# ABSTRACT

Title of dissertation: HAMILTONIAN PATHS AND STRANDS FOR UNIFIED GRID APPROACH FOR COMPUTING AERODYNAMIC FLOWS

Yong Su Jung
Doctor of Philosophy, 2019

Dissertation directed by: Professor James D. Baeder
Department of Aerospace Engineering

A solution algorithm using Hamiltonian paths and strand grids is presented for compressible Reynolds-Averaged Navier–Stokes (RANS) formulation as a unified grid approach. The hidden line-structure is robustly identified on the general unstructured grid with mixed elements which provides a framework for line-solvers similar to that with a structured grid solver. A pure quadrilateral/hexahedral mesh is a prerequisite for the line identification and enables approximate factorization along the lines on the unstructured grid. Among various methods, subdivision is the easiest way to obtain a pure quadrilateral/hexahedral mesh from the general unstructured grid.

Strand based grids have been employed to extend to three-dimension by extruding the surface mesh. As a result, Hamiltonian paths on the surface mesh represents two distinct surface coordinate directions and the strand grids represent the wall normal direction. These structures are analogous to the grid coordinate directions in a structured grid solver, therefore the current method is directly ap-

plicable on the typical structured grid.

The numerical accuracy and convergence rate are investigated under various flow conditions. The numerical efficiency was improved with a line-implicit method compared to a point-implicit method on the unstructured grid. Both stencil- and gradient-based reconstructions are available on the unstructured grid and the numerical accuracy for each method was evaluated on both structured and unstructured grids. The combined reconstruction method was also proposed for the current mesh system which uses both stencil- and gradient-based reconstructions simultaneously but for different grid directions. The solution convergence rate has been improved further using Generalized Minimum Residual (GMRES) method. GMRES requires a preconditioning step which is performed using the line-implicit method. GMRES provides better convergence rate than the pure line-implicit method at various flow conditions.

Both parts of mesh generation and flow solver are parallelized to be executed in parallel using METIS and MPI. During the mesh generation, two different domain partitioning methods are suggested for the strand grid and the unstructured volume mesh, respectively. The capability of the flow solver has been extended for rotary wing simulations: time-accurate method, turbulence model, moving grids, and overset meshes. The flow solver is also integrated into a multi-mesh/multi-solver paradigm through a Python framework which enables a more efficient solution algorithm than a single-solver. The integrated framework has been applied to various practical problems, such as wind turbine, rotor hub, and elastic rotor blades.

# HAMILTONIAN PATHS AND STRANDS FOR UNIFIED GRID APPROACH FOR COMPUTING AERODYNAMIC FLOWS

by

Yong Su Jung

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Dr. James D. Baeder, Chair/Advisor
Dr. Inderjit Chopra
Dr. Christopher Cadou
Dr. Anubhav Datta
Dr. Johan Larsson, Dean's Representative

# Dedication

This dissertation is dedicated to my mother, Soon-Ok Lee, and my father, Woon-Do Jung for their love and support throughout my life.

# Acknowledgments

I owe my gratitude to all the people who have made this dissertation possible. First and foremost, I would like to express my sincere gratitude to my advisor Dr. James Baeder for his continuous support during my Ph.D. study. It is a pleasure and honor for me being his student and working on such an interesting study under his guidance. His immense knowledge and patience have helped me to proceed from obstacles throughout the research. From every discussion with him, I have learned about the attitudes towards the obstacles as a research scientist.

I also would like to thank my dissertation and qualifying exam committee: Dr. Inderjit Chopra, Dr. Christopher Cadou, Dr. Anubhav Datta, and Dr. Johan Larsson for their time to serve on my committee and guidance with insightful feedback on my research.

I would like to acknowledge the financial support provided for this research work by Air Vehicle element of the HPCMP CREATE program and Army Research Laboratory (ARL). I wish to thank Dr. Jayanarayanan Sitaraman for his guidance that helped me to get started with this study.

I am grateful to all my friends and colleagues whom I have met at the University of Maryland over the years. First, Bharath has helped me in all aspects since I first came. I was able to finish this work with his helps. I also would like to thank many friends: Dylan, Seung-Joon, Nishan, Ananth, Masahiko, Daigo, Camli, Bumseok, Vera, Alex, Jannik, Xing, Luke and Minwook. They have affected my everyday life as a graduate student in different ways and I have learned a lot from

them. I would like to specially thank Jiseon for all the good memories she has given me out of the campus. I cannot forget her helps over the years.

I also wish to thank my aunt and her family. They have given a lot of helps to settle down here since I left from home. They always give a warm welcome to me.

Finally, none of this could have happened without my family - my mother, father, and brother who have supported me throughout my entire life. My mother always understands and encourages me better than anyone. My father always wishes for my career. No words can express the gratitude I owe them.

# Table of Contents

# List of Tables

# List of Figures

# Nomenclature

| | |
|---|---|
| $a$ | Speed of sound |
| $c$ | Chord length of the airfoil |
| $C_d$ | Drag coefficient |
| $C_f$ | Skin friction coefficient |
| $C_l$ | Lift coefficient |
| $C_m$ | Moment coefficient |
| $C_n$ | Sectional normal force coeffcient |
| $C_p$ | Pressure coefficient |
| $C_T$ | Thrust coefficient |
| $M$ | Mach number |
| $p$ | Pressure |
| $R$ | Radius of the rotor |
| $Re$ | Reynolds number |
| $Re_\theta$ | Momentum thickness Reynolds number |
| $\overline{Re_{\theta t}}$ | $Re_\theta$ at transition onset |
| $\overline{Re_\theta}$ | Transported $Re_{\theta t}$ |
| $u, v, w$ | Velocity components in the Cartesian directions |
| $x, y, z$ | Cartesian coordinates |
| $y^+$ | Nondimensional wall distance, $(y/\nu)\left(\sqrt{\tau/\rho}\right)$ |
| $\alpha$ | Angle of attack |
| $\gamma$ | Ratio of specific heats / Intermittency |
| $\mu$ | Molecular viscosity |
| $\mu_t$ | Eddy viscosity |
| $\nu$ | Kinematic viscosity |
| $\xi, \eta, \zeta$ | Computational coordinates |
| $\rho$ | Density |
| $\psi$ | Blade azimuthal angle |

**Subscripts**

| | |
|---|---|
| $i, j, k$ | Spatial cell indices |
| $\infty$ | Freestream flow variables |

**Abbreviations**

| | |
|---|---|
| 2D | Two dimensional |
| 3D | Three dimensional |
| ADI | Alternating Direction Implicit |
| BDF1 | Backward Difference formulation, first-order |
| BDF2 | Backward Difference formulation, second-order |
| CFL | Courant-Friedrichs-Lewy |

| | |
|---|---|
| DADI | Diagonalized Alternating Direction Implicit |
| DDADI | Diagonally Dominant ADI |
| DDLGS | Diagonally Dominant LGS |
| DDES | Delayed Detached Eddy Simulation |
| FM | Figure of Merit |
| GMRES | Generalized Minimum Residual Method |
| GPU | Graphic processing unit |
| LGS | Line Gauss-Seidel |
| LHS | Left hand side |
| LLS | Linear Least-Squares |
| LU-SGS | Lower-upper Symmetric Gauss-Seidel |
| MUSCL | Monotone Upstream-Centered Scheme for Conservation Laws |
| PGS | Point Gauss-Seidel |
| RANS | Reynolds Averaged Navier–Stokes |
| RHS | Right hand side |
| RPM | Revolutions per minute |
| SA | Spalart-Allmaras |
| WENO | Weighted Essentially Non-Oscillatory scheme |

# Chapter 1: Introduction

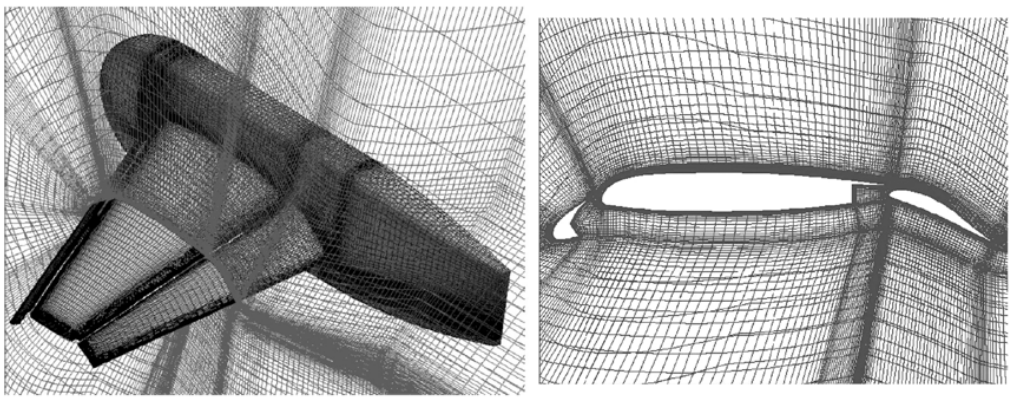
Aerospace applications are largely driven by the advances and development in the field of aerodynamics and have become reliant on advances in computational technology and simulation tools. Accurate modeling of the flow field around vehicle components is required to predict not only airloads from each component such as a wing or rotor blade, but also the effect of interactional aerodynamics between the components. Traditional low-order numerical techniques such as free-wake method or panel methods make modeling assumptions and limit their solutions from capturing detailed aerodynamic effects. With rapid advances in computational power and resources, high-fidelity computational fluid dynamics (CFD) are considered to be a principle technology in the analysis and design of air vehicles. Both solution algorithms for Euler or Navier-Stokes equations and meshing techniques have been matured to improve its robustness, efficiency, and accuracy across various speed ranges from subsonic to supersonic.

