	0.26955	0.01695	0.01167	0.00528	-0.19067	1392.82
CFL3D (struct)	69.2M	0.26932	0.01696	0.01165	0.00530	-0.18976	1535.98
FUN3D-FE (tets)	7.63M	0.27120	0.01698	0.01175	0.00523	-0.19187	-
OF-EPOGS	61.2M	0.27102	0.01695	0.01171	0.00525	-0.19168	1392.18

Table 5.7 shows the quantities obtained for the integrated loads and "maximum" eddy viscosity. The "maximum" eddy viscosity obtained here for the Overflow solutions is not the true maximum eddy viscosity in the flow domain, but rather the maximum eddy viscosity obtained near the wall boundary layer. Because of how the off-body grid is constructed, the wake is refined much further downstream in comparison to the other meshes used in the reference solutions. As a result, this dissipates the wake through increasing the local cell size in the wake away from the wing surface. The tracking of the wake far downstream can result in the turbulent eddy viscosity to continue to grow because of the presence of vorticity from the propagated wake. The destruction term is also deactivated in the wake because its magnitude is inversely correlated to the distance to the geometry surface.

Between the various grid levels, the lift and pitching moment coefficients exhibit a variation range of less than 10 counts, and the variation range in the drag coefficients in the three finest levels are less than a single count. While the convergence rates of the various load quantities are not immediately apparent, the three finest loads exhibit variations below the commonly-accepted level of uncertainty for these quantities in an engineering context. Thus, the results obtained on the finest three grids can be considered to be grid independent.

Table 5.8 shows a comparison of the finest grid level solutions obtained for the various combinations of

Table 5.9: Convergence rates evaluated at finest grid levels for the ONERA-M6 benchmark case.

Solver	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
USM3D (prism/hex)	-2.160	3.796	4.526	2.858	-4.544	-0.527
USM3D (tets)	0.144	4.239	2.883	-4.963	0.063	2.154
FUN3D-FV (prism)	0.285	3.645	2.359	3.545	0.203	-0.220
CFL3D (struct)	-2.948	3.481	1.915	3.767	-7.814	-0.206
FUN3D-FE (tets)	4.399	3.537	3.829	4.619	5.215	-
OF-EPOGS (struct)	2.959	-0.836	-3.013	2.764	2.836	-1.328

Table 5.10: Asymptotic range ratio analysis between finest and next finest grid levels for the ONERA-M6 benchmark case.

Solver	C_L	C_D	$C_{D,p}$	$C_{D,v}$	C_M	$(\mu_t/\mu_{\text{ref}})_{\max}$
USM3D (prism/hex)	0.2222	13.9362	23.0773	1.0056	0.0424	0.6175
USM3D (tets)	1.0984	18.9046	1.0036	0.0319	1.0357	1.0308
FUN3D-FV (prism)	1.2105	12.4362	0.9958	11.6421	1.1403	0.7560
CFL3D (struct)	0.1282	11.1381	0.9952	13.6600	0.0044	0.7580
FUN3D-FE (tets)	20.8412	11.5806	14.1996	24.3325	36.6250	-
OF-EPOGS (struct)	0.9993	0.7160	0.2992	0.9979	0.9991	0.5083

solver and grid type. The lift and pitching moment coefficients of OF-EPOGS does sit outside the range of the quantities of the equivalent or finer grids, but it is within ten counts of the range of variations. On the other hand, the various drag quantities lie within the variation range of the reference cohort.

Table 5.9 and Table 5.10 show the convergence rates and asymptotic range ratio for each combination of solver and grid type at the finest grid level, respectively. As evidenced earlier by Figure 5.9f, most results exhibit a divergence in the variations between grid levels and are very likely not within the asymptotic range. The only solver that exhibits a consistent grid convergence while in the asymptotic range is the USM3D solution computed on a tetrahedral grid family. This behavior is likely due to the fact that the grid family used is significantly finer than the others. With regards to the load convergence, it appears that the convergence rates and evaluation of whether the solutions are the asymptotic range are not consistent across the various load quantities. In the case of the Overflow solutions on the automatic meshes with fracturing, the lift, viscous drag, and pitching moment coefficients exhibit roughly below third-order convergence rates while also being within the asymptotic range. This is generally consistent with the order of accuracy expected from the spatial discretization. On the other hand, the total and pressure drag coefficients do not exhibit such similar convergence rates/behavior. It is possible that the lack of convergence is due to the transonic shocks over the upper surface. Discontinuities can reduce the order of accuracy of the spatial discretization schemes. However, despite this discrepancy, this convergence behavior is comparable, if not exhibits marked improvement in comparison to the other solvers in the reference cohort.

Figure 5.10 shows various surface pressure cuts along the span of the ONERA-M6 wing for the finest grid level of each grid family. Here, the tetrahedral solutions for USM3D and FUN3D-FE were not included due to lack of available data. Despite this, the comparison may be more consistent across the solvers due to the similarities in grid size. Across the various cuts, the OF-EPOGS solution shows good agreement with the pressure profiles obtained from the other solvers, as well as consistent comparability with the experimental data. Figure 5.11 shows the same surface pressure cuts across the various grid

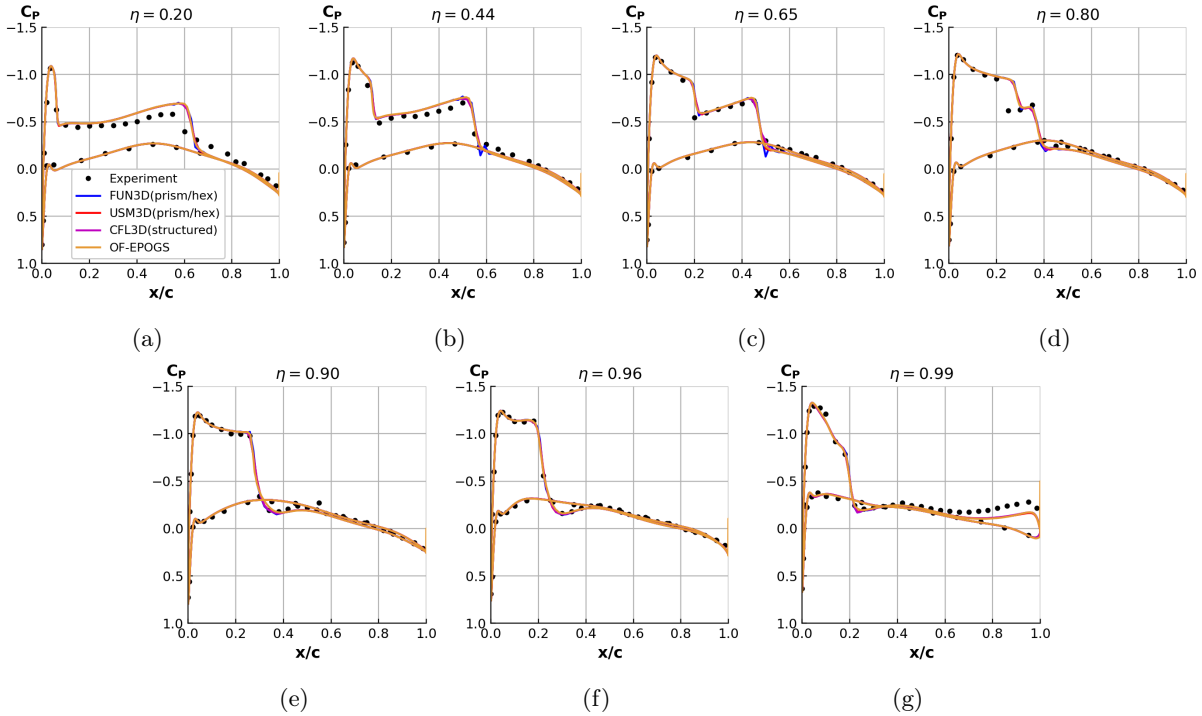


Figure 5.10: Surface C_p cuts along various span-wise locations of ONERA-M6 wing on Level 1 grids.

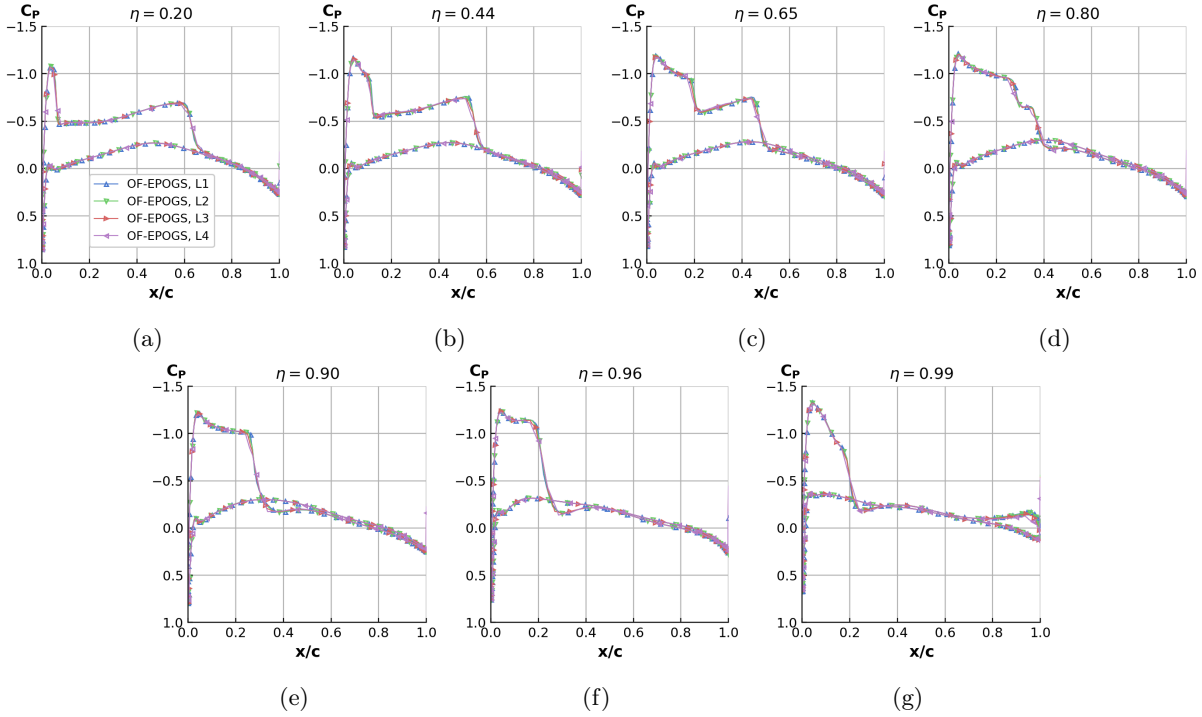


Figure 5.11: Surface C_p constant-span cuts on ONERA-M6 wing for all grid levels in OF-EPOGS mesh family.

levels in the OF-EPOGS flow solution family. Again, the results also show relatively good agreement across the various levels. This is indicative of the small variations across the loads between each grid level as observed in Table 5.7. Minor discrepancies can be found in the resolution and onset of the shocks.

5.2 Concept Vehicle Test Cases

The next set of test cases are larger wing-body concept vehicles that are representative of real aircraft configurations typically analyzed in engineering applications. While the geometries themselves may not contain many detailed features, the size of these cases are of the same order of most CFD analyses of full-size aircraft. As such, additional focus is given to the numerical and computational efficiencies. The following discussion was previously compiled into AIAA Aviation conference papers by Chuen et al [108] and Chan et al [83].

5.2.1 Juncture Flow with F6 Wing

The first test case is the Juncture Flow test vehicle with F6 Wing profile in free air at $\alpha = 5.0°$. This case was originally investigated by Lee and Pulliam [87] as part of a larger experimental campaign [87,95,107,119–124] to improve CFD modeling and simulation of the separation bubble at wing-root juncture regions. However, for the purpose of investigating any effects that the automatic mesh generation and/or fracturing have on the flow solutions, only the free-air computation with the SARC-QCR2013 turbulence model is considered for the purpose of verification of the flow solutions computed on EPOGS meshes. Figure 5.12 shows a representation of the geometry, and Table 5.11 shows the free-air flow conditions.

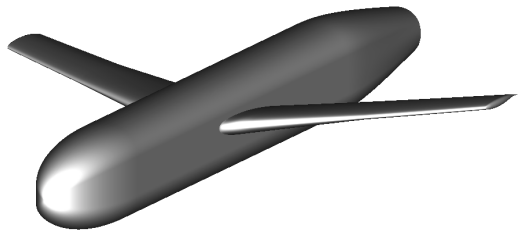


Figure 5.12: Juncture Flow Model with F6 wing.

Table 5.11: Flow conditions for the Juncture Flow Model with F6 wing in free-air.

Flow Parameter	M_∞	α	Re	T_∞
Value	0.189	5.0°	2.4M	$529°R$

5.2.1.1 Manual and Automatic Meshing Procedures

For this particular case, the author only generated the automatic mesh. The manual meshes and flow solutions come from the original study conducted by Lee and Pulliam [87], which was kindly supplied by the original authors. Table 5.12 shows the global mesh generation parameters fed into the automatic mesh generation software. Table 5.13 shows a breakdown of the time spent performing each of the five tasks in the hybrid automatic approach. The second column presents an estimate of the required time

Table 5.12: Mesh parameters used for automatic mesh generation (medium resolution) of the JFM-F6.

Mesh Parameter	SR_{max}	Δs_{max}	np_{min}	Δs_{wall}	$\Delta\theta_{max}$	n_{fringe}
Value	1.2	16.0 mm	21	$2.694\times10^{-3}mm$	4.0°	2

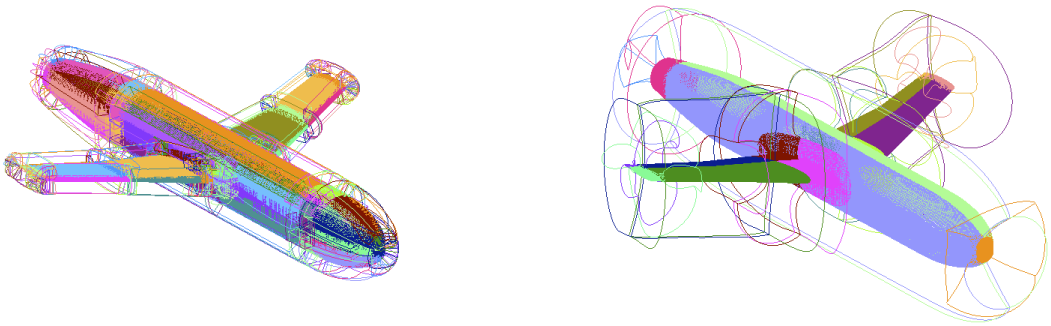
to build the original manual mesh with at least an intermediate level of mesh generation expertise. As expected, the automatic mesh enables a significant reduction in manual labor needed to produce a mesh.

To generate the near-body automatic mesh, care was taken to ensure that the resulting mesh would be comparable to the manual mesh in terms of the surface and near-wall resolution. Generally, this involves specifying the same stretching ratio and a maximum grid spacing based on the resulting maximum spacing of the wing of the manual mesh. Minimum point counts are also specified for certain regions, such as the number of points across the finite-thickness trailing edge. The same wall normal spacing is also specified to produce a similar near-wall clustering and y+ value in the near-body volume meshes. One discrepancy, however, that could not be controlled is the marching distance of the volume meshes normal to the wall. The automatic meshes target a marching distance such that the outer normal spacing is nearly equivalent to the maximum tangential edge spacing on the wall surface. This is to ensure that the outer layer spacing is as isotropic as possible. As a result, the near-body meshes do not march as far out as that of the manual mesh. This can be observed in Figure 5.13, where the outer edges of the near-body volume meshes extended much further away from the vehicle surface for the manual meshes.

Table 5.13: Turnaround times to construct an overset mesh using automatic and manual approaches for an intermediate user, assuming an 8 hour work day. The times for the manual meshes from [87] are given as estimates. (C) is CPU wall-clock time, (M) is manual labor time.

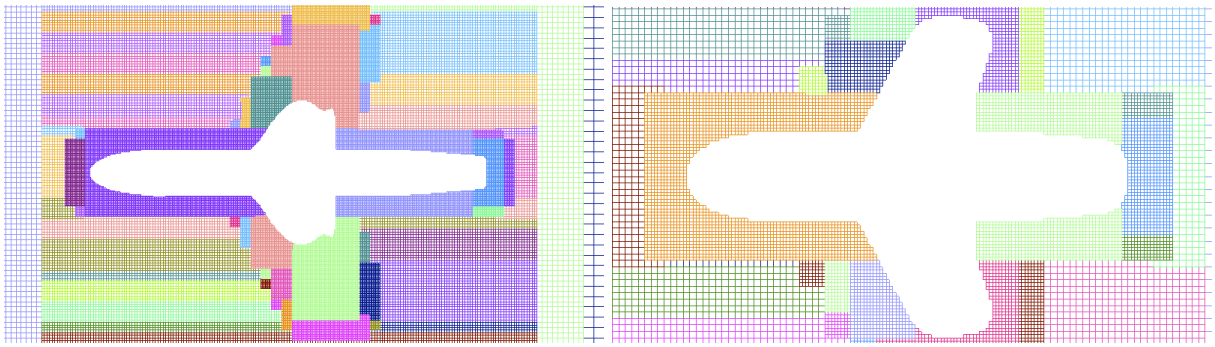
Steps	Automatic	Manual
Surface Meshing	3 min (C)	1-2 days (M)
Volume Meshing	40 s (C)	2-3 hrs (M)
Volume Mesh LCBC	40 min (C)	-
Manual Mesh Repair	1 hrs (M)	-
Domain Connectivity	1 hrs (M)	1 hrs (M)
Flow Solver Inputs	10 min (M)	30 min (M)
Total Turnaround Time	$\sim$ 3 hrs	$\sim$ 2 days

The off-body Cartesian box grids for the automatic mesh were also generated using Overflow's off-body mesh generation scheme to ensure both meshes have similarly constructed flow domains. A different level 1 spacing was used for the automatic mesh, however, due to differences in the outer boundary normal spacing for the near body grids. Because of the differences in marching distance away from the wall surface, the manual mesh has much larger spacing in the normal direction. As a result, the level 1 spacing can be much coarser in the surrounding off-body meshes. Figure 5.14 shows a cut plane of



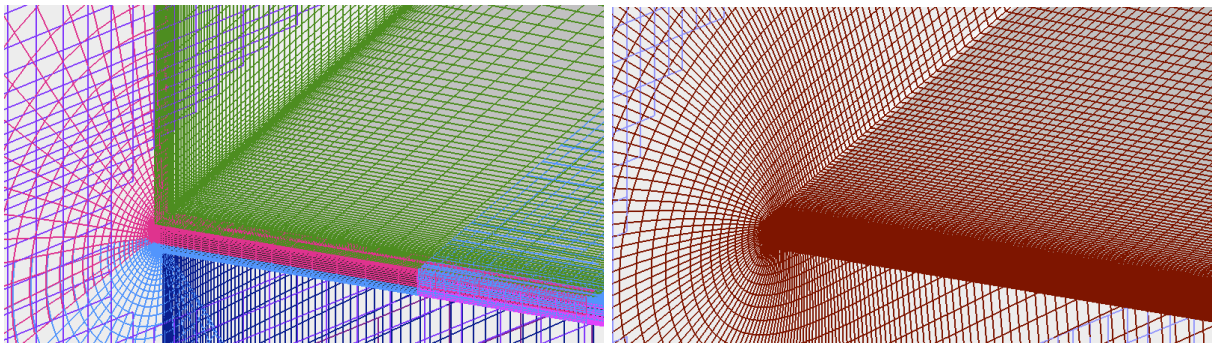
(a) (b)

Figure 5.13: Near-body volume meshes generated for the JFM-F6 test case. (a) Automatic. (b) Manual.