In most of the present day CFD codes which are based on the finite volume method, two types of mesh are utilized: structured/ block-structured grid or unstructured grid systems for discretizing the computational domain. It is a matter of choice to decide on what kind of grid structure or solution algorithm is more

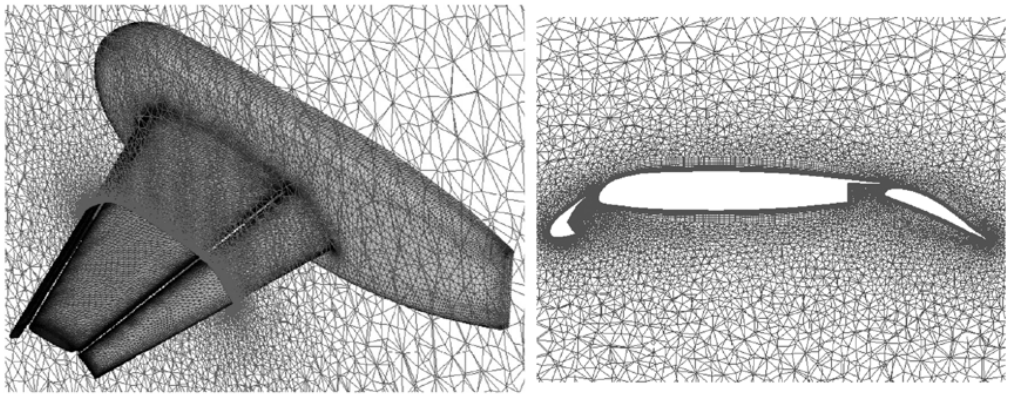
suitable for any specific application. This is because every method has relative advantages and disadvantages compared with one another for each case. Traditionally, structured/ block-structured method has been used, which rely on regular arrays of quadrilateral or hexahedral elements in each two or three-dimensions, respectively. Unstructured grid methods originally emerged as an alternative to structured grid methods for discretizing complex geometries.

Figure 1.1 shows examples of structured and unstructured grids for a complex geometry. This trapezoidal high-lift wing model was tested at NASA Langley in 1998 and NASA Ames in 1999. The multi-block structured grid consists of 586 blocks with about 7.5 million mesh points. More than one month was required to generate the mesh by an expert. On the other hand, the unstructured grid was generated with 13 million mesh points and it took only a few days from CAD data. The more number of mesh points were used on the unstructured grid than the structured grid because it usually requires a larger number of mesh points at the leading-edge and trailing edge sections [1].

With greater flexibility in meshing technique, unstructured grid CFD has been used in diverse applications and easily expanded outside the realm of pure research through most commercial CFD codes in engineering field. However, unstructured grid CFD usually requires more computational costs than structured CFD to solve the same number of unknowns, and this issue is still an important research topic in CFD field. As current CFD methods are aimed to simulate large three-dimensional viscous flow cases, more efficient solution algorithms associated with unstructured grid techniques are highly desirable.



(a) Multi-block structured grid



(b) Unstructured grid

Figure 1.1: Computational mesh for trapezoidal high-lift wing model (left: perspective view and right: cross-sectional view at 50% span [1].

## 1.1 Background

### 1.1.1 Concept of Structured and Unstructured Grid for Flow Simulation

A structured grid has identical connectivity for all interior nodes and elements as a finite difference type grid. All grid points lie on the intersection of two (or three) families of lines, which is considered as defining curvilinear coordinate lines. The goal of creating grids is to ensure every interior node is connected to four (or eight) elements respectively. This results in the type of element to be quadrilateral (or hexahedron). With this inflexibility, the generation of a fully structured grid has substantial disadvantages, especially on a complex geometry or if one requires local modifications (mesh adaptation). Although this limitation leads to the concept of multi-block structured grids, where the domain is divided into sub-regions, the automated identification and creating of the blocks are non-trivial.

Conversely, an unstructured grid can have nodes and elements with irregular connectivity. As a finite element type mesh, it is formed by any type of element, and a combination of triangular and quadrilateral cells (or tetrahedron and prism) is typically used. Unlike a structured grid, the mesh points cannot be identified with coordinate lines, therefore, they cannot be represented by a set of integers, such as $i$, $j$. Consequently, unstructured grid techniques for flow simulation employ a generalized indexing schemes which requires an additional grid information such as connectivity between cells, nodes, and faces. On the other hand, the unstructured

grid technique can provide greater flexibility for discretizing a complex geometry as well as the easy implementation of adaptive meshing techniques, where mesh points may be added, deleted, or moved while mesh connectivity is updated locally.

### 1.1.2 Implicit Time Integration

The simplest method of integrating the system of equations from space discretization is the use of an explicit time marching scheme. In this form, the time derivative is discretized using a finite difference formula at the current time step (n) and the residual is evaluated at the same time step. Runge-Kutta methods are one of many representative explicit methods used in flow simulations. Although the explicit schemes have the advantage of requiring only a simple update, the allowable time step size is limited for the reason of stability. For recent high Reynolds number flow simulations, the explicit time marching methods are too restrictive in terms of time steps and efficiency because of the demand of adequate boundary layer and complicated geometric resolution.

Therefore, implicit time marching methods are widely used for more practical problems. In an implicit method, the residual is evaluated at the next time step (n+1) as a backward finite difference, which results in unconditionally stability in terms of time step size. The Euler implicit method is a representative implicit method. After employing an Euler implicit method, the nonlinear vector of the next time step residual is commonly linearlized in time with a first-order discretization. The linearlization results in a large system of linear equations. To increase time

accuracy a subiteration scheme must be used to reduce the linearlization error.

In the case of a structured grid, the form of the matrix is rather simple because of its regular connectivity, such as a block pentadiagonal form for two-dimensions. Generally, a line-implicit method is used for solving the system; approximate factorizations are made to the linear system itself which results in multiple block tri-diagonal systems. For example, the Alternating Directional Implicit (ADI) method is often used as a factorization method and the resultant block tri-diagonal systems are solved using the Thomas algorithm.

On the other hand, the form of the linear system of equations is very sparse in an unstructured grid. It is hard to apply approximate factorization due to the irregular grid connectivity. The large sparse linear system requires quite expensive computations to solve at each time step. Therefore, an explicit time marching is generally used for an unstructured grid. Although implicit methods have been utilized using simple iterative schemes such as point Jacobi and Gauss-Seidel, it could not achieve similar efficiency with their structured grid counterparts. This is because their convergence rate degrades dramatically with grid size using the local technique.

To enhance convergence rates and efficiency, more advanced iterative techniques such as the multigrid technique or a Krylov method have been used on unstructured grids. Multigrid techniques consist of an operator to be defined on a sequence of coarser grids, an iterative method that evolves the solution, and an interpolation operator that transfer information between the grids. As a result, both low and high-frequency errors are damped efficiently through the successive process.

Generalized Minimum Residual Method (GMRES) is the most widely used method in flow simulations and is one of the Krylov methods. From the approximate solution of the system, GMRES finds the best possible solution over Krylov subspace by solving a minimization problem. Typically, preconditioning of the matrix is done first to minimize the required size of the subspace because both storage and the number of operations increase linearly and quadratically as the size of the Krylov subspace increases.

These advanced iterative techniques for unstructured grids require additional process and memory storage compared to the line-based implicit method for structured grids. Implicit time marching schemes for unstructured methods are still an active area of research.

### 1.1.3 Spatial Discretization

In most unstructured grid CFD solvers, a finite-volume strategy is used as a spatial discretization technique and the numerical algorithms typically involve a reconstruction step to achieve the desired spatial accuracy. A distinction between vertex-based and cell-centered schemes can be made depending on the location where flow variables are stored. In a cell-centered scheme, each mesh-element represents a control volume, and the flow variables are located at the centroid of cell elements. On the other hand, a vertex-based scheme stores the flow variables at the vertices of the mesh and uses the concept of a dual mesh as the control volume.

Finite volume discretization is formed by integrating the fluxes over the bound-

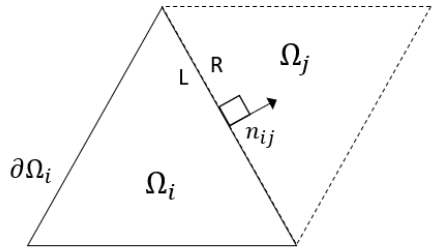



Figure 1.2: Possible candidate for finite control volume, indicated by $\Omega_i$, its boundary $\partial\Omega_i$, normal vector $n_{ij}$, and neighbor cell-$j$.

ary of the control volume as shown in Fig. 1.2. The interface fluxes are computed using the values on either side of the interface which are often referred to as "left" and "right" values in Riemann-based methods (see Eq. 1.1). Here, the reconstruction is required to evaluate the "left" and "right" values from the flow variables which are stored in each control volume.

$$(\vec{F}\Delta S)_{i+1/2,j} \approx \vec{F}(\vec{U}_L, \vec{U}_R, \Delta S_{i+1/2,j}) \tag{1.1}$$

where $\vec{F}$ is the convective flux vector, $\vec{U}$ is the vector of flow variables and $\Delta S$ is the area of the face.

Typical unstructured flow solvers are limited to second-order schemes which involve only nearest neighbors as stencils. It assumes the flow variables vary linearly over the control volumes, and the interface values must be extrapolated from centroidal values to the interface. This is known as gradient-based reconstruction. The gradient for linear reconstruction is usually computed using either least-square or Green-Gauss, along with limiters [2]. Higher-order schemes may be formed sim-

ilarly by using higher-order derivatives of the solution variables. However, it has not been used much in three-dimensional practical problems because it requires a much larger stencil and thus a much larger number of unknowns compared to a second-order scheme.

On the other hand, the use of a high-order scheme is relatively much easier on a structured grid. This is because the line reconstruction methods are applied along grid coordinate directions, and it only requires adding extra stencils along the line for a high-order scheme. It is known as stencil-based reconstruction. As examples, Monotone upstream conservative limited (MUSCL) [3] and weighted essentially non-oscillatory (WENO) [4] schemes are widely used. The use of these line reconstructions on the finite volume method does not guarantee its formal order of accuracy on the mesh system with varying curvature and stretching. However, less computational resources are required compared to gradient-based reconstructions. Details of the methods are explained in section 3.2.

### 1.1.4 Multi-Mesh/Multi-Solver Paradigm

The use of multiple domains with overset technique has become prevalent in current CFD solvers, which allows for relative motion between the domains and efficient grid generation. Typically, the multiple domains were generated using the same type of grid: either structured or unstructured. However, more recently developed multi-mesh/multi-solver paradigms combine the advantages of different grid types, while avoiding the associated disadvantages [5]. They combine unstructured

or curvilinear structured grids in the near-body region and adaptive Cartesian grids in the off-body region through an overset domain connectivity. Further advantage of grid generation for complex geometries can be obtained by employing unstructured grids as compared with structured grids.

Helios software [5] is one well known example of the use of the multi-mesh/multi-solver paradigm; which is the High Performance Computing Modernization Program (HPCMP) Computational Research and Engineering for Acquisition Tools and Environments ($CREATE^{TM}$) for Air Vehicle (AV), rotary-wing. In the Helios framework, a well-established structured or unstructured grid based solver (OVERFLOW and NSU3D, etc.) is employed in the near-body domain and each solver is modularized to be easily plugged in or out from the integrated framework. For the off-body region, the ARC3D code is applied which uses efficient high-order finite difference on a Cartesian grid. The high-order scheme on the Cartesian grid is an effective way to preserve wake structures generated from the near-body domain with much fewer grid points. Also, finite difference based high order schemes on a Cartesian grid is only marginally more expensive as compared to a second-order scheme [6].

Figure 1.3 shows an example of the use of a multi-mesh/multi-solver paradigm using the Helios software for a full UH60 rotorcraft simulation. Unstructured grid methods are used in the near-body domain for the region encompassing the fuselage and blades; while a Cartesian mesh is used in the off-body region. The unstructured solver interfaces in an overset manner with a background structured Cartesian high-order solver. The Cartesian solver can apply adaptive mesh refinement (AMR) to improve computational efficiency further as shown in Fig. 1.3.

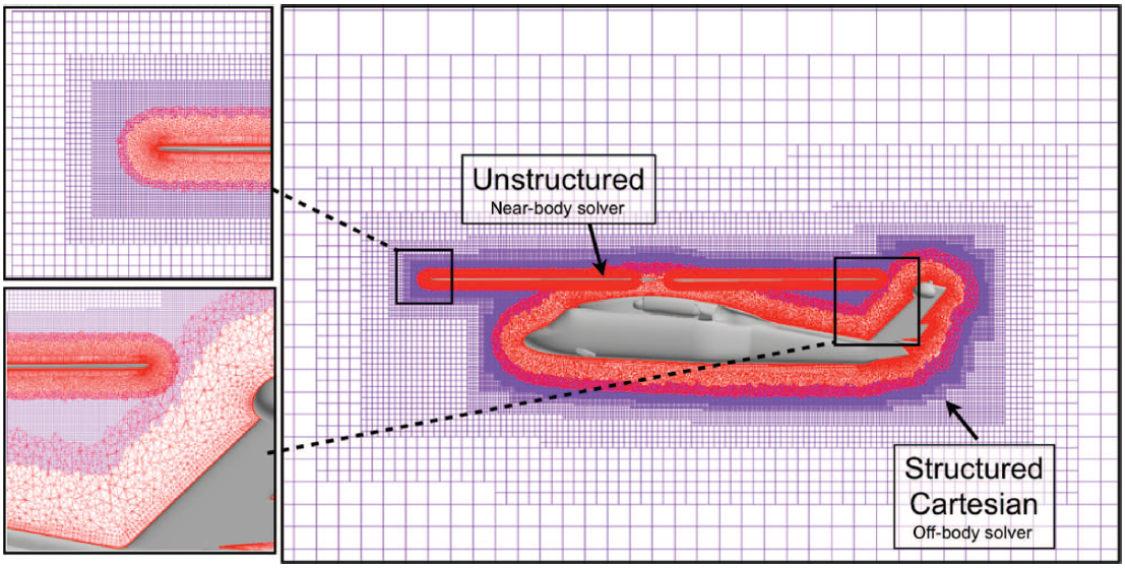



Figure 1.3: Multi-Mesh/Multi-Solver paradigm: Unstructured near-body with adaptive Cartesian off-body [6].

Helios adopts a high-level Python-based software integration framework to implement multi-mesh/multi-solver paradigm efficiently. Within the framework, each of the flow solvers is called as a library and data exchange between the various solvers are performed. The advantage of the Python-based approach is that it allows for loose integration of the component solvers, thus each component is easy to be incorporated/ separated as it is plugged-in/ out. Each solver can utilize its own preferred language and data structures and can be developed separately. Data exchanges are executed without memory copies or file I/O and the infrastructure is run in parallel on large multiprocessor computer systems. Python code is only used at a high level for method calls and setting pointers and all computationally intensive numerics are done inside the component solvers [6].

Although the multi-mesh/multi-solver paradigm can provide more efficient

meshing technique and solution algorithm compared to a single-solver, the limitations associated with unstructured grid based solvers still remain in the near-body region. For example, an unstructured flow solver showed more dissipated wake structures due to lower accuracy in space and less solution convergence rate compared to a structured flow solver in the references [5, 7].

## 1.2 Related Previous Works

Numerous studies have been conducted to improve solution algorithm associated with unstructured grid based solvers, especially for solution accuracy and convergence. To use high-order schemes in space, several methods have been proposed in addition to finite-volume method: finite-element type disretizations, spectral volume method, spectral element methods, etc. However, all of these methods have large penalties in terms of efficiency and robustness and are still limited to more pure research problems.

To accelerate solution convergence, Newton-Krylov solvers or multi-grid techniques are generally used. However, they are not as efficient as line-implicit methods of structured grid solvers, especially for unsteady flow simulations. In this section, several previous studies are introduced, which tried to take advantage of line-based methods on unstructured grids using pseudolines. However, most of the solution algorithms showed limited success because of the difficulties in identifying lines in pure unstructured grids. Even when they were identified, it proved difficult to achieve nesting that is required for approximate factorization methods. Furthermore, the

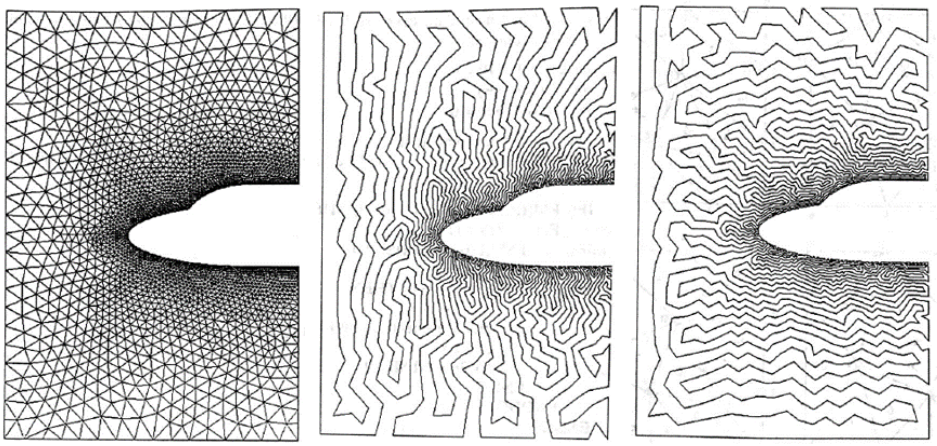
identification of lines was not unique.

Hassan et al [8] first constructed continuous lines by connecting neighbor edges of an unstructured triangular mesh. These lines were also called as a "snake line" which passed once through each node of the unstructured mesh (See, Fig. 1.4). On a two-dimensional grid, a total of two lines were generated to be used for implicit solution algorithm; one is for vertical and the other is for horizontal direction. Then, the system of equations was solved using a line relaxation procedure.

In order to use line relaxation along each line, renumbering on the nodal unknown was required in the order implied by the line. During sweeping each line, a block tridiagonal system of equations was solved at every time step. For a three-dimensional grid, the implicit solution algorithm was combined with an explicit method. The implicit method was employed only for the region in the vicinity of a solid surface, otherwise an explicit method was used for the remainder of the domain. This is because the line construction was not robust enough to be applied on unstructured volume elements, such as tetrahedron.

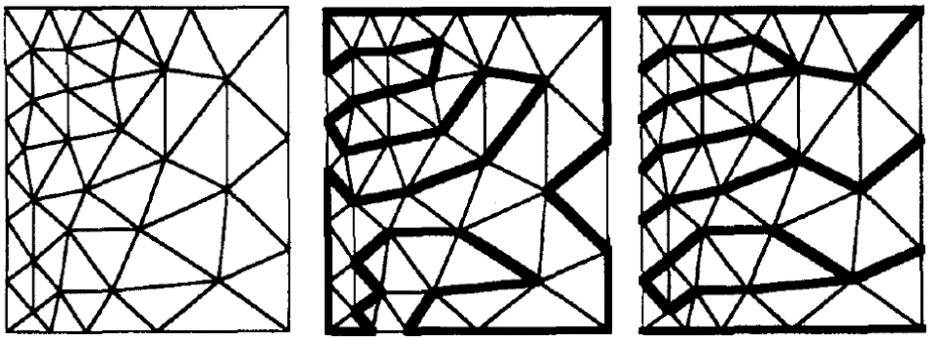
Martin and Lohnor [9] suggested an improved line generation method for a two-dimensional unstructured domain based on the previous snake line by Hassan et al. They re-generated the line structures in a way that flow information could be propagated quickly for better solution convergence. They called this line structures as a "linelet". In this previous study, the information could indeed be propagated to the boundary in a faster way using linelets as compared to snake lines.