(a) (b)

Figure 5.14: Constant $Z = 0.0$ cut of off-body volume meshes surrounding near-body meshes for the JFM-F6 test case. (a) Automatic. (b) Manual.



(a) (b)

Figure 5.15: Trailing edge region of wing-body juncture of JFM-F6. (a) Automatic. (b) Manual.

$Z = 0.0$ of the off-body meshes around the geometry. As expected, the grid spacing is much finer for the automatically generated mesh in comparison to the manual mesh. In the automatic mesh, the level 1 spacing is $\Delta s = 16.0mm$, while the manual mesh has a level 1 spacing of $\Delta s = 40.0mm$, indicating that the off-body edge spacing for the automatic meshes is finer by a factor of 2.5.

Another area of interest where the meshes differ in refinement and point clustering is the wing-body

juncture in the trailing edge region. This region was the main focus of the original study, and as a result the manual mesh was generated with custom refinement in this region. The automatic mesh, on the other hand, is only able to allow manual specification of the number of points across the finite thickness trailing edge. The refinement in this region is otherwise left to the algorithms within the automatic meshing software. For this particular case, the automatic mesh is able to recognize this region as concave and refines accordingly. Figure 5.15 shows the juncture trailing edge region of both the manual and automatic mesh. The local spanwise refinement is relatively high on both the fuselage and wing surface in the manual mesh. In the case of the automatic mesh, the stretching in the span-wise direction is more aggressive and exhibits a much tighter clustering at the junction edge. The manual mesh exhibits a less aggressive stretching and clustering and, instead, appears to have a more uniform spacing.

Table 5.14 shows a quantitative comparison of the grid composition between the automatic and manual grids. Overall, it appears that the surface meshes are quite close in terms of point count ($\sim 1\%$), suggesting that the general surface resolution of the geometry is also relatively close. However, there is nearly an order of magnitude more grids in the auto-generated near-body meshes. This is expected, since the manual user has more discretion in decomposing the surface geometry. The difference in the grid count of the off-body grids is not as large, however. The larger quantity of off-body grids surrounding the automatically-generated near-body grids can be observed in Figure 5.14. Again, this can be attributed to the difference in normal marching distance of the volume meshes.

It is also interesting to note the differences in the grid point count between the automatic and manual meshes. This can mostly be attributed to the differences in the volume mesh marching distances. The manual mesh grew 83 points out in the normal direction compared to the automatic meshes' 66 points. Had the automatic meshes also been grown 83 points out, the relative difference in grid points would match that of their respective surface meshes. Consequently, the off-body grid point count is automatically increased when generated in Overflow to compensate for the difference in coverage. Because nearly all of the critical flow gradients are near the wall, this discrepancy in the marching distance was considered

Table 5.14: Comparison of grid compositions between automatically and manually generated meshes for the JFM-F6 geometry in free-air.

	Automatic			Manual		
	Near-Body	Off-Body	Total	Near-Body	Off-Body	Total
Surface Points	570K	-	-	579K	-	-
Grid Count	181	258	439	17	158	175
Grid Points	37.7M	7.81M	45.5M	48.9M	1.98M	50.9M
Fringe Points	13.1%	6.4%	20.5%	6.4%	1.6%	8.1%
Blanked Points	1.6%	1.1%	2.8%	1%	0.3%	1.3%

Table 5.15: Comparison of Overflow MPI load balancing (280 Broadwell Cores) on automatically and manually generated meshes for the JFM-F6 geometry in free-air.

		Automatic	Manual
Before MPI-Splitting	Grid Count	439	178
	Total Points	45.5M	50.9M
After MPI-Splitting	Grid Count (Split Blocks)	1060	949
	Total Points	52.9M	65.2M
	Average Points/Group	189K	233K
	Chimera Boundary Exchange Time/Step (s)	3.42E-2	3.46E-2
	Parallel Efficiency	88.8%	92.7%
	Wall Time (40K steps)	16.2 hrs	21.9 hrs

acceptable.

The automatic mesh also exhibits over twice the relative overhead of fringe and blanked points compared to the manual mesh. The increase in fringe points is expected due to the larger quantity of grids in both the near and off-body meshes in the automatic meshes. Despite nearly 10 times more grids present in the near-body meshes, the fringe point overhead appears to only increase by around a factor of 2.5. The increase in the off-body mesh fringe point count is due to the use of a finer level 1 spacing and setting the overlap between near and off body meshes to be closer to the geometry surface. The near-body automatic meshes only show a nominal increase in blanked point overhead, while the off-body meshes exhibit a relatively large increase in blanked point overhead. Again, the increase for the off-body meshes can be attributed to the difference in near-body volume marching distance. Overall, this breakdown of the grid compositions provides confidence that the two meshes are comparable despite different approaches in the mesh generation.

Flow solutions for the automatic meshes were then computed using Overflow using 280 Intel Broadwell cores. Following the same procedures originally used by Lee and Pulliam [87], the Roe scheme with third-order Koren MUSCL limiter is used to discretize the inviscid terms. The turbulence is modeled using SA-RC-QCR2013V. Time advancement is performed using the Pulliam-Chaussee scalar pentadiagonal scheme with low-Mach preconditioning. A two-level grid cycling startup procedure is used to step through the transients, followed by a simple time advancement of the solution to reach the convergence criteria. For the automatic mesh, this took approximately 40K iterations.

Table 5.15 compares the resulting computational efficiency of this parallel run for both meshes. It is interesting to note that once the grids are split by Overflow for load balancing with MPI, the resulting grid and point count of the manual mesh increased relatively more than that of the automatic mesh. In

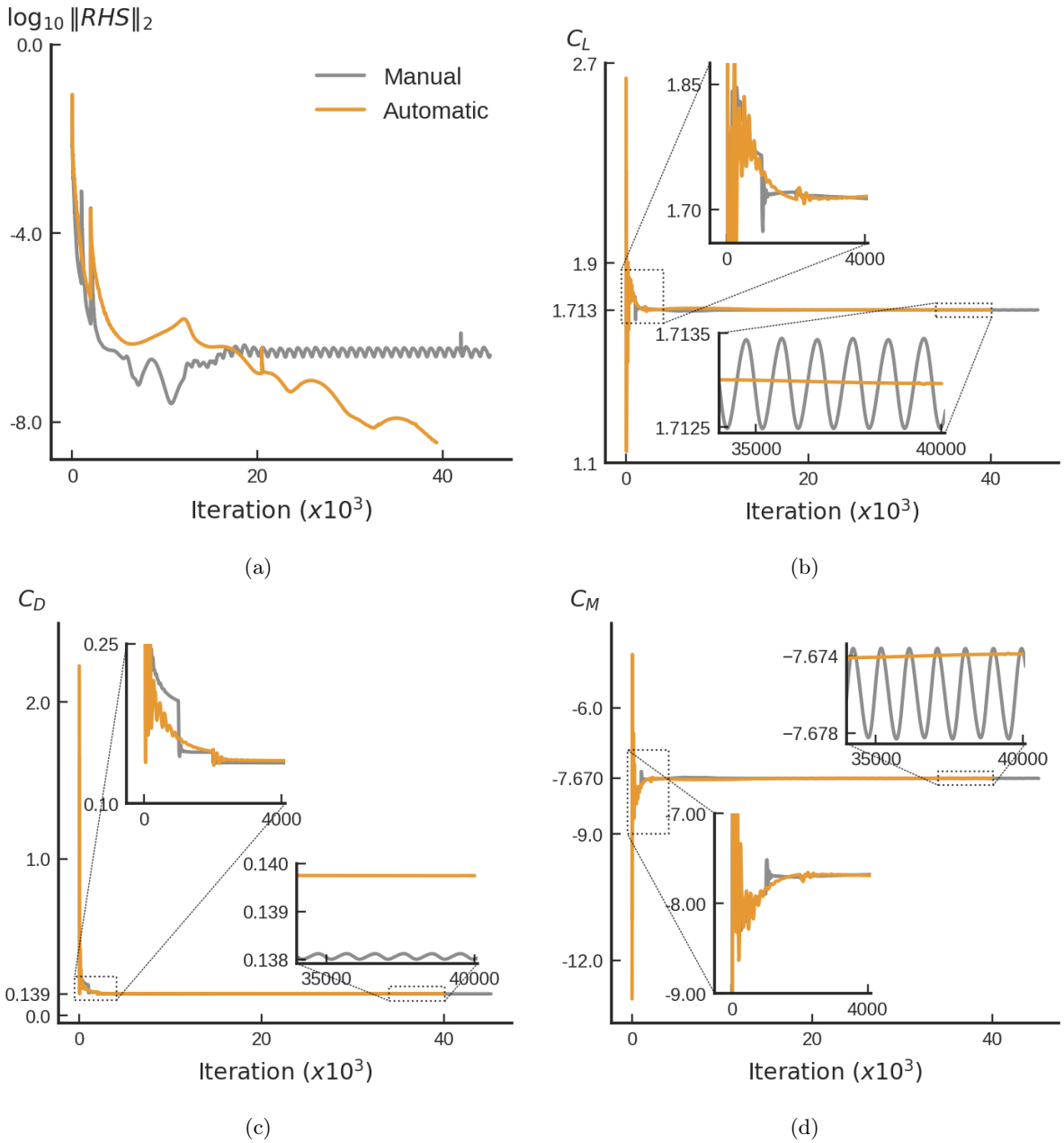


Figure 5.16: Solution convergence behavior of the JFM-F6 for both manual and auto-mesh solutions at medium level refinement. (a) Residual. (b) Lift coefficient. (c) Drag coefficient. (d) Pitching moment coefficient.

fact, the grid count after MPI splitting shows that the manual mesh is just as fractured as the automatic mesh. This is likely due to the fact that the near-body meshes in the manual grids require increased splitting, resulting in added point-matched fringe points for communication between the split chunks. Conversely, because the automatic meshes are already comparatively fractured, less splitting is required to efficiently group the chunks appropriately for each processor. It is also interesting to note that the

Table 5.16: Comparison of the converged aerodynamic loads for the JFM-F6 between the manual and automatic meshes.

	C_L	C_D	C_M
Manual Mesh	1.7130	0.1381	-7.6760
Auto mesh	1.7129	0.1398	-7.6739
% Difference	$\ll 1\%$	1.23%	$\ll 1\%$

increase in fringe points in the automatic mesh resulted in both grids exhibiting roughly similar wall time needed to transfer the solutions across the overset boundaries. Overall, the increase in computational overhead of an equivalent resolution automatic mesh appears to be minimal.

Figure 5.16 shows the residual and load coefficients convergence of both meshes. Both the automatic and manual mesh are able to achieve convergence of the load coefficients at about the same rate, where the initial transients appear to damp out over approximately 2000 iterations. Both grids are able to meet the residual convergence criteria. However, the manual mesh residual eventually stalls at a five-order drop, while the automatic mesh is able to continue converging past seven orders of magnitude residual reduction. Oscillatory behavior can be observed in the manual mesh where the convergence stalls, while the automatic mesh appears to settle to a stationary value. Overall, the final integrated values appear to show excellent agreement. Table 5.16 outlines the final force and moment coefficients obtained from both meshes. For the manual mesh, the values are averaged over the last several thousand steps, since the values oscillate in a consistent sinusoidal pattern. The largest difference between the results is in the drag coefficient, which exhibits a 1.23% difference, or a difference in approximately 17 counts of drag. The lift and moment coefficients show much smaller differences with a percentage difference of less than 0.1%.

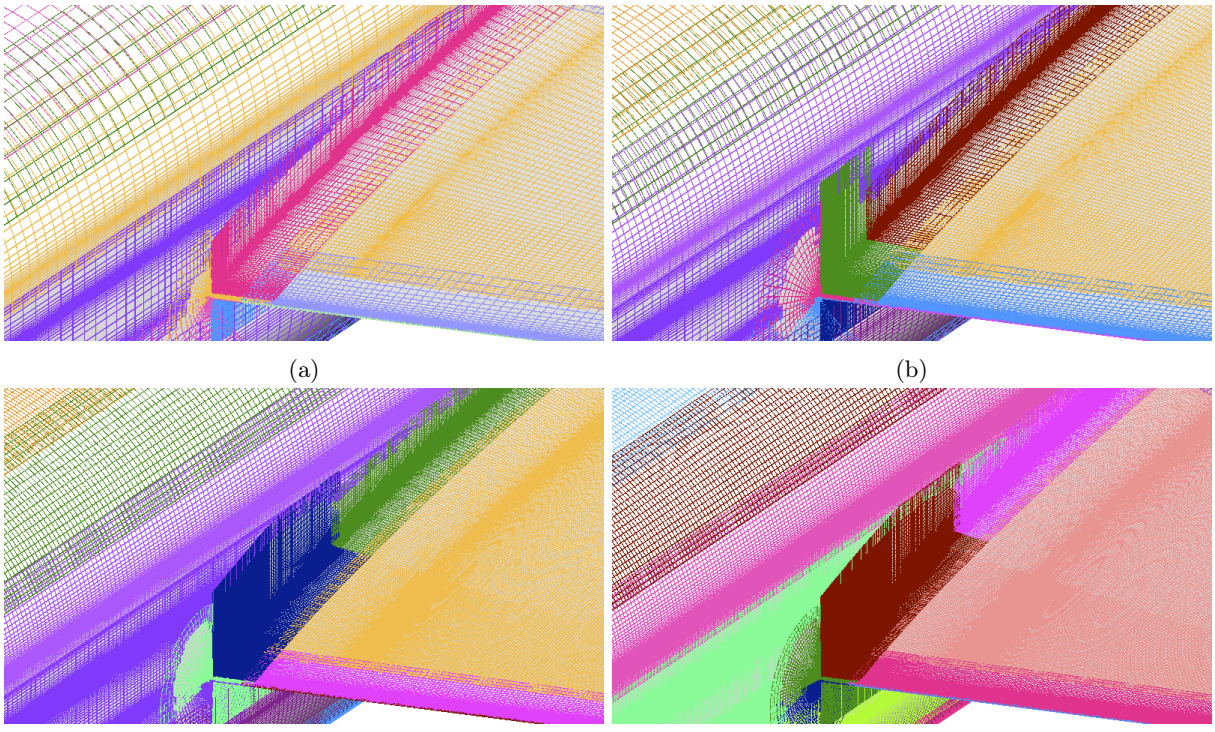
5.2.1.2 Grid Sensitivity Study

Following the comparison of a manual and automatic mesh at the medium level of refinement, a grid sensitivity study was performed to investigate the capabilities of the automatic mesh generation tools in producing grids at various levels of refinement. Table 5.17 outlines the parameters selected to generate a mesh at each level of refinement, as well as the resulting grid sizes produced with such inputs. The choice of these parameters attempt to follow the refinement procedure originally done by Lee and Pulliam [87]. From the resulting grid sizes, it can be observed that the grid point density is successively increased by a factor of 2 to 3 between each grid level.

Figure 5.17 provides a more localized perspective of the effect of varying the grid parameters on the surface mesh spacing, especially on the meshes at the wing root junction near the trailing edge. This particular region is a critical area of interest due to the production of the juncture flow separation [107]. These set of figures also demonstrate how the refinement is performed in the automatic mesh generation

Table 5.17: Mesh input parameters for EPOGS-generated meshes at four levels of refinement for Juncture Flow F6. Here, $n_{fringe} = 2$ for all grid levels due to the flow solver numerical scheme.