Figure 1.5 compares the snake line and linelet on the same two dimensional unstructured grid. The snake line, which often exhibit folding on an unstructured



(a) Unstructured grid (b) First line (c) Second line

Figure 1.4: Construction of relaxation lines (snake line) on an unstructured grid [8].

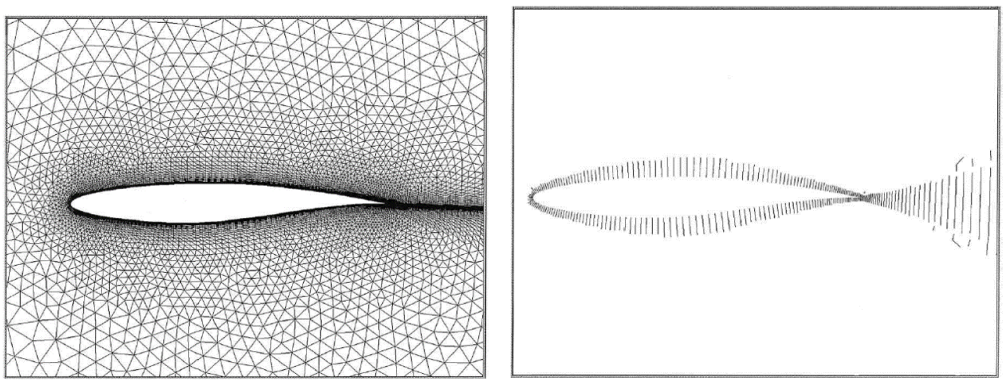


(a) Unstructured grid (b) snake line (c) linelet

Figure 1.5: Comparison of snake line and linelets on an unstructured grid [9].

grid, was broken into multiple linelets and the scheme was made implicitly along these linelets. Although the sensitivity to the orientation could be alleviated by using multiple linelets with different orientations, the approach was not satisfactory because the direction of propagation was still predetermined. Without robust line identification which achieves nesting, both methods showed limitations on general flow simulations.

Mavriplis [10] utilized line structures for the implicit method on high Reynolds



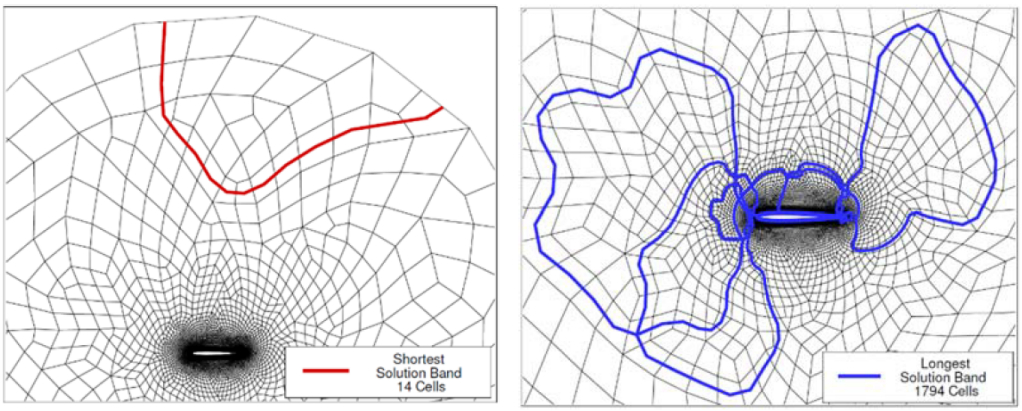
(a) Unstructured grid for RAE 2822 airfoil (b) Line structures for implicit method

Figure 1.6: Directional-implicit line method for RAE 2822 airfoil simulation [10].

number flow simulation. Most of the line structures were generated along the wall normal direction in the viscous grid region of a hybrid mesh system. In order to do this, the computational domains were grouped by using a weighted-graph algorithm, and the line structures could be made only on the highly stretched quadrilateral cells in the vicinity of the wall using the algorithm. Then, a line implicit method along the line structures was combined with an explicit method for the remaining region, which is called a directional-implicit line scheme.

Figure 1.6 (a) shows the unstructured grid over a RAE 2822 airfoil with fine initial wall normal spacing of $10^{-6}$ chord length. In Fig. 1.6 (b), the line structures for the implicit method were generated mostly on the highly stretched cells using the weighted-graph method as shown in the right figure. As a result, the use of the line implicit method for that region nearly doubled the convergence rate compared to an explicit method for the whole domain. Also, the directional-implicit method was unaffected by the degree of grid anisotropy in the vicinity of the wall.

Alternating cell direction implicit formulation (ACDI) [11] was proposed to



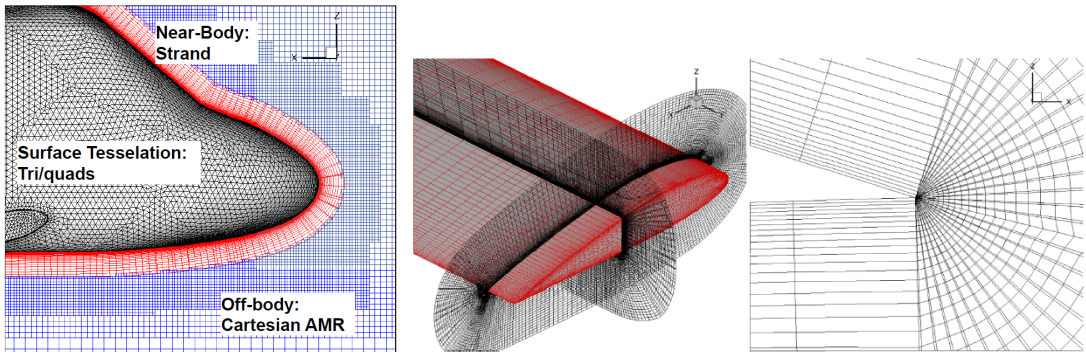

(a) Shortest solution band (b) Longest solution band

Figure 1.7: Solution bands for ACDI method on a unstructured airfoil grid [11].

apply a line implicit method on a two-dimensional unstructured grid including mixed type quadrilateral and triangular cells. In this algorithm, a line implicit method similar to line Gauss-Seidel was applied along a chain of contiguous cells which was called the solution band. The direction of the solution band was designated by passing through opposite faces of cells, therefore, the line could achieve nesting for factorization.

Figure 1.7 shows two examples of solution bands on an airfoil unstructured grid with mixed elements. Each solution band has it's own length, and the shortest band and the longest band is shown on the left and the right, respectively. This method is limited in the two-dimensional domain and remained in the conceptual stage.

For the line-structure on a three-dimensional domain, the concept of strand grids has been introduced [12]. The strands could be projected in the wall-normal direction from any surface tessellation. As a result, it provided automated viscous




(a) Strand grid for UH60 fuselage (b) Multi-strand grid for Martin rotor blade

Figure 1.8: Examples of strand grids for near-body volume domain [13].

volume mesh generation and line structures for a line-based flow solver. Recently, Helios software [6] adopted the concept of strand grid in the near-body domain. As shown in Fig. 1.8 (a), the strands extruded a short distance from the solid boundary and intersected with adaptive Cartesian grids, which covered the rest of the domain to the outer boundaries. The strand and the Cartesian grids were connected though an overset interface.

For further extension, the concept of multi-strand grid has been developed for the region of a sharp corner such as a trailing edge, where the multiple strands were extruded from a single node on the surface. (See, Fig. 1.8 (b)). As one of the near-body solvers in the Helios framework, mStrand [13] used the strand grid for both accelerating solution convergence and automated mesh generation. While the strands grids were used for the line-based method in the wall-normal direction, the other two surface directions on the unstructured grid limited the achievable gains without identified line structure in the wall-tangential directions.

Recently, the concept of Hamiltonian paths has been developed on a purely

unstructured two-dimensional triangular mesh to obtain lines [14]. The primary idea of this method lain in the subdivision of triangles into multiple quadrilaterals. Based on quadrilaterals which possess even number of mutual edges, a unique set of paths were identified by connecting edges to serve as the surface line structures.

Figure 1.9(a) shows the initial unstructured grid with triangles around a NACA0012 airfoil. As the first process for identifying lines, each of the triangle elements has been sub-divided into 12 quadrilaterals as shown in Fig 1.9 (b). Then, colored loops, which is called as "Hamiltonian paths", are identified by connecting opposite face centers along the quadrilateral cells as shown in Fig 1.9 (c). The resultant line structure became loop-shaped as shown in Fig 1.9 (d) and the associated flow solver uniquely used both stencil-based reconstruction and a line implicit method along the Hamiltonian path. Once the process was finished through the whole domain, each edge of a quadrilateral element is part of a unique Hamiltonian path and two distinct Hamiltonian paths intersect at each quadrilateral cell center as shown in Fig 1.9 (e).

This in turn facilitated stencil based discretization and approximate factorization of the implicit operator. The methodology showed promising results for both solution convergence and computational cost when it was compared with a structured grid based flow solver on a transonic inviscid airfoil flow simulation. However, the application was limited only to two-dimensions and execution on a serial processor. To be competitive for realistic applications, this methodology needs to be enhanced for application in three dimensional flows and execution on massively parallel computer systems.

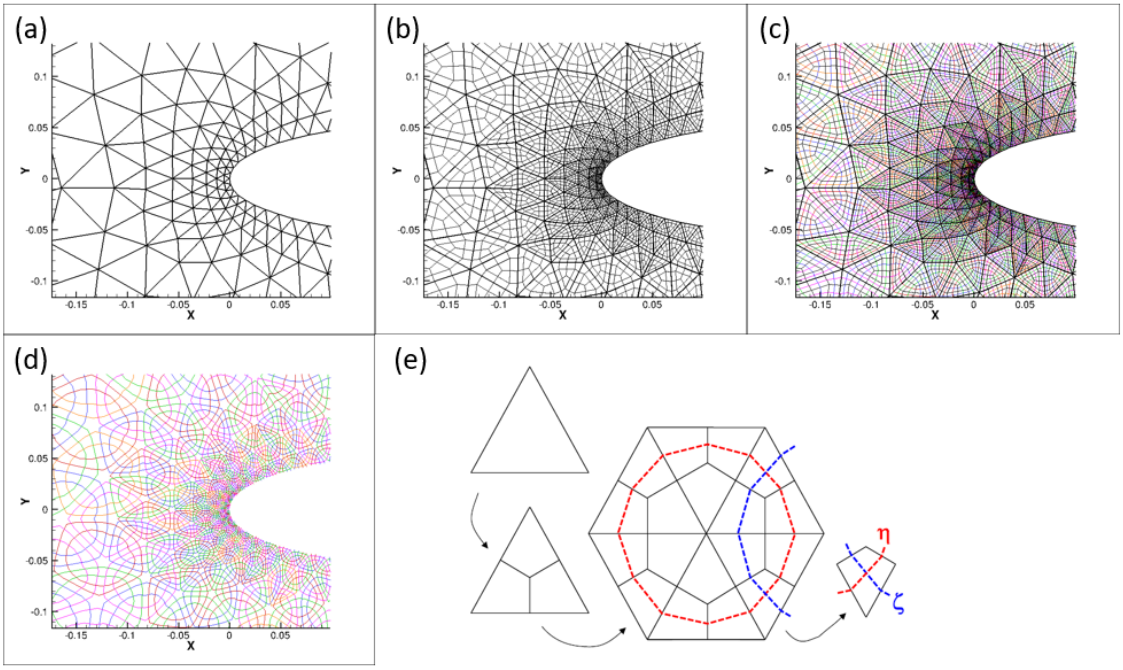



Figure 1.9: Schematic showing quadrilateral sub-division and construction of Hamiltonian paths [14].

## 1.3 Objectives

The first objective of the present study is to investigate robust line identification on general unstructured grids in order to use efficient line-based methods such as stencil-based reconstructions and line-implicit time marching methods on two- and three-dimensional domains. In order to do this, the following objectives are accomplished.

- Two-dimensional Hamiltonian path approach is extended to general two- and three-dimensional unstructured grids with mixed elements. In the previous study, the robust path identification was achieved by ensuring the even number of edges per two-dimensional element. Therefore, each element should be converted to quadrilaterals/hexahedra in the mesh generation process.

- A third direction of line structures is identified by employing the strand grids from the Hamiltonian paths on the unstructured surface mesh. The nature of a strand grid allows line-based method to be used along the wall-normal direction. Also, the Hamiltonian paths on the surface can be preserved in each strand layer, which results in efficient data access inside the flow solver.

- The solution accuracy and the convergence rate are investigated through various flow conditions: inviscid transonic, laminar, and turbulent flows. Depending on the flow condition, proper grid systems are required to better capture the flow features, such as boundary layer, wake structure, shock, etc. The numerical efficiency of a line-implicit method and the solution accuracy of stencil-based reconstruction are compared with the conventional flow solver techniques on the unstructured grid. Similarly, a comparison study with an in-house structured grid based flow solver is also conducted to evaluate the performance of the current flow solver.

The second objective of the current study is to develop a CFD framework for investigating practical problems involving rotary wing systems with overset meshes.

- The developed flow solver and mesh generation code are both parallelized such that it can be executed on distributed computer memory systems. The overall grid is evenly divided into multiple subdomains and the communication between multiple subdomains is performed using a message passing interface (MPI). The concept of ghost cell is adopted along the boundary between subdomains.

- The mesh system is extended to utilize overset meshes. The overset technique allows for multiple mesh systems, which consists of a near-body Hamiltonian/Strand grid and off-body Cartesian nested meshes. Using the overset mesh system, the current flow solver is integrated with the other solver within the multi-mesh/multi-solver paradigm. The integrated framework provides more efficient meshing technique and solution algorithm than a single-solver.

- The capability of the flow solver is extended for unsteady viscous flow simulation with grid motion. The Reynolds-Averaged Navier–Stokes (RANS) formulation is used by applying the turbulence model. In order to allow for laminar-turbulent boundary layer transition, a transition model is also coupled with the turbulence model. For the time-accurate flow simulation capability, a second-order time marching method with dual-time-stepping strategy is explored.

## 1.4 Scope and Organization of Thesis

This thesis is focused on extending a two-dimensional solution algorithm using Hamiltonian paths to three-dimensional RANS method for the interactional aerodynamic problems with overset meshes. The current methodology is improved and verified in both parts of mesh generation and flow solver and the current thesis is organized as follows.

Chapter 2 describes the computational methodology for the mesh generation. The improvements in terms of robust line identification, extension to three-

dimension, smoothing technique and domain decomposition are detailed.

Chapter 3 describes the computational methodology for the flow solver. It includes the evaluation of both inviscid and viscous fluxes on the finite volume method, implicit operators, turbulence and transition modeling, the flow solver architecture, and the Python integration framework using overset method.

The verification and validation of the developed methodology is presented in Chapter 4. The solution accuracy and performance (solution convergence rate and execution CPU time) of the flow solver are investigated through various cases with different types of grid and flow flow conditions. Especially, the computational efficiency of line-implicit method over point-implicit method is validated.

In Chapter 5, the solution convergence rates are further evaluated for the case of loop crossing. Preconditioned GMRES method is also applied in the current method to improve solution convergence rate further. The performance of the GMRES method is validated by comparing the result with the line-implicit method alone. Finally, the parallel efficiency of the flow solver is tested through the strong scalability test.

The developed framework is applied to rotary wing problems: hovering isolated rotor, full wind turbine configuration, rotating rotor hub, and high advance ratio forward flight rotor simulations. Within the multi-mesh/multi-solver paradigm, the current solver is used for the near-body domains with complex configurations. The approaches and the following results are presented in Chapter 6.

Conclusions and observations noted during the development, validation, and application of the methodology are summarized in the Chapter 7.

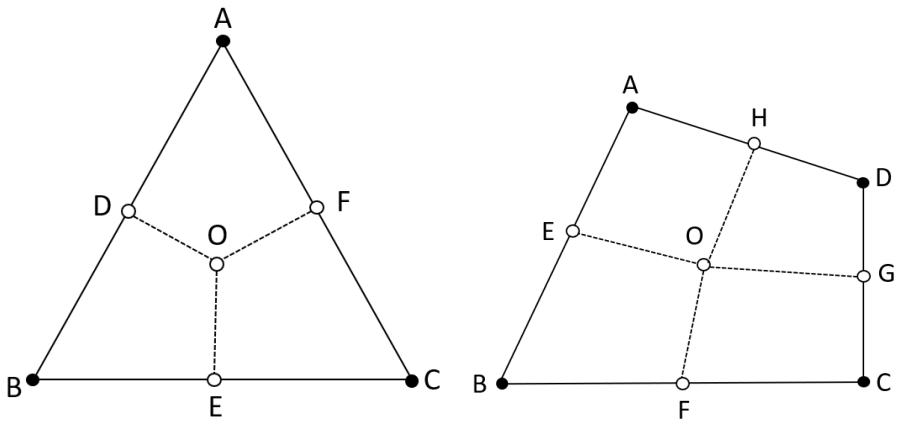
# Chapter 2: Computational Methodology of Mesh Generation

In this chapter, the numerical methods for the current mesh generation code are described. The chapter will initially explain the robust path identification method for general structured and unstructured surface grids. The mesh smoothing and all quadrilateral meshing techniques are also included in this chapter. Following this, the methods for the extension to three-dimensions are discussed. Finally, the domain decomposition method for the current mesh system will be described.

## 2.1 Robust Line Identification

### 2.1.1 Quadrilateral Subdivision and Hamiltonian Path

The generation of Hamiltonian paths has been extended to general two dimensional unstructured grids with both quadrilateral and triangular elements. The process begins by subdividing every triangle into three quadrilaterals and every quadrilateral in the original mesh into four additional quadrilaterals. This subdivision is accomplished by connecting the midpoint of each edge and cell-centroid as shown in Fig. 2.1. Figure 2.1(a) shows the subdivision process on an initial triangle element which is consisted of nodes A, B, and C. Each triangle is divided into three




(a) Initial triangular element (b) Initial quadrilateral element

Figure 2.1: Quadrilateral subdivision on a two-dimensional element.

quadrilaterals by creating edges of DO, EO, and FO (quad-level 0). As shown in Fig. 2.1(b), an initial quadrilateral which is consisted of nodes A, B, C, and D is also divided into four smaller ones by connecting each midpoint edges and cell-centroid. Once a pure quadrilateral mesh is obtained from the initial mesh, each of these quadrilaterals can be refined further into four additional ones if desired using the same approach as shown in Fig. 2.1(b). Then, resultant 12 quadrilaterals are made from an initial triangle and 16 quadrilaterals from an initial quadrilateral (quad-level 1). In the end, the goal of the subdivision process is to arrive at an even number of mutual edges for each cell element starting from an arbitrary unstructured mesh which can consist of mixed triangle and quadrilateral elements.

The process of Hamiltonian path identification for a given surface mesh begins by identifying any edge that dose not have a chain passing through it and the cell associated with this edge. From the edge, the chain is grown by connecting midpoints of opposite edges in the quadrilateral cell, and this process is repeated until either a

closed chain is formed or the chain encounters a boundary (such as wall or far-field). Once all of the edges in the mesh have been linked to a chain, the path identification is complete. Figure 2.2 shows the resulting Hamiltonian paths which are colored line structures obtained from an initial mix of triangles and quadrilaterals. As a result, each edge is part of only one distinct loop and each cell centroid is intersected by two distinct loops. It is also recognized that, for a given mesh, there is only one possible topology of the resulting Hamiltonian paths, making the process unique and robust. These structures are analogous to the grid coordinate directions in a structured grid solver, thus similar solver strategy is applicable along the Hamiltonian loops on the unstructured grid. Most importantly, this quadrilateral subdivision process ensures an even number of edges per elements and guarantees that nesting for a factorization method will always be satisfied. Therefore, line-implicit methods are available along the hidden line structure on the unstructured surface grid.