		Coarse	Medium	Fine	Extra-Fine
Inputs	SR_{max}	1.20	1.15	1.1	1.08
	Δs_{max} [mm]	25.0	17.0	9.0	7.0
	$\Delta\theta_{max}$ [deg.]	5.0	4.0	3.0	2.0
	np_{min}	15	21	31	43
	Δs_{wall} ($\times 10^{-3}$) [mm]	4.04	2.70	1.80	1.20
Outputs	Surface Points	337K	570K	1.40M	2.43M
	Near-Body Volume Points	17.2M	37.7M	130M	282M
	Total Grid Points	22.0M	45.5M	153M	332M



(a) (b) (c) (d)

Figure 5.17: Trailing-edge junction represented by the automatically generated grid family (cont'd.). (a) Coarse. (b) Medium. (c) Fine. (d) Extra-Fine.

framework. Because the same CAD model is referenced when generating these meshes, the meshes generally exhibit a consistent structure at all levels of refinement. This is due to the fact that the automatic mesh generation tools decompose the CAD topology in a similar manner every time prior to generating the individual meshes. However, the placement of boundaries and parameterization of

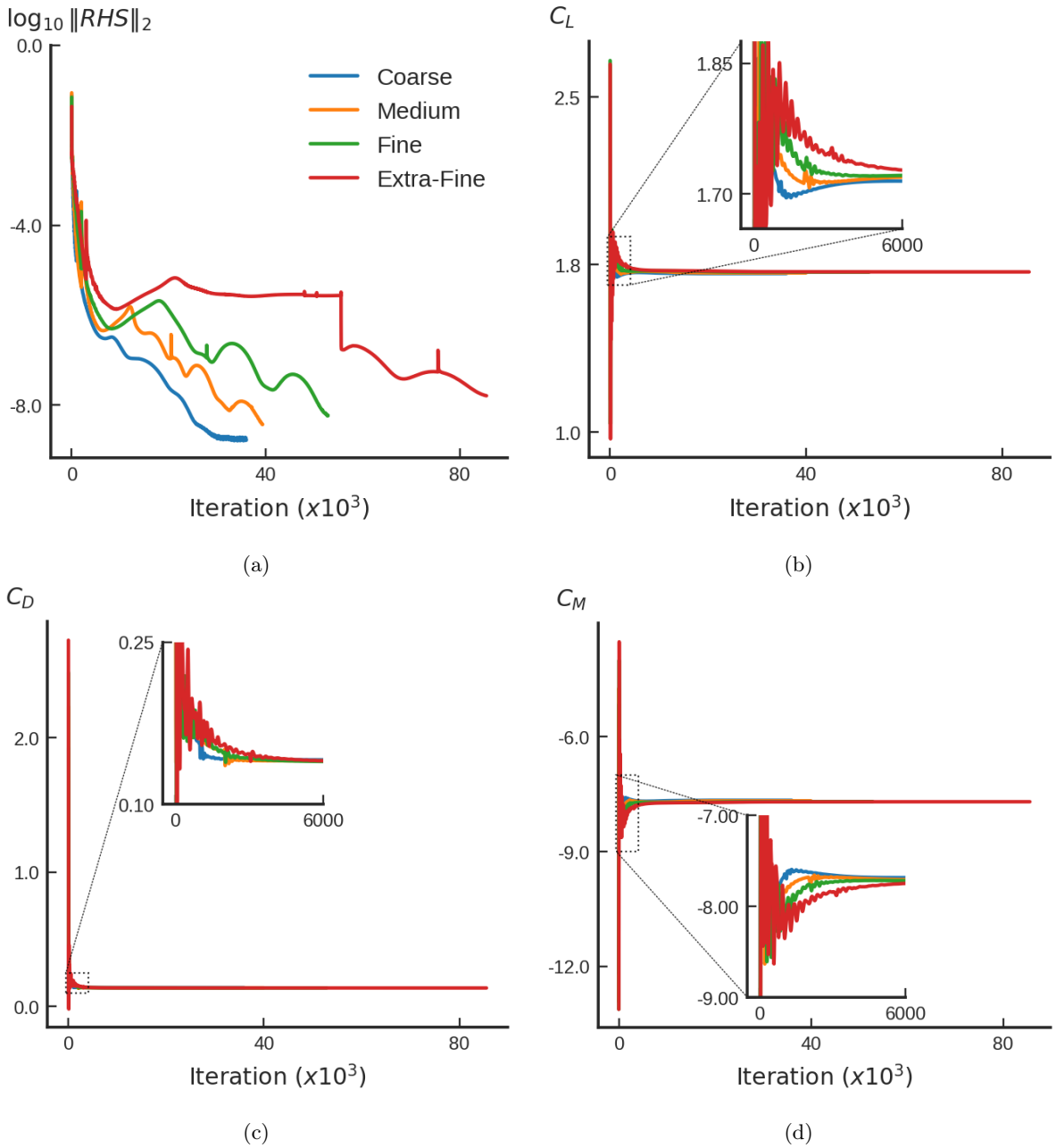


Figure 5.18: Solution convergence behavior for the JFM-F6 computed on the automatically generated grid family (cont'd.). (a) Residual. (b) Lift coefficient . (c) Drag coefficient. (d) Pitching moment coefficient.

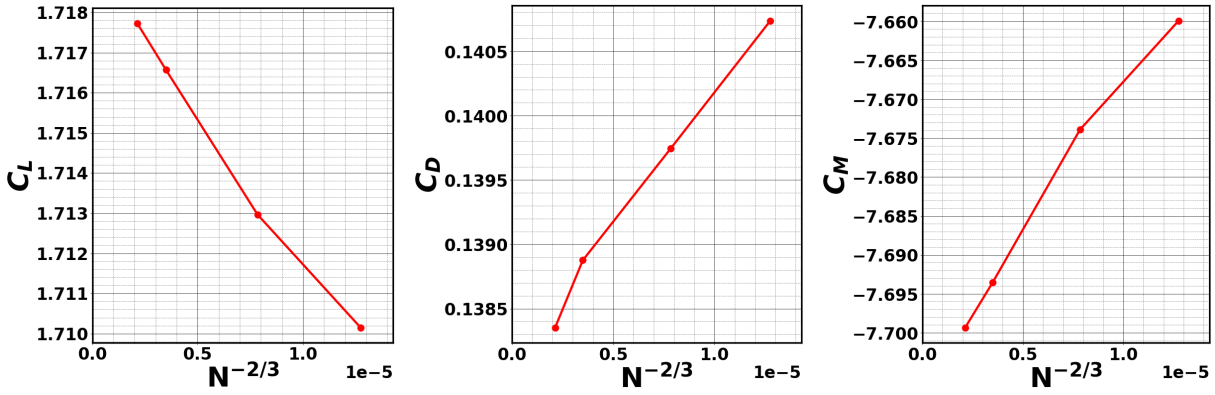
the meshes can still vary based on the meshing parameters, which prevent the mesh levels from being consistently refined globally throughout the mesh. As such, while refinement may be occurring between grid levels, it may not produce a valid grid family such that grid convergence trends can be properly evaluated.

Flow solutions were then computed on each of the grid levels. Like the medium flow solution case, each

level of refinement is computed with an initial grid cycling enabled to step through the initial transients faster. This usually involves stepping through 1000 steps on the coarser grid prior to returning to the original grid. The flow solution is then evolved through simple time advancement.

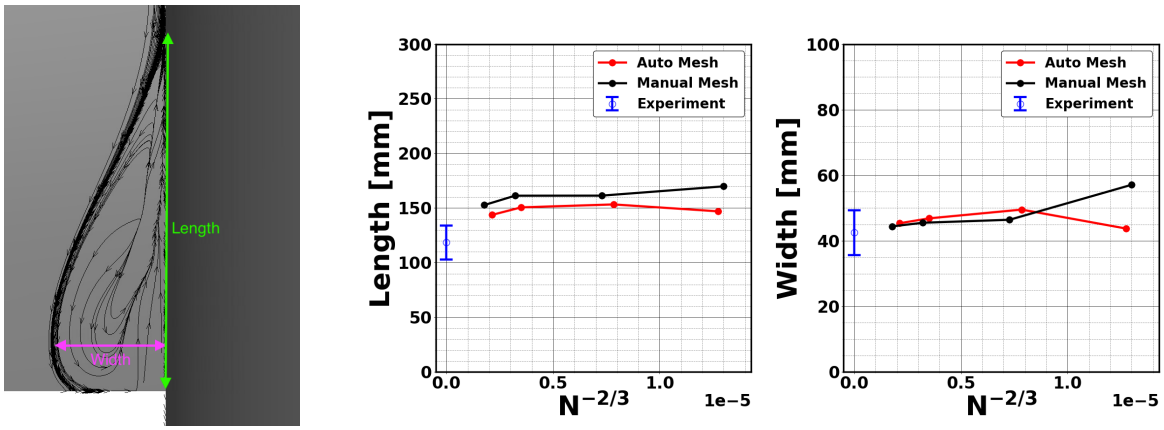
Figure 5.18 shows the convergence behavior for all four grids in the right-hand side residual as well as the force and moment coefficients. Grid levels coarse through fine were able to satisfy the convergence criteria for steady flow solutions. The extra fine case, however, exhibits a different convergence progression in comparison to the other grid levels. In Figure 5.18a, it can be observed that the extra-fine case initially exhibits a stalling in its residual convergence after a residual drop of 4.5 orders of magnitude at around 50K iterations. It was later determined that the meshes in the juncture region were stalling the residual convergence. Lowering the CFL for those specific grids allowed the residual to globally converge after restarting the case at around 55K iterations. At around 85K iterations, the extra-fine case is able to reach the same level of convergence as the other grid levels. It is possible that the local refinement in the juncture region at the extra-fine level may be exceeding some stability criteria. The other levels of refinement, however, exhibit ideal residual convergence behavior and are able to continue converging the solution past the five orders of magnitude drop without any stalling in the residuals. From examining the force and moment convergence plots, the initial transients in the flow solutions are eliminated at all levels approximately four thousand iterations into computing the solution. All grid levels are also able to exhibit steadying of the loads sufficiently below the load convergence criteria at the end of their respective runs.

Figure 5.19 shows the convergence of the lift, drag, and pitching moment coefficients under grid refinement. In all three coefficients, the solutions appear to exhibit a reduction in load changes between medium to fine and fine to extra-fine cases, indicating that the mesh is converging towards the solution obtained at infinite grid. In the lift (Figure 5.19a) and pitching moment (Figure 5.19c), a linear-like



(a) (b) (c)

Figure 5.19: Mesh convergence for the JFM-F6 computed on the automatically-generated grid family (N=total number of volume grid points). (a) Lift coefficient. (b) Drag coefficient. (c) Pitching moment coefficient.



(a) (b) (c)

Figure 5.20: Separation bubble measurements across grid levels in grid sensitivity study. (N=total number of volume grid points). (a) Diagram of separation bubble measurement at juncture region.. (b) Length measurement of separation bubble under grid refinement. (c) Width measurement of separation bubble under grid refinement.

convergence can be observed from the medium through extra-fine solutions, which suggests that those coefficients exhibit a second-order asymptotic convergence rate. The drag coefficient, on the other hand, starts off with a linear convergence profile before deviating from it at the extra-fine solution, indicating that the drag exhibits a mixed or lower than second-order convergence. Overall, the results provide confidence towards the capabilities of the automatic meshing tool in generating viable meshes for production use.

Figure 5.20 shows a comparison of the measurement of the separation bubble as the grid is refined using manual or EPOGS meshing approaches. Overall, both the manual and EPOGS approach exhibits the tendency to over-predict the length of the separation bubble, although the overall trend of the automatic meshes appear to over-predict it by slightly less. In the case of the bubble width, however, it appears that both the manual and automatic exhibit very similar trends, and are both approaching a width measurement within the range of experimental uncertainty.

5.2.2 Side-By-Side Hybrid Fuselage

The next test case is the Side-By-Side Hybrid (SBS) concept vehicle [125] fuselage from NASA's Revolutionary Vertical Lift Technology (RVLT) project. This geometry only contains the static components and does not include the two intermeshing rotors (see Figure 5.21). For this particular test case, only the comparison at the medium level refinement under the cruise condition investigated by Diaz et al [126] is considered. Table 5.18 details the flow

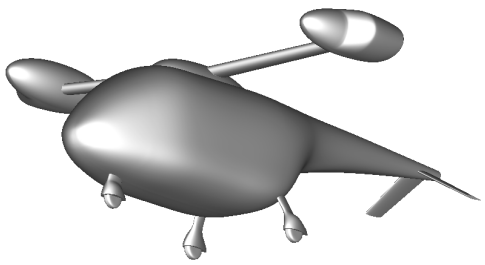


Figure 5.21: Side-By-Side Hybrid vehicle geometry.

Table 5.18: Cruise conditions for the SBS [126].

Flow Parameter	Altitude	T_∞	M_∞	Re
Value	5000 ft	ISA + 20°C	0.172	1.9×10^6

Table 5.19: Mesh parameters for both manual and automatic mesh generation (medium resolution) of the SBS.

Mesh Parameter	SR_{max}	Δs_{max}	np_{min}	Δs_{wall}	$\Delta\theta_{max}$	n_{fringe}
Value	1.16	0.2 ft	9	7.6×10^{-6} ft	4.0°	2

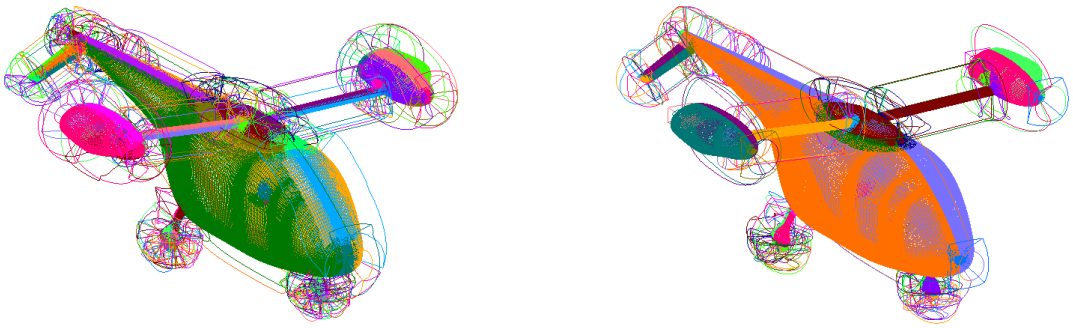
conditions. Table 5.19 lists the mesh parameters used to generate both the manual and automatic meshes.

Both manual and automatic meshes were generated by the authors for this test case. Table 5.20 shows a breakdown of the time spent at each stage in generating both meshes. The manual mesh required a larger proportion of the total mesh generation time in the surface meshing, while surface meshing only took a few minutes for the automatic tool. The automatic volume mesh generation took approximately 1.5 hours to complete, where most of the time was spent in the iterations for the LCBC procedure. A few meshes required some additional manual repairs to remove negative Jacobians. The domain connectivity required more time for the automatic mesh due to the fact that a handful of meshes required manual regeneration to improve the overlap between the meshes. Manual iterations with hole-cutting instructions also contributed to the time. It is interesting to note the large discrepancy in time needed to produce a surface mesh between the automatic approach and the manual approach. This was mainly due to the complex features on the vehicle, such as the presence of landing gears and engine fairings.

Table 5.20: Turnaround time to construct an overset mesh for the SBS using automatic and manual approaches for a novice to intermediate user, assuming an individual works 8 hours/day. Here, (C) is CPU wall-clock time, and (M) is manual labor time.

Steps	Automatic	Manual
Surface Meshing	4 min (C)	2-3 wks (M)
Volume Meshing	12 s (C)	1 days (M)
Volume Meshing LCBC	1.5 hrs (C)	-
Manual Mesh Repair	4 hrs (M)	-
Domain Connectivity	4 hrs (M)	2 hrs (M)
Flow Solver Inputs	10 min (M)	30 min (M)
Total Turnaround Time	$\sim$ 1 day	$\sim$ 2-3 wks

The automatic mesh consists of 40 face meshes and 120 edge and node meshes for a total of 160 meshes with 35.7M points. A total of 80 volume meshes were generated for the manual mesh with 36.5M points. Figure 5.22 shows the volume meshes generated from both processes. While there is



(a) (b)

Figure 5.22: Near-body volume meshes generated for the SBS vehicle. (a) Automatic. (b) Manual.

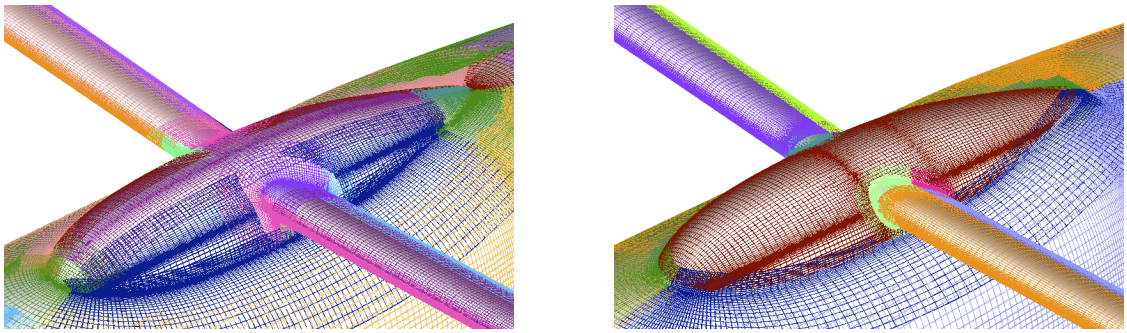
an increase in mesh count in the automatic mesh, it is not as significant of an increase as that of the JFM case. The increased quantity of features on the fuselage requires more meshes to be used in the manual approach.

Table 5.21 shows a quantitative breakdown of the grid composition for both grids. Overall, the meshes show comparable similarity in terms of grid size and point-type composition. The surface meshes show nearly similar grid point densities, which indicates that the two meshes are very close in terms of the surface resolution. As mentioned before, there are twice as many near-body meshes in the automatically-generated mesh. On the other hand, both meshes contain a similar number of off-body grids. A further examination of the grid points indicates a similar composition and structure to the off-body grids. This is expected, since the same off-body mesh generation and similar hole-cutting instructions are used.

The automatically-generated meshes exhibit ∼3% more fringe point overhead compared to the manual mesh. This is to be expected due to the fractured approach to mesh generation. However, the increase in fringe points is not excessive, and is within the variation that can be expected between different individuals that manually generate a mesh. Furthermore, the manual near-body meshes exhibits a slightly larger overhead in blanked points from the hole-cutting instructions. The total relative overhead for both meshes is ∼25% with a difference of only 0.1%.