This process allows the Hamiltonian paths to traverse grids that are both structured and unstructured. Figure 2.3(a) shows a point continuous multi-block system around a NACA0012 airfoil, which consists of a curvilinear structured grid, an unstructured grid, and a Cartesian grid. The curvilinear near body grid is generated using a hyperbolic mesh generator with 160 points in the wrap-around direction and 10 layers in the wall-normal direction. Then, the triangle elements are generated using the Delaunay formulation [15]. Then, the all quadrilateral mesh system is obtained using the subdivision method from the initial mesh system as shown in Fig. 2.3(b). By using quad-level 0 subdivision, each triangle is divided into three quadrilaterals and each quadrilateral is subdivided into four smaller quadrilaterals.

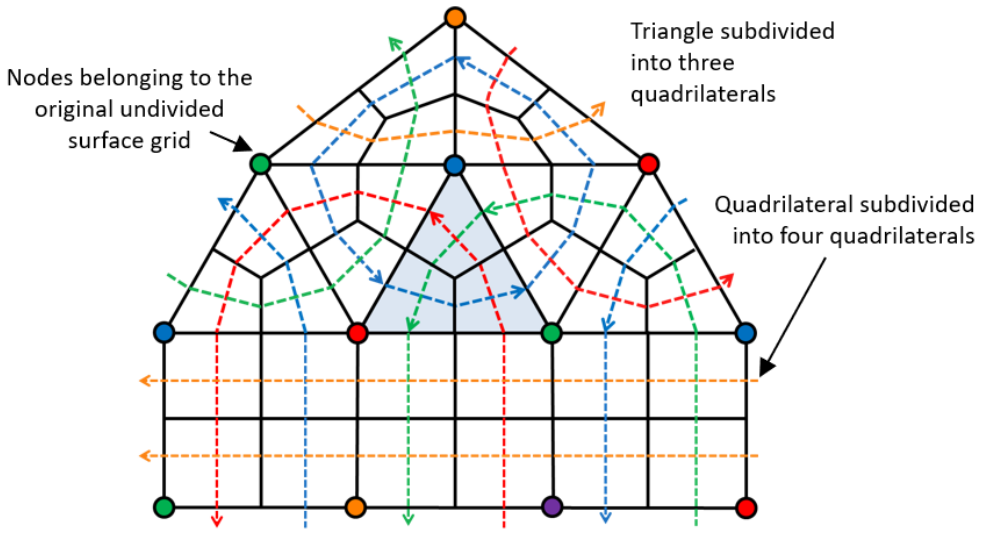
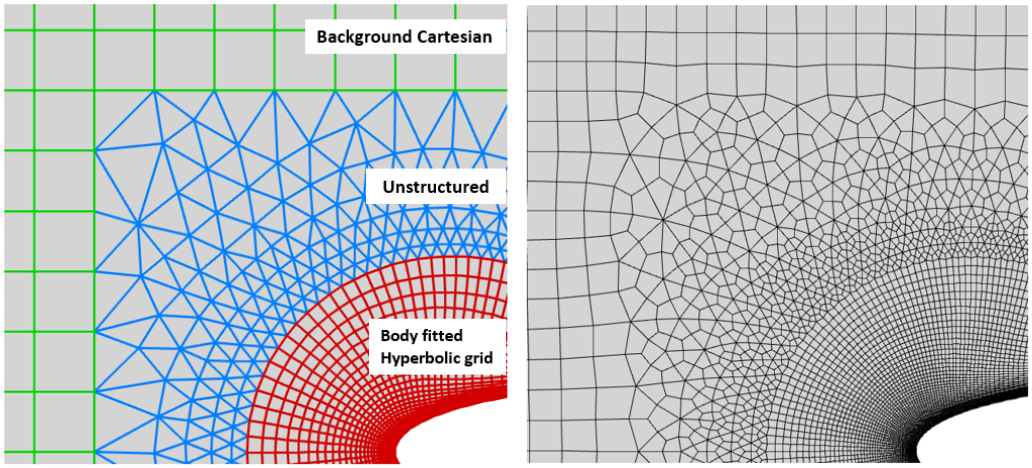



Figure 2.2: Schematic showing Hamiltonian loops formed on a mixed triangle and quadrilateral mesh.

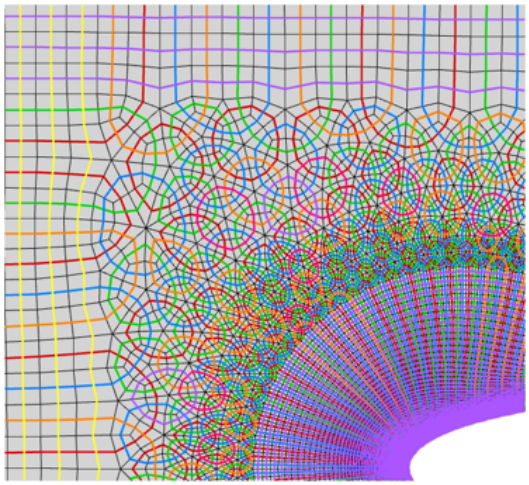
Note that the mesh has been smoothed using a simple Laplacian smoothing operator with fixed boundary nodes to improve the cell and Hamiltonian loop quality. Based on the pure quadrilateral mesh system, Hamiltonian paths are found as shown in Fig. 2.3(c). Here, the Hamiltonian paths are passing through the various domains, thus the mesh systems are united and treated as a single unstructured quadrilateral mesh. Each cell has two loops of distinct colors that pass through it, which act as the coordinate lines for each cell. Note that the Hamiltonian paths on the structured or Cartesian domains are similar with the grid coordinate directions used with a structured grid flow solver.

An arbitrary level of refinement (subdivision) is referred as quad-level in this study. After one additional quad-level, the resulting meshes have four times the number of quadrilateral cells with twice the number of Hamiltonian loops; each




(a) Point continuous multi-block system

(b) All-quadrilateral mesh system



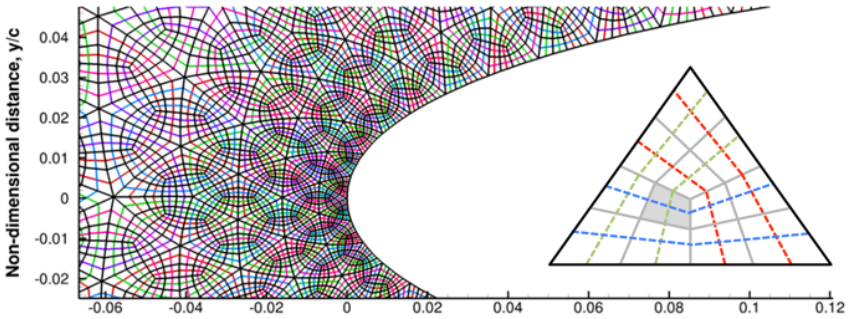
(c) Hamiltonian paths passing through the different domains

Figure 2.3: Hamiltonian path on multi-mesh system for NACA0012 airfoil.

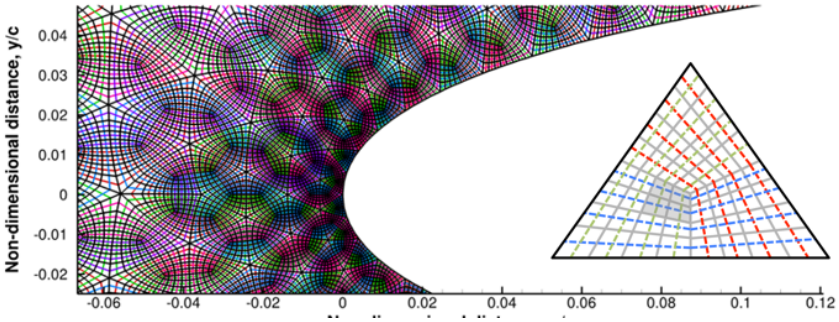
loop now being twice as long. Figure 2.4 shows the NACA 0012 airfoil grids which are subdivided to quad-level 1 and quad-level 2 from the same initial triangular mesh. In this example, the newly created points on the edges of the triangles are flushed to the airfoil surface using the analytic equations for the surface definition of a NACA airfoil. For the two-dimensional surfaces which are not defined by an analytic equation, a cubic interpolation is used to flush newly added points on the surface. Different quad-levels can serve as efficient ways to perform grid-refinement and grid-convergence studies. The additional subdivision of quadrilaterals can also be used to potentially improve the quality of the mesh and resultant Hamiltonian paths (For details, see reference [16]).

### 2.1.2 Hamiltonian Path on Structured Grids

The generation of Hamiltonian paths can be applied on any structured mesh directly because a structured mesh has pure quadrilaterals. The Hamiltonian paths can be assumed as grid coordinate directions of the structured grid in most of the regions. However, the path can across any *coordinate cuts* as a continuous line on the structured mesh, which is defined by the boundaries of computational space and required due to grid mapping from physical space. Figure 2.5 shows identified Hamiltonian paths on two different types of structured airfoil grids near the trailing edge; O-type and C-type meshes. The paths are denoted by red and blue colors and the structured mesh is represented by the black line. It should be noted that only two different colors are required for two-dimensional structure grids to make

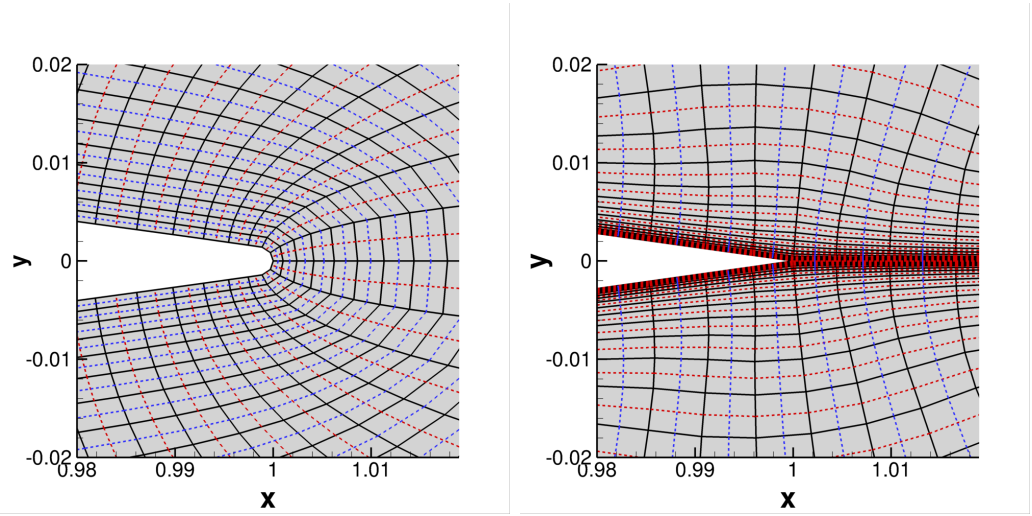



(a) Quad-level 1 subdivision




(b) Quad-level 2 subdivision

Figure 2.4: Multiple level quadrilateral subdivision and application for a two-dimensional airfoil mesh.




(a) O-type structured grid

(b) C-type structured grid

Figure 2.5: Hamiltonian paths on airfoil structured grids at near the trailing edge.

two loops of distinct color intersect at every cell center. In the figures, the paths shown as blue lines can cross the *coordinate cuts*, where an appropriate boundary condition is typically required in structured grid based solvers. Therefore, in the current method, *coordinate cuts* can be handled as a continuous domain without boundary conditions for both reconstruction and the implicit operator.

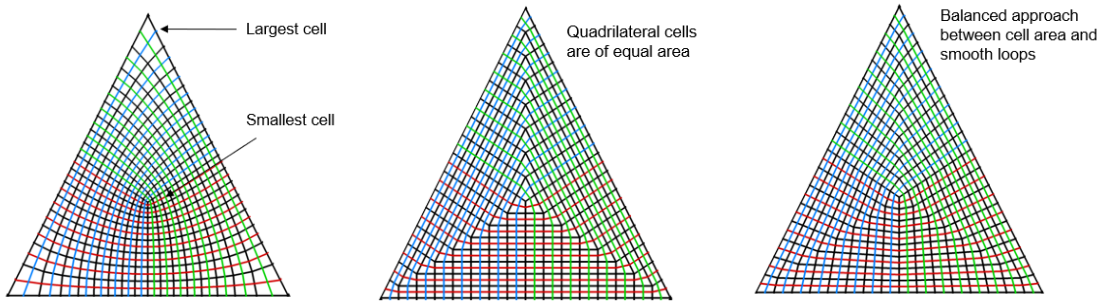
### 2.1.3 Mesh Smoothing

Ideal Hamiltonian loops have a high radius of curvature and the cell sizes are equal or changing gradually from one cell to the next along a given loop. An approach for increasing the radius of curvature is to use less number of initial triangular cells with multiple number of quadrilateral subdivisions to maintain the total number of quadrilateral cells. As quad-level increases, it is necessary to re-position the internal nodes to improve accuracy and convergence of the solution when high-

order stencil-based reconstructions are used along the loops. A naive implementation involves the placement of the internal nodes in any arrangement followed by a smoothing operation, such as a Laplacian smoothing, to ensure smooth Hamiltonian paths. In this smoothing, any given node that is not on the domain boundaries is repositioned based on the simple average of the position of the connecting nodes. The process is repeated until the positions do not change between subsequent iterations. However, Laplacian smoothing results in a wide variation of the quadrilateral cell size; the cells getting smaller towards the center and larger at the corners, as shown in Fig. 2.6(a). Figure 2.6(b) shows the distribution of nodes such that the resulting quadrilaterals are all of equal area, forming a shape-preserving pattern. Note that the loops formed through this distribution have varying curvature. Therefore, a blended mesh is obtained, as shown in Fig. 2.6(c), where the node positions of the previous two node distributions are averaged. The resulting blended mesh produced a distribution that is more appealing than either procedure by themselves and the quadrilateral cell sizes are more uniform than with the previous Laplacian smoothing. A comparison of the flow solution convergence between Laplacian smoothing and blended smoothing can be found in the reference [16].

### 2.1.4 All Quadrilateral Meshing

Current requirement for the robust line identification is an all quadrilateral mesh generation and the resulting quadrilateral meshes can be classified into several classes based on the degree of regularity as shown in Fig. 2.7. First, a regular mesh

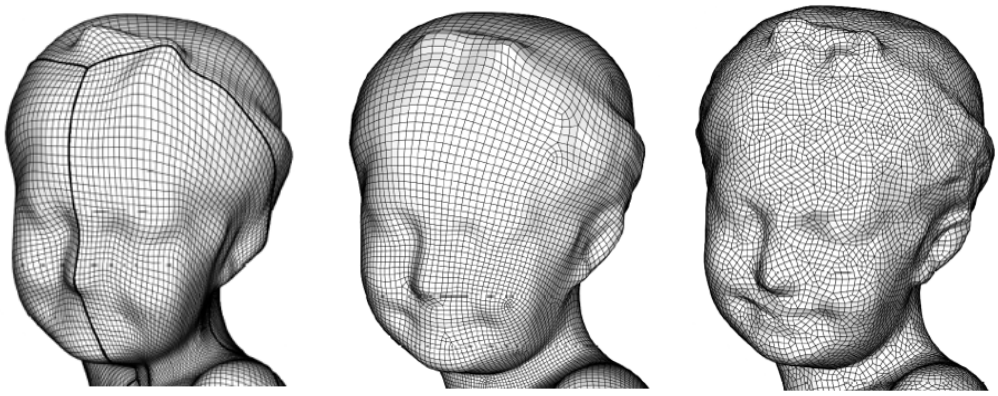



(a) Laplacian (b) Equal area (c) Blended

Figure 2.6: Various distributions of cells within a triangle.

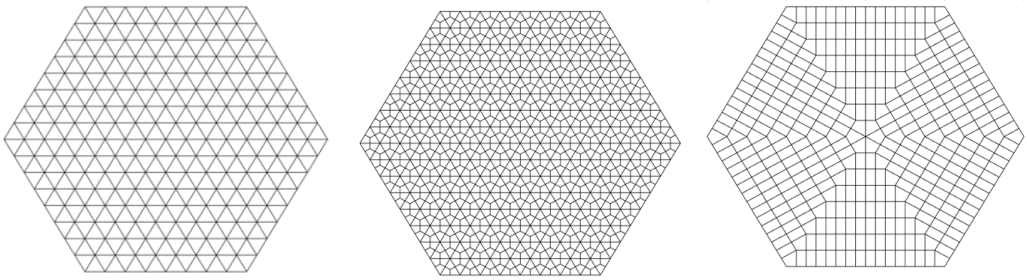
which corresponds to structured mesh can be globally mapped to a 2D arrays of quadrilaterals like computational space. In the regular mesh, all node points have four valence which is the number of its incident edges, and this limitation makes it difficult to be used on an arbitrary shape. Second, in a semi-regular mesh, most of its nodes have four valence, but not every node. Even with a small number of valence semi-regular, the mesh generation can be much simpler than for a regular mesh. Last, an unstructured mesh has a larger fraction of its nodes which are irregular compared to the semi-regular mesh [17].

The quadrilateral subdivision method from a pure triangular mesh corresponds to the unstructured all-quad mesh and it naturally provides circle-shaped loops which become rather short and curved when compared to the shape of loops on a structured grids (grid coordinate lines). For smoother and longer shape of paths from an initial triangular mesh, the HAMSTRAN method [18] has been introduced to mimic the semi-regular mesh type. In this method, triangles or mixed elements are converted to all quadrilaterals mostly by re-locating node points and recreating the nodal connectivity information. A subdivision process is not required in this



(a) Multi-block structured (b) Valance semi-regular (c) Unstructured

Figure 2.7: Quadrilateral meshes categories [17].



(a) Initial triangular mesh (b) Subdivision (c) HAMSTRAN

Figure 2.8: Different pure quadrilateral meshing technique [18].

method, thus the number of elements can be better preserved compared to the subdivision method.

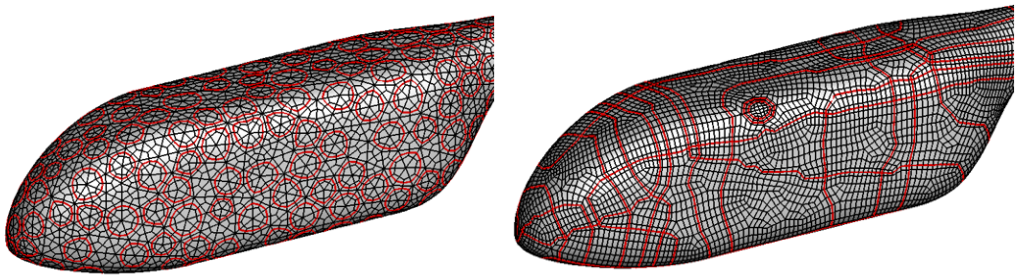
Figure 2.8 shows a simple example of two different quadrilateral meshing on a hexagon. From the initial triangular mesh with 384 elements, the pure quadrilateral mesh with 1,152 elements or 384 elements can be obtained using either subdivision or HAMSTRAN method. The HAMSTRAN method can ensure that most of the node points have valence four, as shown in Fig. 2.8 (c), which means better grid quality with less skewness for the pure quadrilateral mesh.