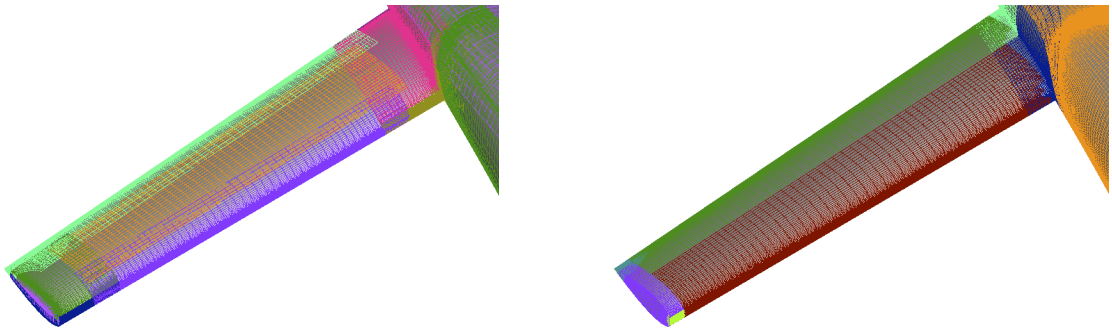
Table 5.21: Comparison of grid compositions between automatically and manually generated meshes for the SBS geometry.

	Automatic			Manual		
	Near-Body	Off-Body	Total	Near-Body	Off-Body	Total
Surface Points	497K	-	-	508K	-	-
Grid Count	160	252	414	80	248	328
Grid Points	35.7M	11.8M	47.5M	36.5M	11.7M	48.2M
Fringe Points	10.8%	7.4%	18.2%	8.6%	6.9%	15.5%
Blanked Points	5.1%	1.9%	7.0%	7.5%	2.3%	9.8%



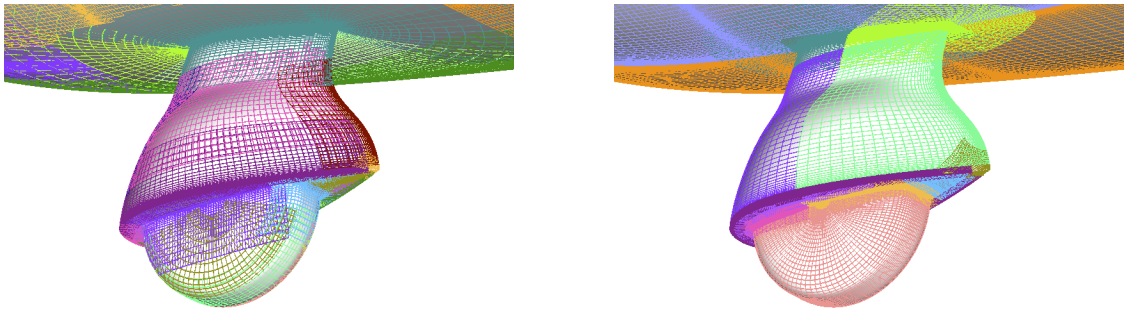
(a) (b)

Figure 5.23: Surface meshes generated for the Center Pod of the SBS vehicle. (a) Automatic. (b) Manual.



(a) (b)

Figure 5.24: Surface meshes generated for the right tail of the SBS vehicle. (a) Automatic. (b) Manual.



(a) (b)

Figure 5.25: Surface meshes generated for the front landing gear of the SBS vehicle. (a) Automatic. (b) Manual.

To complete the comparison, Figures 5.23, 5.24, and 5.25 show comparisons of between the surface meshes for various features on the vehicle, such as the center pod, right tail surface, and front landing gear, respectively. These figures indicate that, not only are the grids similar in size, but the grids are also generally similar in local resolution and refinement of the geometry. In all regions, the automatic mesh is much more broken up in comparison to the manual mesh. At certain locations, such as the front of the center pod and at the junction of the front landing gear, the automatic mesh is relatively coarser in

Table 5.22: Comparison of the effect of Overflow MPI load balancing (196 Broadwell cores) on automatically and manually generated meshes for the SBS geometry.

		Automatic	Manual
Before MPI-Splitting	Grid Count	414	328
	Total Points	47.5M	48.2M
After MPI-Splitting	Grid Count	838	823
	Total Points	53.9M	56.7M
	Chimera Boundary Exchange Time/Step (s)	4.10E-2	3.81E-2
	Parallel Efficiency	94.0%	91.0%
	Wall Time (62K iterations)	10.8 hrs	11.7 hrs

comparison to the manual mesh. Both meshes exhibit local refinement to reflect the intersection between the pod geometry and the motor struts. However, the junction between the center pod and the rest of the fuselage exhibits tighter refinement in comparison to the manual mesh. The tail surface mesh, on the other hand, appears to be relatively close in terms of the grid spacing between the two meshes. However, the collar mesh at the tail-fuselage junction on the automatic mesh appears to be relatively coarser than that of the manual mesh. The fuselage surface mesh surrounding the front landing gear in Figure 5.25a for the automatic mesh is also relatively coarser than that of the manual mesh.

Flow solutions at the cruise condition were then computed on the meshes. The third-order Roe scheme with Koren limiter is also used here, along with the SA-RC turbulence model. Time-stepping was also performed using the Pulliam-Chaussee scalar pentadiagonal scheme with low-Mach preconditioning. For this particular case, 196 Broadwell cores were used on both meshes. Table 5.22 shows how the grids compare when Overflow performs the MPI splitting. Both grids are split into a similar quantity of grid chunks, and the grid point count after the splitting is also similar in quantity. The difference in the split grid size is around 5%. It is interesting to observe that the splitting has resulted in the manual mesh exhibiting a greater increase in point count compared to the automatic mesh in absolute and relative terms to their respective initial grid point counts. Again, this may be attributed to the larger individual grid sizes of the manual mesh, since a manual approach may attempt to represent the surface with as few individual meshes as possible. As in previous cases, the manual mesh grid count is similar to that of the automatic mesh grid count after MPI splitting, demonstrating a comparable level of fracturing. It can also be observed that the time spent performing the exchange of data across overlap boundaries is approximately the same between the two meshes.

The convergence behavior of the two grids can be observed in Figure 5.26. Figure 5.26a reveals that both solutions achieve residual convergence at roughly around 15K iterations. The force and moment

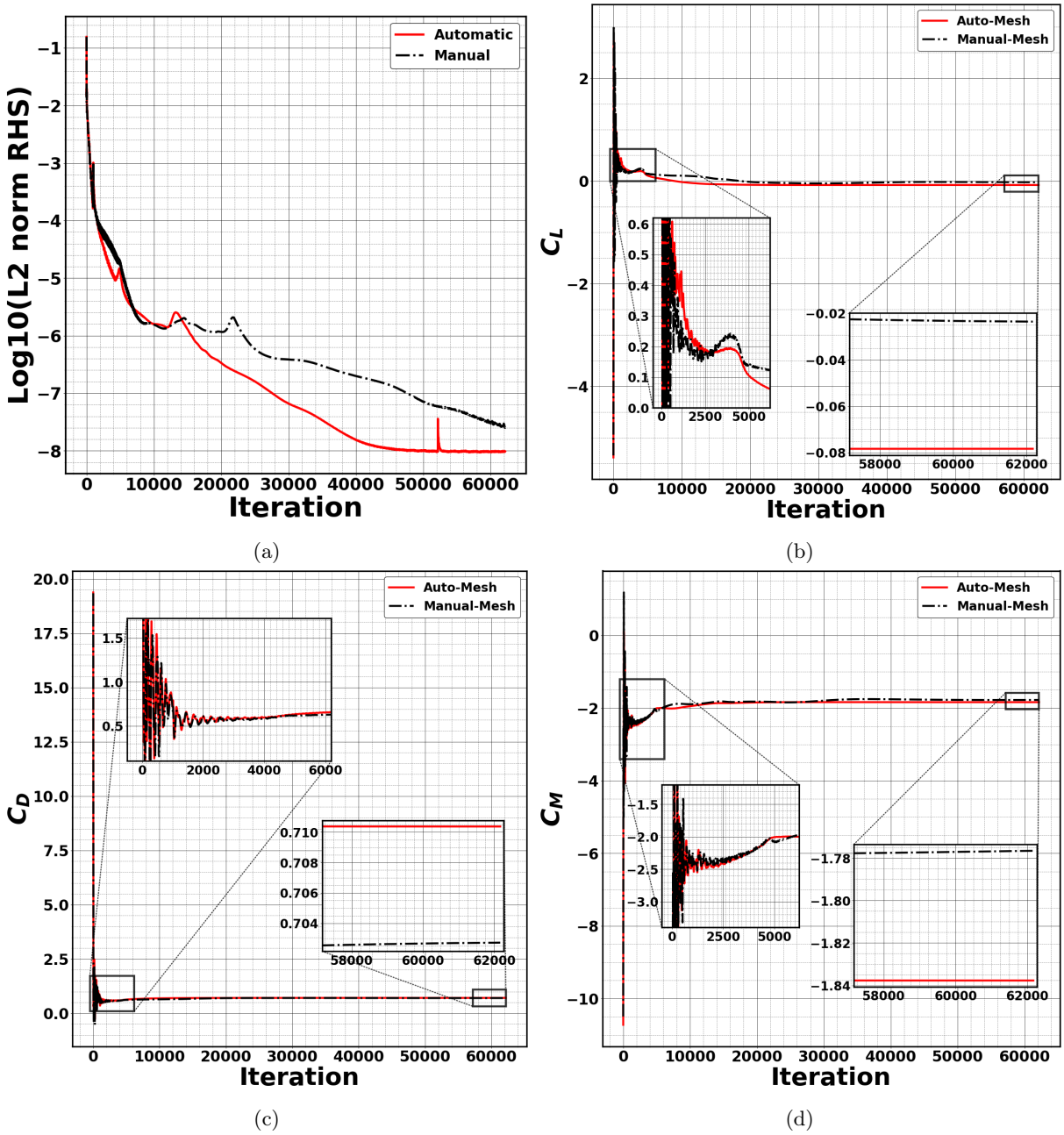


Figure 5.26: Solution convergence behavior for the SBS computed on the automatic and manual meshes.(a) Residual. (b) Lift coefficient. (c) Drag coefficient. (d) Pitching moment coefficient.

histories (Figures 5.26b-5.26d) also show that the initial large transient oscillations damp out at around roughly the same number of iterations for both grids. The remainder of the iterations for both meshes is to converge the load coefficients. During this phase of the convergence, the manual meshes exhibit a slower residual drop compared to the automatic meshes. Both solutions are able to achieve a residual convergence drop beyond six orders of magnitude. In the last few thousand iterations, both meshes exhibit force and moment coefficient variations that satisfy the convergence criteria for vehicle loads.

Table 5.23 shows a comparison of the final force and moment coefficients, and the third row shows the percentage deviation of the solution computed on the automatic mesh compared to that of the manual mesh. For the drag and pitching moment coefficients, the variations between the two solutions fall below 5%. The lift coefficient, however, shows the largest relative difference of the three loads. It is important to note that the lift coefficients are very small to begin with and thus, it may be more useful to consider the absolute differences between the computed loads instead. In this case, we can see the lift coefficient only differs in the hundredths-place. Furthermore, like the Lift+Cruise geometry, the coefficients are computed with a reference area of $1[m^2]$, which is the methodology for RVLT vehicle analyses at the time of the writing of this manuscript. Thus, the absolute difference may actually be much smaller by nearly an order of magnitude when scaled by a more physically-relevant reference area. With this consideration in mind, the difference in lift is likely to be within the range of acceptable variations in the results.

Table 5.23: Comparison of the converged aerodynamic loads for the SBS between the manual and automatic meshes.

	C_L	C_D	C_M
Manual Mesh	-0.0229	0.7030	-1.7781
Auto mesh	-0.0789	0.7113	-1.8387
$\|\Delta\|$	0.056	0.0083	0.0606
% Difference	245%	1.18%	3.41%

Figure 5.27 shows the pressure coefficient cuts at various locations along the strut fairing supporting

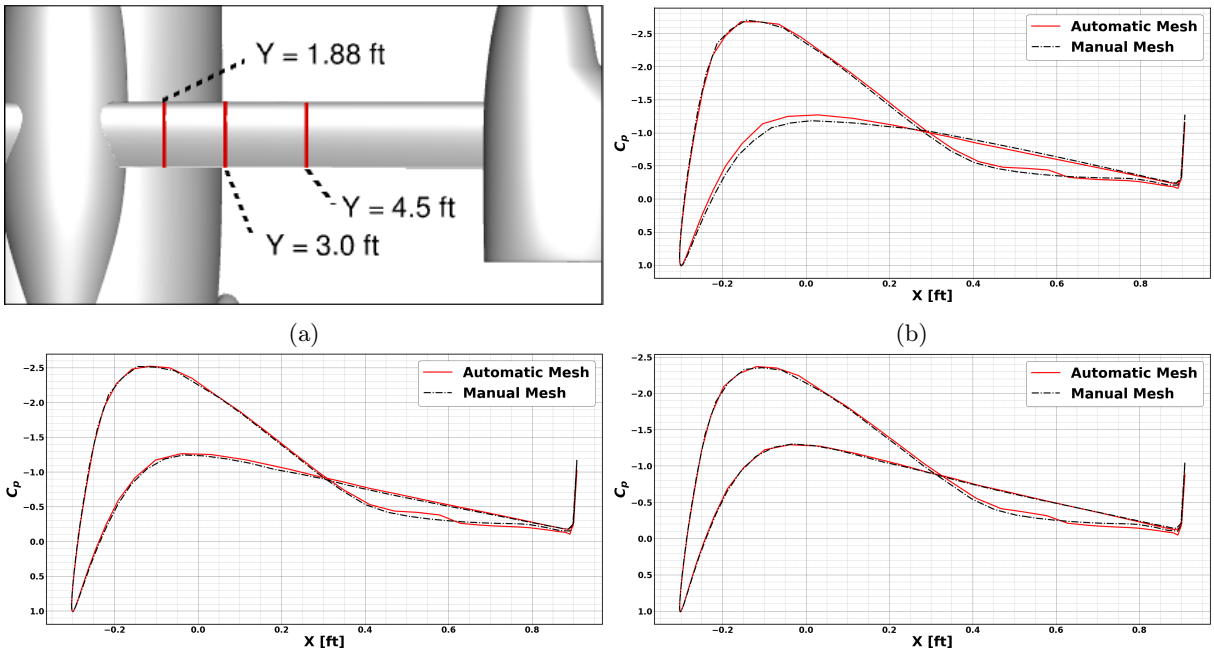
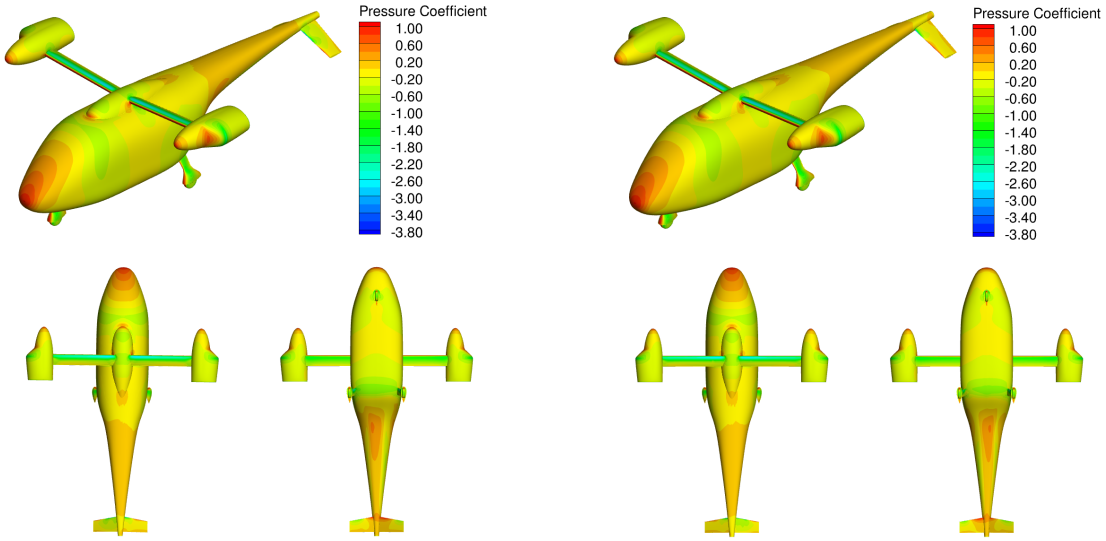


Figure 5.27: Pressure-coefficient cuts on the SBS strut .(a) Pressure-cut locations. (b) Y=1.88 ft. (c) Y=3.0 ft. (d) Y=4.5 ft.



(a) (b)

Figure 5.28: Surface pressure coefficient of the SBS in forward flight. (a) Automatic. (b) Manual.

the right engine pods. Overall, the two meshes show good agreement in capturing the suction peak and adverse pressure gradient over the upper surface of the strut. In some locations, however, there appears to be a slight discrepancy in the flow acceleration on the lower surface leading edge. The discrepancy tends to be larger at cuts closer towards the junction between the strut and fuselage (Figure 5.27b). The cuts near the mid-span region tend to show less differences in the leading edge region (Figures 5.27c, 5.27d). In the back-half of the pressure cut, the upper surface exhibits a decline in the adverse pressure gradient. Consistently across the cuts, however, the automatic mesh appears to exhibit slight deviations from the manual mesh all the way up to the trailing edge. These discrepancies can potentially be traced back to the differences in the construction of the automatic meshes in representing the geometry surface.

Figure 5.28 shows the pressure coefficient contours on the surface of the vehicle. In the front half of the vehicle from the rear landing gears, the two meshes show good agreement in variations of the surface pressure. In the rear half of the vehicle, however, a left/right asymmetry can be observed on the empennage. The cause of the asymmetry is likely due to the choice of turbulence model and massive separation in the region. Because the tail surfaces are downstream of these features, the computed behavior over these surfaces should be treated with some skepticism. Nevertheless, the effect on the aerodynamic loads is small. Further investigation of this anomaly is beyond the scope of the current work and will be left for future study. It is possible that the large recirculation behind the fuselage and underneath the empennage may actually require unsteady RANS or some form of scale resolving simulation to obtain the proper results.

5.2.3 Drag Prediction Workshop 7 Common Research Model

This test case is the High-Speed Common Research Model (CRM) [127] under Case 2b of the 7th AIAA Drag Prediction Workshop (DPW-7) [128]. Case 2b from the workshop consists of a set of angle-of-attack sweeps at specified angles with corresponding aero-elastic wing deflections. For the purposes of verification, only the case at $\alpha = 3.00°$ is considered. Table 5.24 shows the flow conditions for the case computed in free-air.

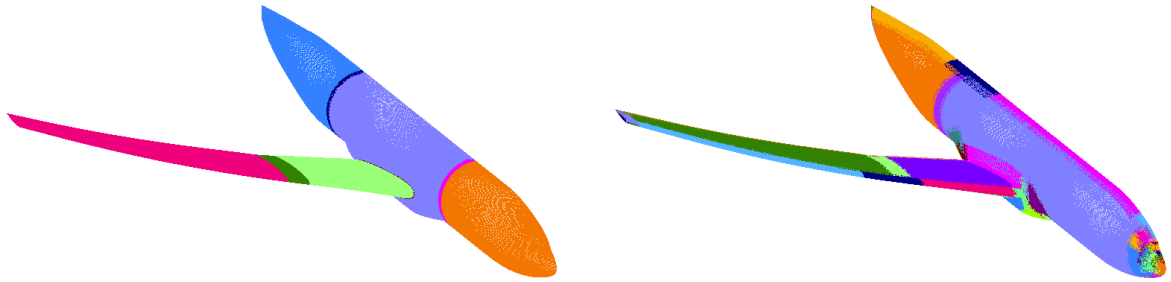
Table 5.24: Flow for conditions the High-Speed CRM, Case 2b in free-air at $\alpha = 3.00°$.

Flow Parameter	M_∞	α	Re	T_∞
Value	0.85	3.00°	5M	$100°F$

Two sets of overset grids were generated using the EPOGS utility at a "coarse" and "fine" grid level. The input parameters are given in Table 5.25. For the coarse mesh, EPOGS fully automated the mesh generation and flow case setup for Overflow. With the fine mesh, on the other hand, only the near-body meshes were generated. The off-body meshes and domain connectivity were performed using Overflow. The turn-around time to generate both sets of meshes and set up a CFD case takes approximately 4 hours. This also includes time required for manual repair of a handful of meshes to improve cell quality and overlap quality. Both sets of grids are able to be generated on a MacBook Pro with M2 processor and 12 OpenMP threads. Overall, this is a significant improvement in turn-around time, which would normally require a user a few days to produce a useable mesh and properly set up a simulation. Figure 5.29 shows a comparison of the automatically-generated meshes from EPOGS and the manually-generated overset meshes originally provided by the workshop. It is immediately apparent that the automatic coarse level mesh is significantly more fractured than the manually-generated mesh. Table 5.26 shows a breakdown of the grid composition for automatic grids and the overset grid provided by the workshop. In terms of the surface mesh resolution, the coarse mesh comes out to be coarser than the original workshop medium level mesh in the near-body resolution, while the fine mesh is more comparable to the medium level resolution. The workshop grids extend from the surface of wing-body all the way to the outer far-field boundary, so there are no off-body grids in the workshop grids.

Table 5.25: Mesh parameter inputs for the EPOGS-generated grids for High-Speed CRM, Case 2b, $\alpha = 3.00°$.

Mesh Parameter	SR_{max}	Δs_{max}	np_{min}	$\Delta\theta_{max}$	n_{fringe}
Coarse	1.2	10.0 in	13	4.0°	2
Fine	1.1	5.0 in	17	4.0°	2



(a) (b)

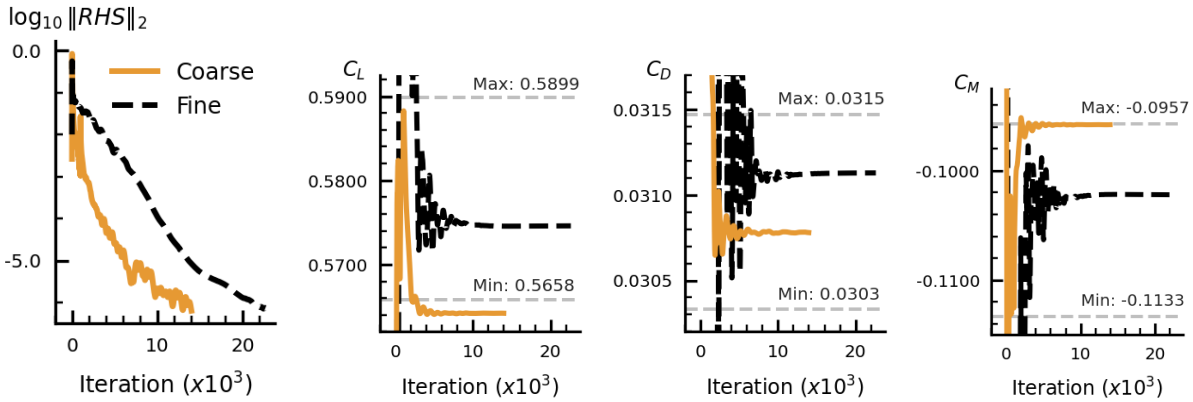
Figure 5.29: Comparison of (a) manually-generated meshes and (b) automatically-generated coarse-level mesh for the high-speed CRM at $\alpha = 3.00°$.

Table 5.26: Comparison of Coarse, Fine, and workshop-provided grid compositions generated for High-Speed CRM, Case 2b, $\alpha = 3.00°$.

	Coarse			Fine			Workshop
	Near-Body	Off-Body	Total	Near-Body	Off-Body	Total	Total
Grid Count	70	1	71	73	379	452	9
Grid Points	13.4M	42.9M	56.3M	66.1M	17.1M	83.2M	43.3M

Steady-state RANS solutions are then computed on both sets of grids. The equations are discretized using a second-order central differencing scheme with variable artificial dissipation, and the ARC3D scalar pentadiagonal scheme is used to advance the solution in pseudo-time. The Spalart-Allmaras turbulence model with curvature correction (RC) and QCR2000 is used to model the turbulence. This scheme is selected to produce a numerically-comparable flow solution with the Overflow submission in the original set of DPW7 solutions. The solutions are then iterated until the residual exhibits at least a five-order magnitude reduction, the lift and pitching moment coefficients vary less than 0.001, and the drag coefficient varies less than a single count of drag.

Figure 5.30 shows the convergence behavior of the steady RANS solutions computed on the two grid levels generated by EPOGS. In both sets of grids, the solutions exhibit relatively monotonic convergence as the residual is reduced by five-orders of magnitude. The coarse case shows the load coefficient transients being damped within the first several thousand iterations, while the fine case requires nearly 8000 iterations before the large transients are damped. In the case of the coarse mesh, the resulting drag coefficient is able to lie within the spread of values obtained by the participants, while the lift and pitching moment coefficient are at the outer bounds or just outside the values of the participants. However, by refining the mesh using EPOGS, the finer grid is able to produce load coefficients that all consistently lie within the range of solutions obtained by the participants.



(a) (b) (c) (d)

Figure 5.30: DPW7-CRM solution residual and aerodynamic loads convergence history for both coarse (orange) and fine (black dash) meshes. Minimum and maximum load coefficients from DPW7 participants [128] indicated in gray. (a) Residuals (L2-norm of right-hand side). (b) Lift coefficient. (c) Drag coefficient. (d) Pitching moment coefficient.

Table 5.27: Integrated load contributions of CRM components comparisons with original submissions[1] to AIAA DPW7 for Case 2b, $\alpha = 3.00°$.

	Total			Wing			Fuselage		
Solver	C_L	C_D	C_M	C_L	C_D	C_M	C_L	C_D	C_M
Coarse	0.5642	0.0308	-0.0959	0.4903	0.0193	-0.1459	0.0738	0.0115	0.0501
Fine	0.5746	0.0311	-0.1022	0.5000	0.0195	-0.1525	0.0746	0.0116	0.0502
DPW7 Min	0.5658	0.0303	-0.1133	0.4918	0.0189	-0.1635	0.0740	0.0114	0.0500
DPW7 Max	0.5899	0.0315	-0.0957	0.5146	0.0199	-0.1460	0.0753	0.0115	0.0503
DPW7 Overflow	0.5658	0.0305	-0.0957	0.4918	0.0189	-0.1460	0.0740	0.0115	0.0503

Table 5.27 shows a breakdown of the load coefficients into contributions from the wing and fuselage components. In both meshes, the wing drag contribution is within the range of workshop results, while the fine mesh fuselage is just slightly greater than the range of results. On the other hand, the coarse mesh underpredicts the wing lift contribution over 10 lift-counts below the minimum value from workshop. This appears to be the main source of discrepancy in the total lift coefficient, along with the slight overprediction of the wing pitching moment coefficient. However, it should also be noted that the coarse level actually demonstrates a very good comparison with the original Overflow submission to the workshop, with at most a 2% difference across all loads. The largest discrepancy is the wing lift at approximately 15 counts of lift. Nevertheless, this large discrepancy from a relative perspective is less than 0.5% in loading. The fine level grid, on the other hand, shows an ability of the automatic meshes to produce solutions

[1]Data for integrated loads kindly provided by Ben Rider from The Boeing Company.

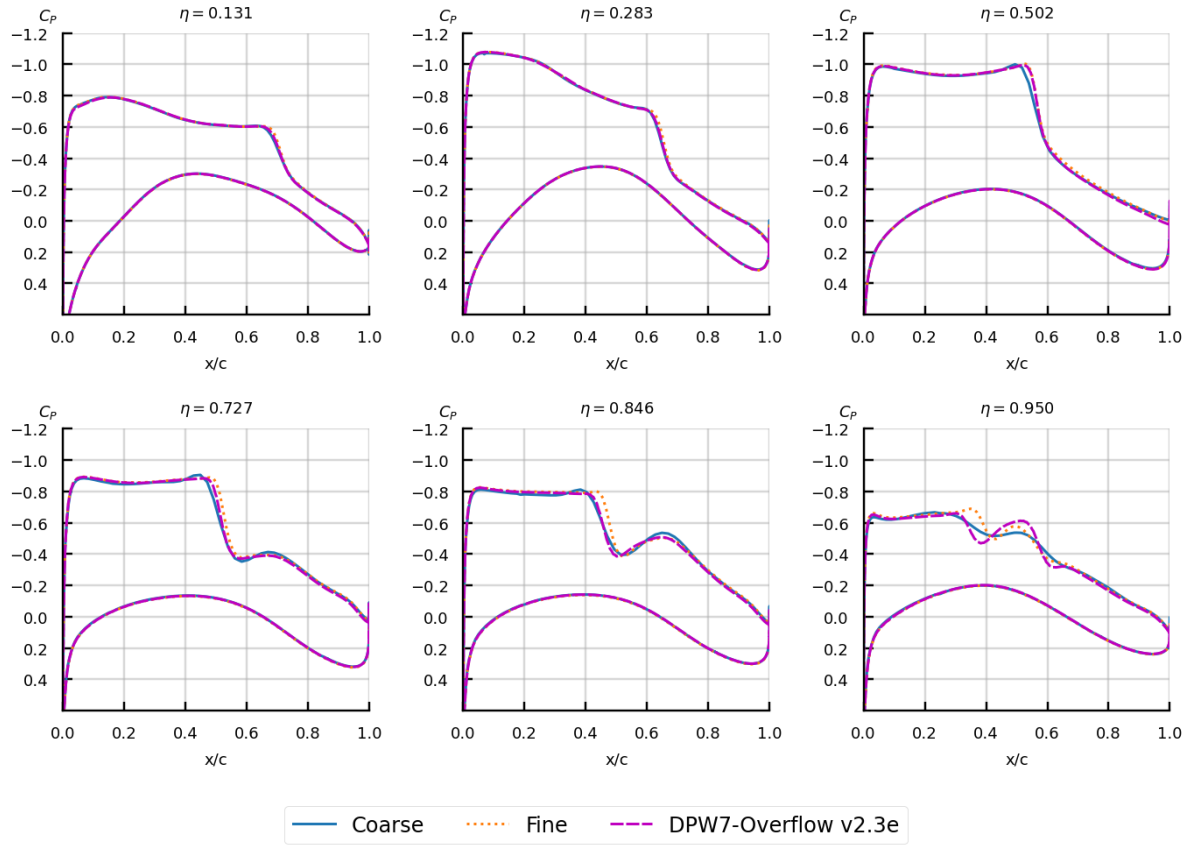


Figure 5.31: Surface pressure cuts[2] along the main wing of Case 2b, $\alpha = 3.00°$.

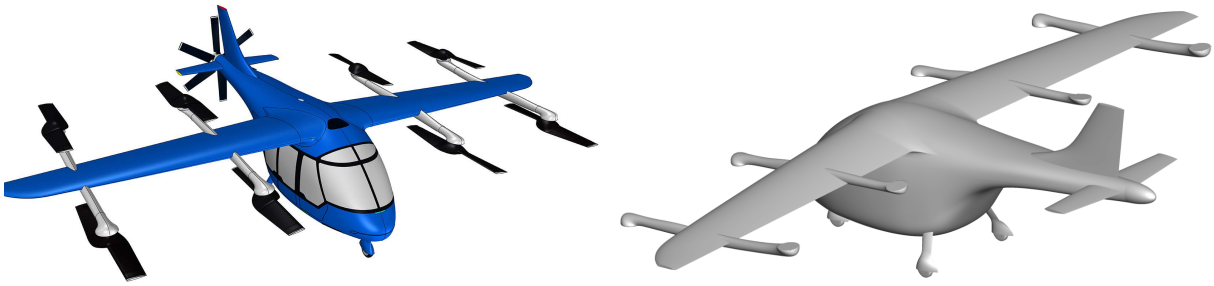
that exhibit good agreement with other solvers.

Figure 5.31 shows a comparison of the surface pressure cuts of the coarse and fine solutions with the Overflow solutions submitted in the workshop. It should be noted here that the original workshop submission uses an older version of Overflow (version 2.3e), while the solutions computed on the EPOGS-generated grids uses Overflow version 2.4c, which is the current version at the time of writing. Overall, the inboard cuts show that both grids have very good agreement with the workshop results. The coarse grid is able to maintain relatively good agreement with the grid up until outermost cut ($\eta = 0.950$). At the wing tip, the flow forms a lambda shock over the upper surface. Here, the coarse grid actually shows good agreement in terms of the actual shock *positions*. However, the shocks are significantly smeared in comparison and does not pick up the low pressure peak between the shocks. This is possibly due to coarser grid resolution at the tip region. On the other hand, the fine grid predicts shock positions that are further downstream at the outer two cuts ($\eta = 0.846, 0.950$). In the lambda shock, the fine grid shows a first shock that is further downstream, and is also able to capture a low pressure peak between the shocks. The latter shock shows good agreement with the workshop submission.

[2]Data for surface pressure cuts kindly provided by Ben Rider from The Boeing Company.

5.3 Lift+Cruise Static Component Study

The final test case is a demonstration of how the EPOGS automatic mesh generation tool can reduce turn-around time for high-fidelity RANS flow analysis with structured overset grids in a design application. The particular geometry in consideration is the Lift+Cruise (L+C) concept vehicle [106] from the NASA RVLT Project. The following discussion consists of the author's contributions to recently published conference papers at the 6th Decennial Aeromechanics Specialists' Conference by Hosseini et al [109] and the 2024 AIAA Aviation Forum by Chan et al [83].

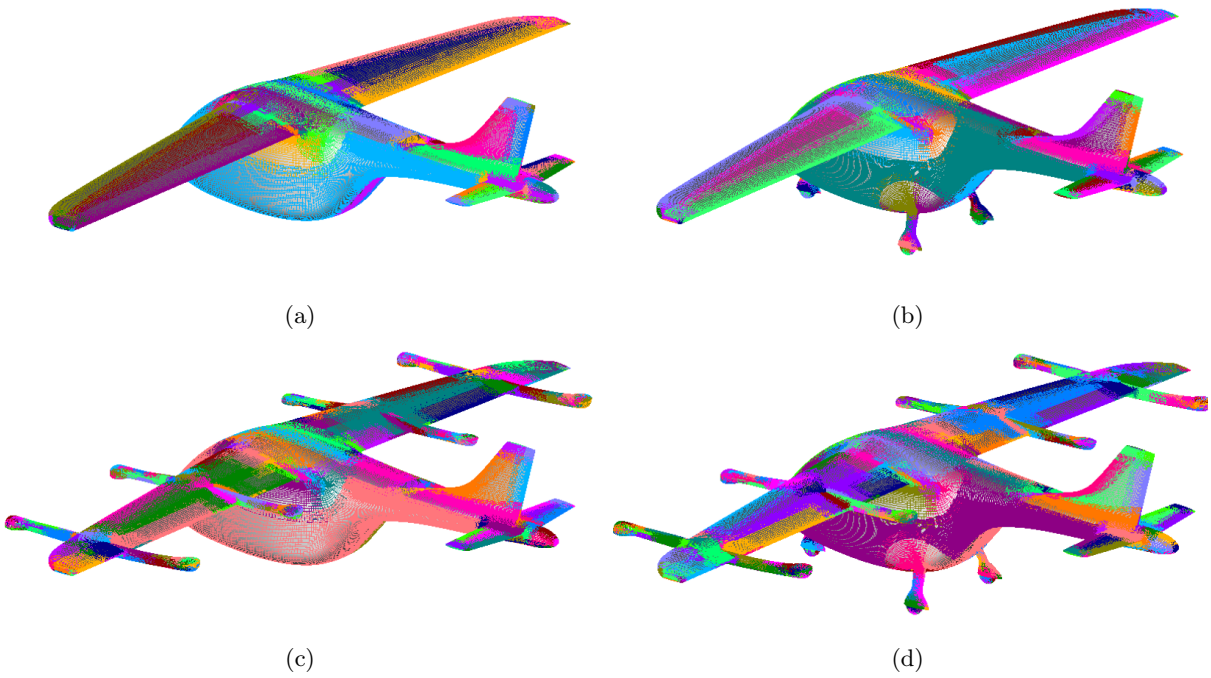


(a) (b)

Figure 5.32: Lift+Cruise concept vehicle. (a) Rendering of the concept vehicle. (b) Geometry of analyzed full configuration (note the lack of lifting rotors and tail propeller blades).

Figure 5.32 shows a 3D CAD rendering of the concept vehicle design, which exhibits a particularly complex configuration due to the quantity of protruding components. This includes the landing gear and the wing-mounted pylons, which house the powertrains that drive the rotors for VTOL operations, or the "lift" portion of its flight profile. It is very likely that the flow around the vehicle components will exhibit complex aerodynamic behavior as well as increased parasitic drag. Characterizing the effect of the landing gear and pylons can be done feasibly with CFD. In structured overset simulations, however, a major impediment of the analysis procedure is generating a suitable near-body mesh. This process can require a significant amount of user time and effort depending on the complexity of the geometry. With the EPOGS tools, of course, the effort and time can be reduced.

The main interest of this study is to characterize the effects of the *static* components, which consists of the entire geometry without the rotors and tail propellers, on the aerodynamic performance (i.e. lift, drag, pitching moment) in cruise flight. Analysis of the vehicle with moving components (i.e. with rotors and tail prop) will be the subject of a future study. Four different configurations are considered here. Figure 5.32b shows the *baseline+landing gear+pylon* configuration, where the *baseline* configuration only consists of the main wing, empennage, and fuselage. The other two configurations considered are the *baseline+landing gear* and *baseline+pylons* geometries, which allow for the incremental analysis of the landing gear and pylons, respectively. Figure 5.33 shows the surface meshes generated with EPOGS for the four analyzed configurations. Table 5.28 details the EPOGS inputs to generate meshes for each configuration. This set of inputs were obtained through a grid sensitivity study originally performed in



(a) (b) (c) (d)

Figure 5.33: L+C configurations under study. (a) Baseline. (b) Baseline + Landing Gear. (c) Baseline + Pylons. (d) Baseline + Landing Gear + Pylons.

Ref. [108]. The same inputs are used for all configurations—EPOGS utilizes its own internal component scaling to perform component-based refinement as necessary. Concerns with numerical accuracy can be addressed by repeating a similar grid sensitivity study or by performing a grid convergence study using the same analysis procedures performed for the benchmark cases in Section 5.1. However, this is beyond the scope of this study.

Table 5.28: Mesh parameter inputs for generating configurations of the Lift+Cruise vehicle. All configurations utilized the same input parameters.

Mesh Parameter	SR_{max}	Δs_{max}	$\Delta\theta_{max}$	Δs_{wall}	n_{fringe}
Value	1.13	0.33 ft	3.0°	4.10×10^{-6} ft	2

Table 5.29 shows a comparison of the turnaround times for using an EPOGS-aided mesh generation approach and manual meshing. Here, mesh repairs can refer to fixing poor grid quality issues that can occur within EPOGS, such as grids with negative Jacobians/cell volumes, or poor connectivity between grids. Normally this is only necessary for a handful grids, depending on the complexity of the geometry or CAD model provided (the reader is referred to Table 2 in [83] for the latest grid quality statistics of EPOGS). Across all configurations, there is a significant time reduction in using EPOGS to set up a case from a solid CAD model to starting a CFD run in Overflow. The setup for all configurations can be performed with EPOGS in the same amount of time it takes to manually generate a mesh and

Table 5.29: Turnaround time for automatic and manual mesh generation of all four L+C configurations (B=Baseline,P=Pylons,LG=Landing Gear). The manual meshing times are obtained from approximations from previous experience with similar geometries for intermediate to advanced users of manual mesh generation [83].

	EPOGS				Manual Meshing
Configuration	Surf+Vol Gen	Domain Conn.	Mesh Repairs	Total Time	Total Time
Baseline	4 min 20 s	30 min	∼1 day	∼ 1 day	∼ 1 wk
B+LG	9 min 55 s	2 hrs	∼1 day	∼ 2 days	∼ 2 wks
B+P	6 min 48 s	1.2 hrs	∼1 day	∼ 2 days	∼ 2 wks
B+P+LG	11 min 25 s	∼2 hrs	∼1 day	∼ 2 days	∼ 3 wks

setup a CFD run for any one of the configurations with additional components. This demonstrates a marked reduction in time to setup structured overset CFD simulations across multiple different geometry configurations.

Steady RANS flow solutions are then computed using Overflow on each of the static configurations. Table 5.30 details the flow parameters for the Lift+Cruise at cruise flight. Here, each run is computed using the third order Roe scheme with Koren limiter and SA-RC turbulence model. Solutions are advanced using SSOR with low-Mach preconditioning to help accelerate the solver until steady-state convergence criteria is achieved.

In general, each case requires approximately 30 to 50 hours of wall time to reach a steady state convergence. If resources are available, all four configurations can be computed in parallel, meaning that the total turnaround in wall time can be approximately 50 hours. With the assistance of the autoemeshing tools, the analysis can potentially be completed in the span of two "business" 'weeks (assuming only five 8-hour work-days are in each week). In comparison, performing this analysis through the manual meshing approach may require four business weeks, assuming the user re-uses the same meshes for similar surfaces across the configurations. This includes the three weeks to generate the full landing gear+pylon configuration, as well as an extra few days to generate the other configurations through elimination of components, and another two days to run the flow solutions. Thus, the autoemeshing tools allows for a nearly 50% reduction in user time to perform the analysis. Continued improvements of the tool should allow for additional reductions in user time.

Table 5.30: Flow conditions for the Lift+Cruise in cruise flight.

Flow Parameter	M_∞	α	Re	T_∞
Value	0.1189	0.0°	2.9M	$77.2°F$

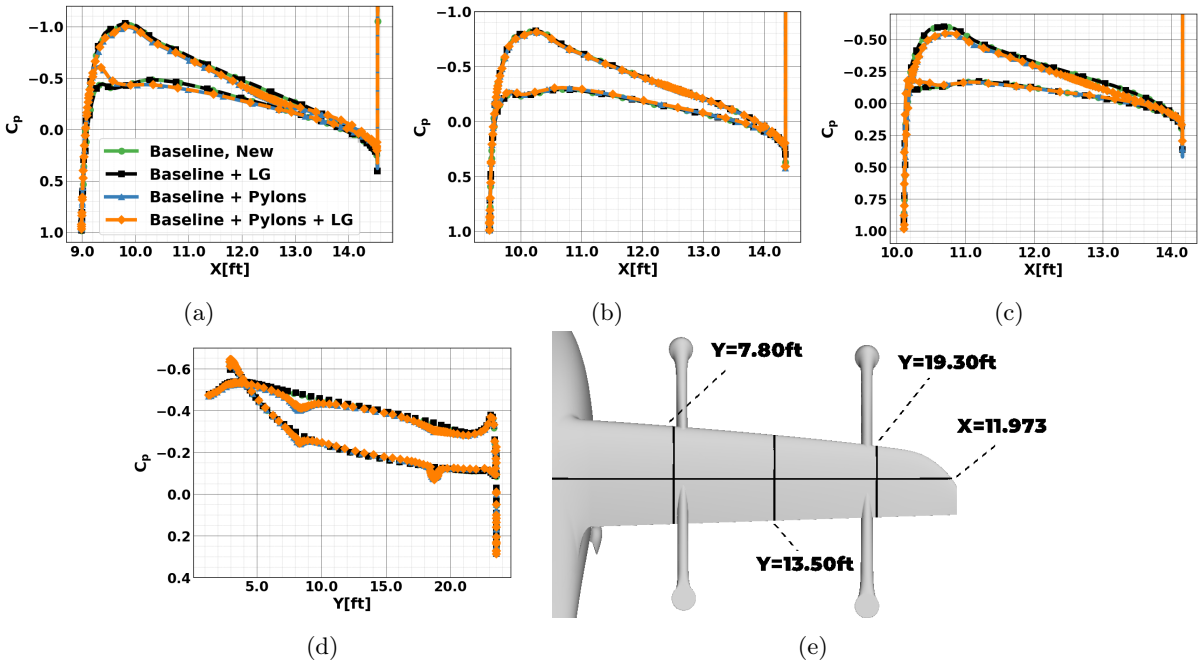


Figure 5.34: Surface pressure cuts on the right wing for each configuration. (a) $Y = 7.80ft$. (b) $Y = 13.5ft$. (c) $Y = 19.3ft$. (d) $X = 11.973ft$. (e) Locations of C_P cuts.

Figure 5.34 show the surface pressure cuts along the main wing for all four configurations. Two cuts are taken near the pylon locations, with an additional cut half-way between the pylons. A fourth cut is taken approximately mid-chord across the span of the wing. From the various plots, it can be observed that the inclusion of the pylons in the configuration appear to have the largest impact on the pressure profile, especially in regions closest to the pylons. In the cuts near the pylons (see Figure 5.34a and Figure 5.34c), the inclusion of the pylons results in a reduced low-pressure region over the suction side of the wing, with overall lower suction peaks. A smaller reduction can be seen in Figure 5.34b, while Figure 5.34d shows pressure divots at the pylon locations. The landing gear, on the other hand, show little overall effect on the surface pressure of the wing between the baseline and baseline+landing gear case, as well as the baseline+pylon and baseline+pylon+landing gear case. This should be expected since the landing gear are underneath the fuselage, and thus are less likely to significantly affect the flow over the main wing.

Table 5.31 shows the total integrated loads (lift, drag, and pitching moment), as well as the *deltas* from the baseline case for each configuration. There appears to be a slight *superposition* effect, where the presence of each individual component on the geometry has almost a linear compounding effect on the total integrated loads. The effect of the landing gear appears to have a slight increase across all load coefficients by approximately 3 to 7%. The relatively minimal drag increase is likely due to the aerodynamic fairings surrounding the landing gear. The lift and pitching moment increases can be attributed to the dihedral angle present in the rear landing gear. On the other hand, the pylons appear

Table 5.31: Component build-up on aerodynamic loads and moments. Here, Δ indicates the difference of each configuration from the Baseline case.

	C_L	ΔC_L	C_D	ΔC_D	C_M	ΔC_M
Baseline	0.1507	—	0.0276	—	−0.0442	—
B + LG	0.1593	+0.0086 (+5.71%)	0.0284	+0.0008 (+2.96%)	−0.0409	+0.0033 (+7.42%)
B + P	0.1245	−0.0262 (−17.4%)	0.0327	+0.0051 (+18.6%)	−0.0301	+0.0141 (+31.9%)
B + LG + P	0.1315	−0.0192 (−12.7%)	0.0328	+0.0052 (+18.8%)	−0.0276	+0.0166 (+37.6%)

to both increase the drag and decrease lift by approximately 18%, and increase the pitching moment by 32%. The resulting changes in lift can be attributed to the reduced gradient over the suction side of the main wing, as seen in Figure 5.34. The increased drag can be attributed to the increased profile drag and viscous drag from the presence of the pylons in the flow.

Chapter 6

Conclusions and Future Work

In this present work, different aspects of automatic structured overset mesh generation have been considered. These aspects spanned topics such as algorithmic developments, numerical methods for grid generation, and applications in CFD and aerodynamics analysis. The breadth of topics are due to the fact that developments in automatic overset mesh generation are still nascent, and enhancements to existing methodologies are necessary to support the goal of automation. Additionally, the proposed meshing approach also involves a relatively unorthodox divergence from typical overset CFD simulation methodologies, and as such requires additional study to characterize any downstream effects that the mesh generation has on the ultimate goal of performing flow simulation.

In Chapter 4, a boundary coupling approach utilizing overset boundaries have been presented as a method for maintaining appropriate overlap between hyperbolically-generated meshes with overlapping initial data. Mathematical arguments based on properties of the initial geometry that are parameterization independent demonstrate the compatibility of the boundary condition for the hyperbolic mesh generation equations. These arguments were found to apply for all variations of the hyperbolic mesh generation equations (two-dimensional field, surface, and volume). An overview of the numerical implementation was also presented, with two different types of coupling (loose and tight) between the different grids. The overall numerical implementation follows that of overset boundaries typically used for overset CFD simulations. Applications to several geometries show that the coupling maintains the overlap between the generated surface or volume meshes such that the overlap is consistent with the overlap present within the initial curves or surfaces, respectively.

Future efforts should focus on improving the robustness of the boundary coupling. Such efforts can be separated into considerations for the initial data and considerations for the solution of the hyperbolic PDE itself. In the case of the initial data, a "reconstruction" strategy was introduced as a necessary step for mesh coupling when generating volume meshes. It is critical to ensure that the logically-rectangular parameterizations lie completely within the bounds of the geometry itself. While this may work for

simple situations where a face is bounded by two to four edges, or in regions where a logically-rectangular decomposition may be intuited, a more general approach will eventually be needed to handle arbitrary face geometries that can be bounded by an arbitrary number of edges or exhibit shapes not ideal for logically-rectangular parameterizations. In the case of solving the hyperbolic PDE, other areas with room for improvement or additional development efforts include the enhancement of the transfer of data between grids in the tight-coupling approach. The "lagging" of the boundary data can potentially be augmented through incorporation of other known data transfer algorithms when solving PDEs, such as Restrictive Additive Schwarz. This may allow for further enhancements of convergence at each grid level and reduce the wall time for the combined grid generation.

In Chapter 5, RANS applications for various configurations were explored through computing flow solutions on the grids generated using EPOGS. Through three-dimensional RANS benchmark cases, it was found that the "fracturing" that is characteristic of automatically-generated meshes from EPOGS has little impact on the accuracy of the numerical solution and the grid convergence behavior. Spurious behavior regarding the convergence of some loads appear to be common across all meshing approaches that have also been compared. Larger configurations have also been considered, which also demonstrate that the fracturing has minimal effect on accuracy of computations while also having a neutral effect on the computational load. Future work should consider higher-fidelity flow computations, such as large-eddy simulation or hybrid variants of RANS and LES. The different treatment and fidelity of the Navier-Stokes equations typically result in more stringent meshing requirements, which may serve to provide essential feedback for further advancements in the automation methodology. Whereas the fracturing may be acceptable for RANS solutions, it remains to be seen how the fracturing will fare with the higher-fidelity approaches, or even other computational analyses such as aero-acoustics. It is possible the interpolation schemes at the overset boundaries can interfere with the higher-frequency fluctuation content of LES or aero-acoustic simulations. As such, it may be necessary that future efforts should be focused on improving interpolation schemes between grids. These efforts can also include exploring other methods that can enable connectivity between grids in flow solutions while simultaneously avoid issues with conservation and misaligned overlap. This can include incorporating unstructured or mesh-free methods to connect all the various blocks together. Additionally, it may be imperative that alternative domain decomposition approaches that can minimize the fracturing be explored.

Finally, more general efforts can be considered to improve the automatic mesh generation process overall. The main issue that is still plaguing mesh generation in general is elimination of negative Jacobians or negative cell volumes in hyperbolic grid generation. The current methods available cannot guarantee elimination of or minimize the presence of these low grid quality features. From a computational perspective, another area of improvement can also involve the refactoring of algorithms to properly utilize parallelization for reducing turnaround times or handling much larger configurations.

Appendix A

Uncertainty Analysis for CFD Grid Convergence Studies

The uncertainty analysis outlined in this section originates from the analysis procedure outlined by the Fluids Engineering Division of ASME [114], as well as some additional considerations from the verification procedure from the National Program for Applications-Oriented Research in CFD (NPARC) Alliance [129] between NASA Glenn Research Center (GRC) and the Arnold Engineering Development Center (AEDC). This approach is based on Richardson extrapolation to perform the error estimation of a grid convergence study on the finest three grid levels of a grid family.

The first step is to define a characteristic edge length h_i for each ith grid level in a grid family. For a three-dimensional grid, this can be computed using

$$h_i = \left(\frac{1}{N_i}\right)^{1/3}, \tag{A.1}$$

where N_i is the number of grid points for the ith grid. The ratio r between grid levels is given by

$$\begin{aligned} r_{21} &= h_2/h_1 \\ r_{32} &= h_3/h_2 \end{aligned} \tag{A.2}$$

where $h_1 < h_2 < h_3$, and h_1 corresponds to the finest grid level. We can also consider the solution or quantity of interest ϕ_i computed for each grid level. The difference between each grid level is given by

$$\begin{aligned} \epsilon_{21} &= \phi_2 - \phi_1 \\ \epsilon_{32} &= \phi_3 - \phi_2 \end{aligned} \tag{A.3}$$

The apparent order p of convergence of the solutions can be evaluated by

$$p = \frac{1}{\ln\left(r_{21}\right)}\left(\ln|\epsilon_{31}/\epsilon_{32}| + q(p)\right) \tag{A.4}$$

$$q(p) = \ln\left(\frac{r_{21}^p - s}{r_{32}^p - s}\right) \tag{A.5}$$

$$s = 1 \cdot sgn(\epsilon_{32}/\epsilon_{21}) \tag{A.6}$$

Here, p can be computed using fixed-point iteration. The absolute value is not taken around Equation (A.4) to allow for determination of divergence if the order p is negative. The numerical solution at the infinite grid ($h = 0.0$), or continuum value, can be extrapolated using Richardson extrapolation. This is given by

$$\phi_{ext}^{21} = \frac{r_{21}^{p}\phi_1 - \phi_2}{r_{21}^{p} - 1} \tag{A.7}$$

Several error terms can then be computed. The approximate relative fine-grid error is given by

$$e_a^{21} = \left| \frac{\phi_1 - \phi_2}{\phi_1} \right|. \tag{A.8}$$

and the extrapolated relative fine-grid error is given by

$$e_{ext}^{21} = \left| \frac{\phi_{ext}^{21} - \phi_1}{\phi_{ext}^{21}} \right|, \tag{A.9}$$

The grid convergence index (GCI) at the fine grid level is given by

$$GCI_{\text{fine}}^{21} = \frac{1.25 e_a^{21}}{r_{21}^{p} - 1} \tag{A.10}$$

where 1.25 is a "safety factor". The GCI is a measure of the uncertainty, or error band, of the infinite grid solution from the true numerical solution. The GCI can also be used to measure whether a set of solutions within a grid family are in the asymptotic range of convergence. This can be evaluated by computing the ratio between GCI^{21} and GCI^{32}, which is given by

$$\alpha = \frac{r^{p} GCI^{21}}{GCI^{32}}. \tag{A.11}$$

If the ratio α is close to 1.0, this indicates that the solutions are within the asymptotic range of convergence.

Appendix B

Tricubic Hermite Interpolation

Tricubic interpolation allows for an approximate evaluation of quantities at any point lying within a three-dimensional grid. In structured meshes for CFD, it is frequently used for constructing a mapping of the grid to a "computational domain", or a separate uniform, rectangular grid [115]. This mapping then allows for evaluation of grid or field quantities at arbitrary points within the grid. At a given cell $\mathbf{I}_{j,k,l} = [\xi_j, \xi_{j+1}] \times [\eta_k, \eta_{k+1}] \times [\zeta_l, \zeta_{l+1}]$, the interpolant $f_{j,k,l}(\xi, \eta, \zeta)$ is given by

$$f_{j,k,l}(\xi,\eta,\zeta) = \left[\sum_{p=0}^{3} \alpha_p N_p(\xi)\right]\left[\sum_{q=0}^{3} \beta_q N_q(\eta)\right]\left[\sum_{r=0}^{3} \gamma_r N_r(\zeta)\right]. \tag{B.1}$$

Here, f can be a scalar or vector quantity, and ξ, η, ζ correspond to the real, continuous representation of the integer grid indices j, k, l. $N_i(x)$ are the shape functions of the one-dimensional Hermite spline. At a one-dimensional cell $\mathbf{I}_j$, the shape functions are given by

$$\begin{aligned}
N_0(x) &= 2\left((x-x_j)/h_j\right)^3 - 3\left((x-x_j)/h_j\right)^2 + 1 \\
N_1(x) &= -2\left((x-x_j)/h_j\right)^3 + 3\left((x-x_j)/h_j\right)^2 \\
N_2(x) &= \left((x-x_j)/h_j\right)^3 - 2\left((x-x_j)/h_j\right)^2 + \left((x-x_j)/h_j\right) \\
N_3(x) &= \left((x-x_j)/h_j\right)^3 - \left((x-x_j)/h_j\right)^2,
\end{aligned} \tag{B.2}$$

and $h_j = x_{j+1} - x_j$. The first two shape functions are associated directly with the interpolated quantities at $x_j = 0$ and x_{j+1}, respectively. The latter two shape functions are associated with the first derivatives at the nodes. The nodal values α_p, β_q, and γ_r are not evaluated individually, but instead they are evaluated as products since the products directly equate to the nodal vertices and derivatives of the grid itself. For each cell, there are eight quantities needed to generate the interpolant, and each cell contains eight nodes. In total, each cell has 64 coefficients that must be computed. In the cell $\mathbf{I}_{jkl}$, these are given

by the following,

$$\begin{aligned}
f\left(\xi_{j+p},\eta_{k+q},\zeta_{l+r}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=0,1, \quad q=0,1, \quad r=0,1\\
\frac{\partial f}{\partial \xi}\left(\xi_{j+p-2},\eta_{k+q},\zeta_{l+r}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=2,3, \quad q=0,1, \quad r=0,1\\
\frac{\partial f}{\partial \eta}\left(\xi_{j+p},\eta_{k+q-2},\zeta_{l+r}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=0,1, \quad q=2,3, \quad r=0,1\\
\frac{\partial f}{\partial \zeta}\left(\xi_{j+p},\eta_{k+q},\zeta_{l+r-2}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=0,1, \quad q=0,1, \quad r=2,3\\
\frac{\partial^2 f}{\partial \xi\partial \eta}\left(\xi_{j+p-2},\eta_{k+q-2},\zeta_{l+r}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=2,3, \quad q=2,3, \quad r=0,1\\
\frac{\partial^2 f}{\partial \xi\partial \zeta}\left(\xi_{j+p-2},\eta_{k+q},\zeta_{l+r-2}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=2,3, \quad q=0,1, \quad r=2,3\\
\frac{\partial^2 f}{\partial \eta\partial \zeta}\left(\xi_{j+p},\eta_{k+q-2},\zeta_{l+r-2}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=0,1, \quad q=2,3, \quad r=2,3\\
\frac{\partial^3 f}{\partial \xi\partial \eta\partial \zeta}\left(\xi_{j+p-2},\eta_{k+q-2},\zeta_{l+r-2}\right) &= \alpha_p\beta_q\gamma_r, \qquad p=2,3, \quad q=2,3, \quad r=2,3
\end{aligned} \tag{B.3}$$

The derivatives can be approximated using second-order centered finite differences. For example,

$$\frac{\partial f}{\partial \xi}\left(\xi_j,\eta_k,\zeta_l\right) \approx \frac{f_{j+1,k,l}-f_{j-1,k,l}}{2}. \tag{B.4}$$

Likewise, mixed derivatives can be computed by evaluating centered differences of other approximate derivatives. For example, a mixed derivative in ξ and η can be evaluated with

$$\frac{\partial^2 f}{\partial \xi\partial \eta}\left(\xi_j,\eta_k,\zeta_l\right) \approx \frac{\frac{\partial f}{\partial \xi}\left(\xi_j,\eta_{k+1},\zeta_l\right)-\frac{\partial f}{\partial \xi}\left(\xi_j,\eta_{k-1},\zeta_l\right)}{2} = \frac{f_{j+1,k+1,l}-f_{j-1,k+1,l}-f_{j+1,k-1,l}+f_{j-1,k-1,l}}{4}. \tag{B.5}$$

At the grid boundaries, one-sided differencing can be used. For example, at the first index $j=1$ and last index $j=nj$ in ξ, this is given by

$$\begin{aligned}
\frac{\partial f}{\partial \xi}\left(\xi_1,\eta_k,\zeta_l\right) &\approx \frac{3}{2}\left(f_{2,k,l}-f_{1,k,l}\right)-\frac{1}{2}\left(f_{3,k,l}-f_{2,k,l}\right)\\
\frac{\partial f}{\partial \xi}\left(\xi_{nj},\eta_k,\zeta_l\right) &\approx \frac{3}{2}\left(f_{nj,k,l}-f_{nj-1,k,l}\right)-\frac{1}{2}\left(f_{nj-2,k,l}-f_{nj-3,k,l}\right)
\end{aligned} \tag{B.6}$$

Evaluating these derivatives can be done just once to generate the interpolant across all cells.

Quantities at desired points can then be evaluated using the generated interpolants. The process starts by finding the cell that contains the desired point. This involves determining the parameterization of that point in ξ,η, and ζ. If the parameterization of the point is completely unknown, this may require a root-finding procedure using an iterative method like Newton's method or fixed-point iteration. The parameterization can then allow for the corresponding cell to be found, where the parameterization can then be inputted into the corresponding interpolant function to obtain the desired quantities.

References

[1] F. T. Johnson, E. N. Tinoco, and N. J. Yu, "Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle," *Computers & Fluids*, vol. 34, pp. 1115–1151, December 2005, doi: 10.1016/j.compfluid.2004.06.005.

[2] D. Reckzeh, "Aerodynamic design of the high-lift-wing for a Megaliner aircraft," *Aerospace Science and Technology*, vol. 7, pp. 107–119, March 2003, doi: 10.1016/S1270-9638(02)00002-0.

[3] M. R. Malik and D. M. Bushnell, "Role of Computational Fluid Dynamics and Wind Tunnels in Aeronautics R&D," NASA TP-2012-217602, September 2012.

[4] K. S. Abdol-Hamid, F. Ghaffari, and E. B. Parlette, "Ares I Vehicle Computed Turbulent Ascent Aerodynamic Data Development and Analysis," *Journal of Spacecraft and Rockets*, vol. 49, pp. 596–608, July 2012, doi: 10.2514/1.A32112.

[5] M. Wright, K. Edquist, C. Tang, B. Hollis, P. Krasa, and C. Campbell, "A Review of Aerothermal Modeling for Mars Entry Missions," AIAA Paper 2010-443, January 2010, doi: 10.2514/6.2010-443.

[6] M. Gusman, J. Housman, and C. Kiris, "Best Practices for CFD Simulations of Launch Vehicle Ascent with Plumes - OVERFLOW Perspective," AIAA Paper 2011-1054, January 2011, doi: 10.2514/6.2011-1054.

[7] J. G. Meeroff, D. J. Dalle, S. E. Rogers, A. C. Burkhead, D. G. Schauerhamer, and J. F. Diaz, "Advances in Space Launch System Booster Separation CFD," AIAA Paper 2023-0238, January 2023, doi: 10.2514/6.2023-0238.

[8] M. W. Lee, D. J. Dalle, M. M. Sander, and C. J. Addona, "Development of Aerodynamic Loads Databases for the Space Launch System Booster Separation Event," AIAA Paper 2024-0465, January 2024, doi: 10.2514/6.2024-0465.

[9] J. L. Steger, F. C. Dougherty, and J. A. Benek, "A Chimera grid scheme," in *Advances in Grid Generation*, K. Ghia and U. Ghia, Eds., vol. 5. New York, New York: American Society of Mechanical Engineers, Fluids Engineering Division, 1983, pp. 59–69.

[10] J. L. Steger and J. A. Benek, "On the use of composite grid schemes in computational aerodynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 64, pp. 301–320, October 1987, doi: 10.1016/0045-7825(87)90045-4.

[11] R. Gomez, D. Vicker, S. Rogers, M. Aftosmis, W. Chan, R. Meakin, and S. Murman, "STS-107 Investigation Ascent CFD Support," AIAA Paper 2004-2226, June 2004, doi: 10.2514/6.2004-2226.

[12] J. U. Ahmad, S. A. Pandya, W. M. Chan, and N. M. Chaderjian, "Navier-Stokes Simulation of Air-Conditioning Facility of a Large Modern Computer Room," in *2005 ASME Fluids Engineering Division Summer Meeting and Exhibition*, vol. 2. Houston, Texas: ASMEDC, January 2005, pp. 651–656, doi: 10.1115/FEDSM2005-77225.

[13] S. Pandya, J. Onufer, W. Chan, and G. Klopfer, "Capsule Abort Recontact Simulation," AIAA Paper 2006-3324, June 2006, doi: 10.2514/6.2006-3324.

[14] C. C. Kiris, D. Kwak, W. Chan, and J. A. Housman, "High-fidelity simulations of unsteady flow through turbopumps and flowliners," *Computers & Fluids*, vol. 37, pp. 536–546, June 2008, doi: 10.1016/j.compfluid.2007.07.010.

[15] C. Kiris, J. Housman, M. Gusman, W. Chan, and D. Kwak, "Time-Accurate Computational Analysis of Ignition Overpressure in the Flame Trench," in *Computational Fluid Dynamics Review 2010*. World Scientific, July 2010, pp. 377–397, doi: 10.1142/9789814313377_0015.

[16] S. E. Rogers, D. J. Dalle, and W. M. Chan, "CFD Simulations of the Space Launch System Ascent Aerodynamics and Booster Separation," AIAA Paper 2015-0778, January 2015, doi: 10.2514/6.2015-0778.

[17] J. A. Housman, E. Sozer, and C. C. Kiris, "LAVA Simulations for the First AIAA Sonic Boom Prediction Workshop," AIAA Paper 2014-2008, June 2014, doi: 10.2514/6.2014-2008.

[18] S. A. Pandya, A. Uranga, A. Espitia, and A. Huang, "Computational Assessment of the Boundary Layer Ingesting Nacelle Design of the D8 Aircraft," AIAA Paper 2014-0907, January 2014, doi: 10.2514/6.2014-0907.

[19] J. A. Housman and C. C. Kiris, "Numerical Simulations of Shock/Plume Interaction Using Structured Overset Grids," AIAA Paper 2015-2262, June 2015, doi: 10.2514/6.2015-2262.

[20] ——, "Structured Overlapping Grid Simulations of Contra-Rotating Open Rotor Noise," AIAA Paper 2016-0814, January 2016, doi: 10.2514/6.2016-0814.

[21] J. M. Derlaga, C. W. Jackson, and P. G. Buning, "Recent Progress in OVERFLOW Convergence Improvements," AIAA Paper 2020-1045, January 2020, doi: 10.2514/6.2020-1045.

[22] J. Slotnick, A. Khodadoust, J. Alonso, and D. Darmofal, "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," NASA CR-2014-218178, March 2014.

[23] J. R. Chawner, J. F. Dannenhoffer, and N. J. Taylor, "Geometry, mesh generation, and the CFD 2030 vision," AIAA Paper 2016-3485, June 2016, doi: 10.2514/6.2016-3485.

[24] J. F. Thompson, J. L. Steger, and H. Yoshihara, "Three-Dimensional Grid Generation for Complex Configurations: Recent Progress," AGARD AG-309, Neuilly-sur-Seine, France, January 1988.

[25] J. F. Thompson, "A Reflection on Grid Generation in the 90s: Trends, Needs, and Influences," in *Numerical Grid Generation in Computational Field Simulations*, B. Soni, J. F. Thompson, J. Haeuser, and P. R. Eiseman, Eds., Starkville,Mississippi, 1996, pp. 1–45.

[26] W. M. Chan, "Best practices on overset structured mesh generation for the high-lift CRM geometry," AIAA Paper 2017-0362, January 2017, doi: 10.2514/6.2017-0362.

[27] W. M. Chan and J. L. Steger, "Enhancements of a three-dimensional hyperbolic grid generation scheme," *Applied Mathematics and Computation*, vol. 51, pp. 181–205, October 1992, doi: 10.1016/0096-3003(92)90073-A.

[28] R. Meakin, "Object X-rays for cutting holes in composite overset structured grids," AIAA Paper 2001-2537, June 2001, doi: 10.2514/6.2001-2537.

[29] W. M. Chan, S. A. Pandya, and R. Haimes, "Automation of Overset Structured Surface Mesh Generation on Complex Geometries," AIAA Paper 2019-3671, June 2019, doi: 10.2514/6.2019-3671.

[30] W. J. Gordon and C. A. Hall, "Construction of curvilinear co-ordinate systems and applications to mesh generation," *International Journal for Numerical Methods in Engineering*, vol. 7, pp. 461–477, January 1973, doi: 10.1002/nme.1620070405.

[31] W. J. Gordon and L. C. Thiel, "Transfinite mappings and their application to grid generation," *Applied Mathematics and Computation*, vol. 10-11, pp. 171–233, January 1982, doi: 10.1016/0096-3003(82)90191-6.

[32] L.-E. Eriksson, "Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation," *AIAA Journal*, vol. 20, pp. 1313–1320, October 1982, https://arc.aiaa.org/doi/10.2514/3.798010.2514/3.7980.

[33] ——, "Three-Dimensional Spline-Generated Coordinate Transformations for Grids Around Wing-Body Configurations," in *NASA Langley Research Center Numerical Grid Generation Techniques*, NASA Langley Research Center, Virginia, 1980, pp. 253–264.

[34] J. F. Thompson, F. C. Thames, and C. Mastin, "Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies," *Journal of Computational Physics*, vol. 15, pp. 299–319, July 1974, doi: 10.1016/0021-9991(74)90114-4.

[35] P. D. Thomas and J. F. Middlecoff, "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations," *AIAA Journal*, vol. 18, pp. 652–656, June 1980, doi: 10.2514/3.50801.

[36] J. F. Thompson, "A general three-dimensional elliptic grid generation system on a composite block structure," *Computer Methods in Applied Mechanics and Engineering*, vol. 64, pp. 377–411, October 1987, doi: 10.1016/0045-7825(87)90047-8.

[37] B. K. Soni, "Elliptic grid generation system: control functions revisited—I," *Applied Mathematics and Computation*, vol. 59, pp. 151–163, December 1993, doi: 10.1016/0096-3003(93)90086-T.

[38] A. Khamayseh and C. Mastin, "Surface grid generation based on elliptic PDE models," *Applied Mathematics and Computation*, vol. 65, pp. 253–264, September 1994, doi: 10.1016/0096-3003(94)90181-3.

[39] S. Spekreijse, "Elliptic Grid Generation Based on Laplace Equations and Algebraic Transformations," *Journal of Computational Physics*, vol. 118, pp. 38–61, April 1995, doi: 10.1006/jcph.1995.1078.

[40] A. Khamayseh and B. Hamann, "Elliptic grid generation using NURBS surfaces," *Computer Aided Geometric Design*, vol. 13, pp. 369–386, June 1996, doi: 10.1016/0167-8396(95)00050-X.

[41] P. Rubbert and K. Lee, "Patched coordinate systems," *Applied Mathematics and Computation*, vol. 10-11, pp. 235–252, January 1982, doi: 10.1016/0096-3003(82)90192-8.

[42] N. P. Weatherill and C. R. Forsey, "Grid Generation and Flow Calculations for Complex Aircraft Geometries Using a Multi-Block Scheme." AIAA Paper 1984-1665, June 1984, doi: 10.2514/6.1984-1665.

[43] K. Miki and T. Takagi, "A domain decomposition and overlapping method for the generation of three-dimensional boundary-fitted coordinate systems," *Journal of Computational Physics*, vol. 53, pp. 319–330, February 1984, doi: 10.1016/0021-9991(84)90044-5.

[44] J. F. Thompson, "Composite grid generation code for general 3-D regions - The EAGLE code," *AIAA Journal*, vol. 26, pp. 271–272, 1988, doi: 10.2514/3.9884.

[45] A. Arabshahi and D. Whitfield, "A multiblock approach to solving the three-dimensional unsteady Euler equations about a wing-pylon-store configuration," AIAA Paper 1989-3401, August 1989, doi: 10.2514/6.1989-3401.

[46] J. F. Dannenhoffer, III, "A block-structuring technique for general geometries," AIAA Paper 1991-145, January 1991, doi: 10.2514/6.1991-145.

[47] R. Sorenson and K. McCann, "A method for interactive specification of multiple-block topologies," AIAA Paper 1991-147, January 1991, doi: 10.2514/6.1991-147.

[48] J. R. Chawner and J. P. Steinbrenner, "Automatic structured grid generation using GRIDGEN," in *Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamic (CFD) Solutions*, Y. K. Choo, Ed. NASA-CP, 1995, pp. 463–476.

[49] G. Starius, "Constructing orthogonal curvilinear meshes by solving initial value problems," *Numerische Mathematik*, vol. 28, pp. 25–48, 1977, doi: 10.1007/BF01403855.

[50] J. L. Steger and D. S. Chaussee, "Generation of Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations," *SIAM Journal on Scientific and Statistical Computing*, vol. 1, pp. 431–437, December 1980, doi: 10.1137/0901031.

[51] D. W. Kinsey and T. J. Barth, "Description of a hyperbolic grid generating procedure for arbitrary two-dimensional bodies," AFWAL-TM 84-191-FIMM, 1984.

[52] J. Cordova and T. Barth, "Grid generation for general 2-D regions using hyperbolic equations," AIAA Paper 1988-520, January 1988, doi: 10.2514/6.1988-520.

[53] J. L. Steger and Y. M. Rizk, "Generation of Three-Dimensional Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations," NASA TM-86753, June 1985.

[54] J. L. Steger, "Generation of Three-Dimensional Body-Fitted Grids by Solving Hyperbolic Partial Differential Equations," NASA TM-101069, January 1989.

[55] ——, "Notes on surface grid generation using hyperbolic partial differential equations," Internal Report TM CFD/UCD 89-101, Department of Mechanical, Aeronautical and Materials Engineering, Univ. of Calif., Davis, 1989.

[56] ——, "Grid generation with hyperbolic partial differential equations for application to complex configurations," in *Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, A. Arcilla, J. Hauser, P. R. Eiseman, and J. F. Thompson, Eds., North Holland, New York, 1991, pp. 871–886.

[57] W. M. Chan and P. G. Buning, "Surface grid generation methods for overset grids," *Computers & Fluids*, vol. 24, pp. 509–522, June 1995, doi: 10.1016/0045-7930(95)00003-U.

[58] E. A. Volkov, "The development of a grid method for the solution of Laplace's equation in finite or infinite regions with piecewise-smooth boundaries," *Mathematical Notes of the Academy of Sciences of the USSR*, vol. 2, pp. 747–755, October 1967, doi: 10.1007/BF01093653.

[59] E. Volkov, "The method of composite meshes for finite and infinite regions with piecewise smooth boundary," in *Proceedings of the Steklov Institute of Mathematics*, I. Petrovskii and S. Nikol'skii, Eds., 1968, pp. 145–185.

[60] G. Starius, "Composite mesh difference methods for elliptic boundary value problems," *Numerische Mathematik*, vol. 28, pp. 243–258, June 1977, doi: 10.1007/BF01394455.

[61] ——, "On composite mesh difference methods for hyperbolic differential equations," *Numerische Mathematik*, vol. 35, pp. 241–255, September 1980, doi: 10.1007/BF01396411.

[62] B. Kreiss, "Construction of a Curvilinear Grid," *SIAM Journal on Scientific and Statistical Computing*, vol. 4, pp. 270–279, June 1983, doi: 10.1137/0904021.

[63] F. C. Dougherty, J. A. Benek, and J. L. Steger, "On Applications of Chimera Grid Schemes to Store Separation," NASA TM-88193, October 1985.

[64] S. J. Parks, P. G. Buning, J. L. Steger, and W. M. Chan, "Collar grids for intersecting geometric components within the chimera overlapped grid scheme," AIAA Paper 1991-1587, June 1991, doi: 10.2514/6.1991-1587.

[65] R. Meakin, "Automatic off-body grid generation for domains of arbitrary size," June 2001, doi: 10.2514/6.2001-2536.

[66] W. M. Chan, "Advances in Software Tools for Pre-processing and Post-processing of Overset Grid Computations," in *9th International Conference on Numerical Grid Generation in Computational Field Simulations*, San Jose, California, 2005, pp. 1–8.

[67] S. E. Rogers, N. E. Suhs, and W. E. Dietz, "PEGASUS 5: An Preprocessor for Overset-Grid Computational Fluid Dynamics," *AIAA Journal*, vol. 41, pp. 1037–1045, June 2003, doi: 10.2514/2.2070.

[68] D. Jespersen, T. Pulliam, and P. Buning, "Recent enhancements to OVERFLOW," AIAA Paper 1997-644, January 1997, doi: 10.2514/6.1997-644.

[69] W. M. Chan, "Overset grid technology development at NASA Ames Research Center," *Computers and Fluids*, vol. 38, pp. 496–503, 2009, doi: 10.1016/j.compfluid.2008.06.009.

[70] M.-S. Liou and Y. Zheng, "A novel approach of three-dimensional hybrid grid methodology: Part 2. flow solution," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 37, pp. 4173–4193, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782503003864

[71] W. Henshaw, "Overture: An Object-Oriented Framework for Overlapping Grid Applications," AIAA Paper 2002-3189, June 2002, doi: 10.2514/6.2002-3189.

[72] G. Chesshire and W. D. Henshaw, "Composite overlapping meshes for the solution of partial differential equations," *Journal of Computational Physics*, vol. 90, pp. 1–64, 1990, doi: 10.1016/0021-9991(90)90196-8.

[73] K. Nakahashi, F. Togashi, and D. Sharov, "Intergrid-Boundary Definition Method for Overset Unstructured Grid Approach," *AIAA Journal*, vol. 38, pp. 2077–2084, November 2000, doi: 10.2514/2.869.

[74] W. Chan and R. Gomez, III, "Advances in automatic overset grid generation around surface discontinuities," AIAA Paper 1999-3303, November 1999, doi: 10.2514/6.1999-3303.

[75] J. F. Dannenhoffer and R. L. Davis, "Automated creation of overset grids directly from solid-model feature trees," AIAA Paper 2009-3991, 2009, doi: 10.2514/6.2009-3991.

[76] J. Dannenhoffer and R. Haimes, "Automated Creation of 3-D Overset Grids Directly from Solid Models," AIAA Paper 2011-3540, June 2011, doi: 10.2514/6.2011-3540.

[77] S. Pandya, W. Chan, and J. Kless, "Automation of Structured Overset Mesh Generation for Rocket Geometries," AIAA Paper 2009-3993, June 2009, doi: 10.2514/6.2009-3993.

[78] W. M. Chan, "Strategies Toward Automation of Overset Structured Surface Grid Generation," AIAA Paper 2017-3451, June 2017, doi: 10.2514/6.2017-3451.

[79] M. Gammon, H. Bucklow, and R. Fairey, "A review of common geometry issues affecting mesh generation," AIAA Paper 2018-1402, January 2018, doi: 10.2514/6.2018-1402.

[80] S. A. Pandya, W. M. Chan, and R. Haimes, "An automated marching scheme for overset structured surface mesh generation," in *10th International Conference on Computational Fluid Dynamics (ICCFD10)*. Barcelona, Spain: ICCFD, 2018, pp. 1–14.

[81] R. Haimes and J. Dannenhoffer, "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry," AIAA Paper 2013-3073, June 2013, doi: 10.2514/6.2013-3073.

[82] W. M. Chan, S. A. Pandya, and A. M. Chuen, "Automation of Overset Structured Mesh Generation on Boundary Representation Geometries," AIAA Paper 2022-3607, June 2022, doi: 10.2514/6.2022-3607.

[83] W. M. Chan, A. M. Chuen, and J. C. Jensen, "Advances in Automation of Overset Structured Volume Mesh Generation and Domain Connectivity," AIAA Paper 2024-4305, July 2024, doi: 10.2514/6.2024-4305.

[84] R. M. Beam and R. Warming, "An implicit finite-difference algorithm for hyperbolic systems in conservation-law form," *Journal of Computational Physics*, vol. 22, pp. 87–110, September 1976, doi: 10.1016/0021-9991(76)90110-8.

[85] W. Chan, R. Gomez, S. Rogers, and P. Buning, "Best Practices in Overset Grid Generation," AIAA Paper 2002-3191, June 2002, doi: 10.2514/6.2002-3191.

[86] B. C. Systems, "ANSA: The advanced CAE pre-processing software for complete model build-up," https://www.beta-cae.com/, 2025, accessed: 2025-02-19.

[87] H. C. Lee and T. H. Pulliam, "Overflow Juncture Flow Computations Compared with Experimental Data," AIAA Paper 2019-0080, January 2019, doi: 10.2514/6.2019-0080.

[88] I.-T. Chiu and R. Meakin, "On automating domain connectivity for overset grids," AIAA Paper 1995-854, January 1995, doi: 10.2514/6.1995-854.

[89] C. Hirsch, *Numerical computation of internal and external flows. Volume 2: Computational methods for inviscid and viscous flows*. Chichester, England: John Wiley & Sons, Ltd, 1990, vol. 2.

[90] P. R. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," *La Recherche Aerospatiale*, pp. 5–21, 1994.

[91] P. Spalart, "Trends in turbulence treatments," AIAA Paper 2000-2306, June 2000, doi: 10.2514/6.2000-2306.

[92] M. L. Shur, M. K. Strelets, A. K. Travin, and P. R. Spalart, "Turbulence Modeling in Rotating and Curved Channels: Assessing the Spalart-Shur Correction," *AIAA Journal*, vol. 38, pp. 784–792, May 2000, doi: 10.2514/2.1058.

[93] P. Spalart, "Strategies for turbulence modelling and simulations," *International Journal of Heat and Fluid Flow*, vol. 21, pp. 252–263, June 2000, doi: 10.1016/S0142-727X(00)00007-2.

[94] M. Mani, D. Babcock, C. Winkler, and P. Spalart, "Predictions of a Supersonic Turbulent Flow in a Square Duct," AIAA Paper 2013-860, January 2013, doi: 10.2514/6.2013-860.

[95] C. L. Rumsey, J. R. Carlson, and N. N. Ahmad, "FUN3D Juncture Flow Computations Compared with Experimental Data," AIAA Paper 2019-0079, January 2019, doi: 10.2514/6.2019-0079.

[96] C. L. Rumsey, H. C. Lee, and T. H. Pulliam, "Reynolds-Averaged Navier-Stokes Computations of the NASA Juncture Flow Model Using FUN3D and OVERFLOW," AIAA Paper 2020, January 2020, doi: 10.2514/6.2020-1304.

[97] A. Jameson, W. Schmidt, and E. Turkel, "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes," AIAA Paper 1981-1259, June 1981, doi: 10.2514/6.1981-1259.

[98] P. L. Roe, "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981, doi: 10.1016/0021-9991(81)90128-5.

[99] B. Koren, "Upwind schemes, multigrid and defect correction for the steady Navier-Stokes equations," in *11th International Conference on Numerical Methods in Fluid Dynamics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 344–348, doi: 10.1007/3-540-51048-6_52.

[100] T. Pulliam and D. Chaussee, "A diagonal form of an implicit approximate-factorization algorithm," *Journal of Computational Physics*, vol. 39, pp. 347–363, February 1981, doi: 10.1016/0021-9991(81)90156-X.

[101] R. Nichols, R. Tramel, and P. Buning, "Solver and Turbulence Model Upgrades to OVERFLOW 2 for Unsteady and High-Speed Applications," June 2006, doi: 10.2514/6.2006-2824.

[102] Y.-H. Choi and C. Merkle, "The Application of Preconditioning in Viscous Flows," *Journal of Computational Physics*, vol. 105, no. 2, pp. 207–223, 4 1993, doi: 10.1006/jcph.1993.1069.

[103] J. Weiss and W. Smith, "Preconditioning applied to variable and constant density time-accurate flows on unstructured meshes," AIAA Paper 1994-2209, 6 1994, doi: https://arc.aiaa.org/doi/10.2514/6.1994-220910.2514/6.1994-2209.

[104] A. M. Chuen and W. M. Chan, "Overlap Preservation Using Loosely-Coupled Boundary Conditions for Body-Fitted Structured Overset Grids," AIAA Paper 2022-0216, January 2022, doi: 10.2514/6.2022-0216.

[105] W. Johnson, "A quiet helicopter for air taxi operations," in *Aeromechanics for Advanced Vertical Flight Technical Meeting 2020*, 2020, pp. 36–47.

[106] C. Silva, W. R. Johnson, E. Solis, M. D. Patterson, and K. R. Antcliff, "VTOL Urban Air Mobility Concept Vehicles for Technology Development," AIAA Paper 2018-3847, June 2018, doi: 10.2514/6.2018-3847.

[107] M. A. Kegerise and D. H. Neuhart, "Wind Tunnel Test of a Risk-Reduction Wing/Fuselage Model to Examine Juncture-Flow Phenomena," NASA TM-2016-219348, November 2016.

[108] A. M. Chuen, S. S. Hosseini, J. C. Jensen, and W. M. Chan, "Aerodynamic Simulations for Complex Geometries Using Automatically Generated Structured Overset Meshes," AIAA Paper 2023-3602, June 2023, doi: 10.2514/6.2023-3602.

[109] S. S. Hosseini, A. M. Chuen, and W. M. Chan, "Computational Aerodynamics Study of the Lift+Cruise VTOL Concept Vehicle Components," in *6th Decennial Aeromechanics Specialists' Conference*, Santa Clara, CA, February 2024.

[110] NASA Langley Research Center, "Turbulence Modeling Resource," https://turbmodels.larc.nasa.gov, 2024, [Online]. Accessed: 2025-01-04.

[111] B. Diskin, W. K. Anderson, M. J. Pandya, C. L. Rumsey, J. Thomas, Y. Liu, and H. Nishikawa, "Grid Convergence for Three Dimensional Benchmark Turbulent Flows," AIAA Paper 2018-1102, January 2018, doi: 10.2514/6.2018-1102.

[112] P. L. Roe and J. Pike, "Efficient construction and utilisation of approximate Riemann solutions," in *Computing methods in applied sciences and engineering, VI*, 1984, pp. 499–518.

[113] B. Koren, "Upwind Schemes for the Navier-Stokes Equations," in *Nonlinear Hyperbolic Equations — Theory, Computation Methods, and Applications*. Wiesbaden: Vieweg+Teubner Verlag, 1989, pp. 300–309, doi: 10.1007/978-3-322-87869-4_31.

[114] I. B. Celik, U. Ghia, P. J. Roache, C. J. Freitas, H. Coleman, and P. E. Raad, "Procedure for Estimation and Reporting of Uncertainty Due to Discretization in CFD Applications," *Journal of Fluids Engineering*, vol. 130, p. 078001, 2008, doi: 10.1115/1.2960953.

[115] P. G. Buning and T. H. Pulliam, "Near-body grid adaption for overset grids," AIAA Paper 2016-3326, June 2016, doi: 10.2514/6.2016-3326.

[116] T. Hsieh, "An investigation of separated flow about a hemisphere-cylinder at incidence in the Mach number range from 0.6 to 1.5," AIAA Paper 1977-179, January 1977, doi: 10.2514/6.1977-179.

[117] V. Schmitt and Charpin F., "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," Fluid Dynamics Panel Working Group 04, AGARD AR 138, 1979.

[118] J. Mayeur, A. Dumont, D. Destarac, and V. Gleize, "RANS simulations on TMR 3D test cases with the Onera *elsA* flow solver," AIAA Paper 2016-1357, January 2016, doi: 10.2514/6.2016-1357.

[119] C. L. Rumsey, D. Neuhart, and M. A. Kegerise, "The NASA Juncture Flow Experiment: Goals, Progress, and Preliminary Testing (Invited)," AIAA Paper 2016-1557, January 2016, doi: 10.2514/6.2016-1557.

[120] H. C. Lee, T. H. Pulliam, D. Neuhart, and M. A. Kegerise, "CFD Analysis in Advance of the NASA Juncture Flow Experiment," AIAA Paper 2017-4127, June 2017, doi: 10.2514/6.2017-4127.

[121] C. L. Rumsey, "The NASA Juncture Flow Test as a Model for Effective CFD/Experimental Collaboration," in *2018 Applied Aerodynamics Conference*. Atlanta, Georgia: AIAA, June 2018.

[122] C. L. Rumsey, J.-R. Carlson, T. H. Pulliam, and P. R. Spalart, "Improvements to the Quadratic Constitutive Relation Based on NASA Juncture Flow Data," *AIAA Journal*, vol. 58, pp. 4374–4384, October 2020, doi: 10.2514/1.J059683.

[123] C. L. Rumsey, "Insights and Lessons Learned from the NASA Juncture Flow Experiment," *Journal of Aircraft*, vol. 59, pp. 1493–1499, November 2022, doi: 10.2514/1.C036838.

[124] C. L. Rumsey, N. N. Ahmad, J.-R. Carlson, M. A. Kegerise, D. H. Neuhart, J. A. Hannon, L. N. Jenkins, C.-S. Yao, P. Balakumar, S. Gildersleeve, S. M. Bartram, T. H. Pulliam, M. E. Olsen, and P. R. Spalart, "NASA Juncture Flow Computational Fluid Dynamics Validation Experiment," *AIAA Journal*, vol. 60, pp. 4789–4806, August 2022, doi: 10.2514/1.J061600.

[125] S. J. Bianco, C. T. Chevalier, J. S. Litt, J. K. Smith, J. W. Chapman, and J. L. Kratz, "Revolutionary vertical lift technology (RVLT) side-by-side hybrid concept vehicle powertrain dynamic model," in *Proceedings of the ASME Turbo Expo*, vol. 4. American Society of Mechanical Engineers, June 2021, doi: 10.1115/GT2021-59375.

[126] P. Ventura Diaz, W. Johnson, J. Ahmad, and S. Yoon, "The Side-by-Side Urban Air Taxi Concept," AIAA Paper 2019-2828, June 2019, doi: 10.2514/6.2019-2828.

[127] J. C. Vassberg, M. A. Dehaan, S. M. Rivers, and R. A. Wahls, "Development of a Common Research Model for Applied CFD Validation Studies," AIAA Paper 2008-6919, August 2008, doi: 10.2514/6.2008-6919.

[128] E. Tinoco, O. P. Brodersen, S. Keye, K. R. Laflin, J. C. Vassberg, B. Rider, R. A. Wahls, J. H. Morrison, B. W. Pomeroy, D. Hue, and M. Murayama, "Summary Data from the Seventh AIAA CFD Drag Prediction Workshop," AIAA Paper 2023-3492, June 2023, doi: 10.2514/6.2023-3492.

[129] NASA Glenn Research Center, "Examining Spatial (Grid) Convergence," https://www.grc.nasa.gov/www/wind/valid/tutorial/spatconv.html, 2024, [Online]. Accessed: 2025-01-04.