As shown in Fig. 2.9, part of the resultant paths using either the HAMSTRAN method or subdivision method have been compared with each other on the Robin-Mod7 fuselage surface mesh which originally consists of all triangles. The Robin-Mod7 fuselage was developed at NASA Langley to be representative of a generic helicopter. When it is compared to the mesh using the subdivision method (quad-level 0) in Fig. 2.9 (a), the quadrilateral mesh using the HAMSTRAN method becomes a more semi-regular type as shown in Fig. 2.9 (b). For the two different quadrilateral meshes, the shapes of resultant loops are quite different as well. The smoother and longer resultant loops on the semi-regular mesh can take better advantage of line based methods in the flow solver, such as for numerical accuracy and convergence (for details, see reference [18]).

One interesting phenomenon is that the paths on a semi-regular mesh can potentially self cross as shown in Fig. 2.9 (b). The current flow solver strategy can handle the self-crossing of loops robustly and the solution convergence was not affected by the self-crossing loops. Further investigation is shown in section 5.1.

## 2.2 Extension to Three-Dimensions

To extend the formulation to three-dimensions, two different types of volume grids have been employed in the present study: a strand based grid and a traditional unstructured volume grid.



(a) Unstructured mesh with quadrilateral subdivision

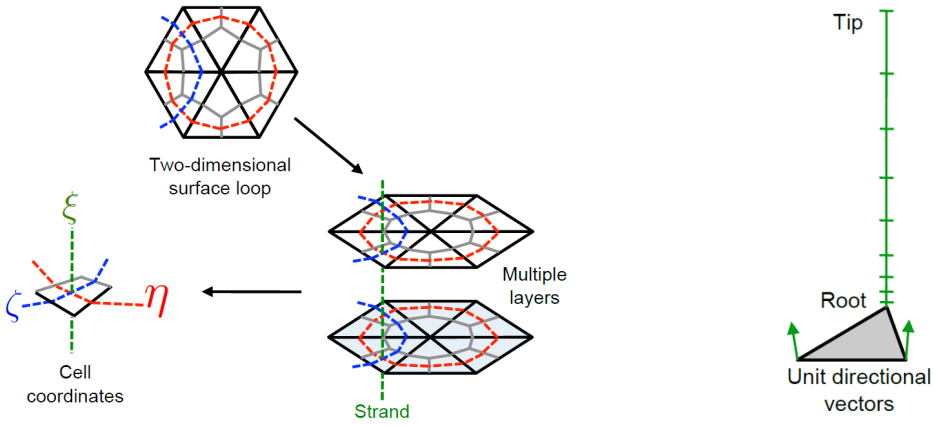
(b) Valence semi-regular mesh without quadrilateral subdivision

Figure 2.9: Hamiltonian path on two different quadrilateral mesh for Robin-Mod7 fuselage.

### 2.2.1 Strand Grids

The role of the strand grids is to provide a structure in the wall normal direction, thus allowing for line-implicit methods and stencil-based reconstructions to be used along this direction. Figure 2.10 shows a schematic of the construction of the strand grids from a triangular mesh element. This triangular element is subdivided into quadrilaterals and the construction of Hamiltonian loops around a given triangular node can be performed. Therefore, two loops pass through each quadrilateral and are the local coordinate directions. Strand grids emanate from the cells on the surface of the body and are extruded in the wall-normal direction. These strand grids pass through multiple layers of Hamiltonian loops and form the third cell coordinate direction, as shown in Figs. 2.10(a). Therefore, for a given surface geometry, Hamiltonian loops are constructed on the surface of the body representing two "in-plane" spatial directions and the strands represent the third "out-of-plane"

spatial direction. Each strand grid is based on the strand template as shown in Fig. 2.10(b). The strand template is identified as a wall normal direction vector and a one-dimensional grid point distribution along the strand which originates from each node on the surface mesh. The grid spacing on each strand grid is identical. Consequently, layers of hexahedra are generated by extruding the surface.




(a) Construction of strands (b) Strand template

Figure 2.10: Schematic depicting the creation of strand layers from multiple layers of Hamiltonian surface loops.

By using a strand grid to form the volume domain, Hamiltonian paths on the surface are preserved along the strand grid. Therefore, efficient solution algorithms are possible and storage requirements are minimal as the structured data on the surface grid is maintained in the wall normal direction. Furthermore, strand grids are also readily amenable to parallelization techniques. However, it is relatively difficult to use a strand grid around complex geometries, especially with a concave corner, as compared to a typical unstructured volume mesh which uses tetrahedrons. This is because the strands eventually cross each other as they extrude along straight lines

in the wall normal direction at a concave corner. Although strand-clipping technology [12] was developed to handle strands crossing over each other, an alternative strategy is employed in the present study.

This uses a modified advancing front-like technique to generate the volume mesh and prevent the stands from crossing over each other [16, 19]. In this formulation, the strands are no longer assumed to be straight lines but can be curvilinear, which means the normal direction of the strand can be varied along the strand. Strands are marched from the wall surface using the local normal to a finite distance to the next outer surface. At this new surface, the normals of the strands are recomputed and averaged based on the surrounding normals, following which they are marched out again to form the subsequent outer surface. This process of averaging the normals and marching out is repeated for the total number of strand layers in the volume mesh.

The resulting volume mesh around the Robin-Mod7 fuselage is shown in Fig. 2.11. 57 strands are used with an initial wall spacing of $5 \times 10^{-6}$ fuselage length as an initial wall normal spacing. However, the smoothing of the normals by itself is insufficient to prevent the normals from intersecting in the ramp region of the fuselage for relatively large cell sizes. Therefore, preferential weighting of the normals in the radial direction is applied to be necessary in the highlighted regions as shown in Fig. 2.11. The use of the weighting factor is dependent upon the simulation model geometry and whether it has concave corners or not.

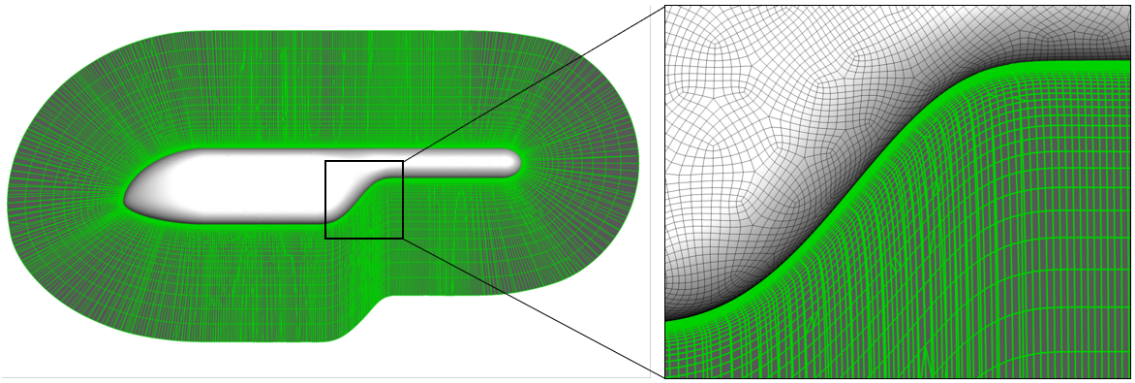


Figure 2.11: Longitudinal slice of the mesh around Robin-Mod7 fuselage highlighting the curved strands in regions of concave ramp region.

### 2.2.2 Unstructured Volume Grid

Second extension to three dimensions uses a typical unstructured volume mesh which usually consists of both prisms and tetrahedrons. Unlike a strand grid which is based on hexahedrons, robust line identification is not possible on a typical unstructured volume mesh, because each element does not have an even number of faces. Therefore, each initial element is subdivided into multiple hexahedrons in a manner similar to that in two-dimensions. Each tetrahedron is divided into four hexahedrons and each prism is divided into three hexahedrons. Then, Hamiltonian paths are constructed on the divided volume domain by connecting opposite face centers of each hexahedron, which is exactly the same algorithm as the one used for a two-dimensional surface mesh. Figure 2.12 shows an example of Hamiltonian path generation on two prisms and three tetrahedrons. After the subdivision process, a total of 18 hexahedra are used to form the Hamiltonian paths which are shown as colored loops. Unlike with a strand volume grid, a Hamiltonian path can be either

opened loop or closed loop. Compared to a strand grid, the use of both prisms and tetrahedrons makes the robust volume meshing to be possible around complex geometries which have concave corners. However, the Hamiltonian paths on surface are not preserved in the wall normal direction and the resultant loops on volume domain are formed with much shorter length and various curvatures. These are not favorable to line based flow solvers especially for stencil based reconstruction schemes.

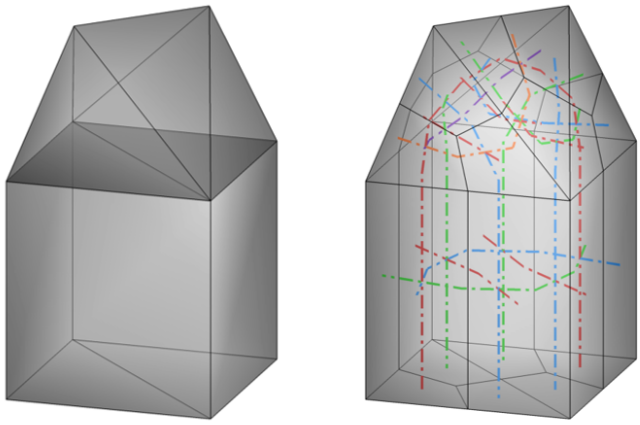
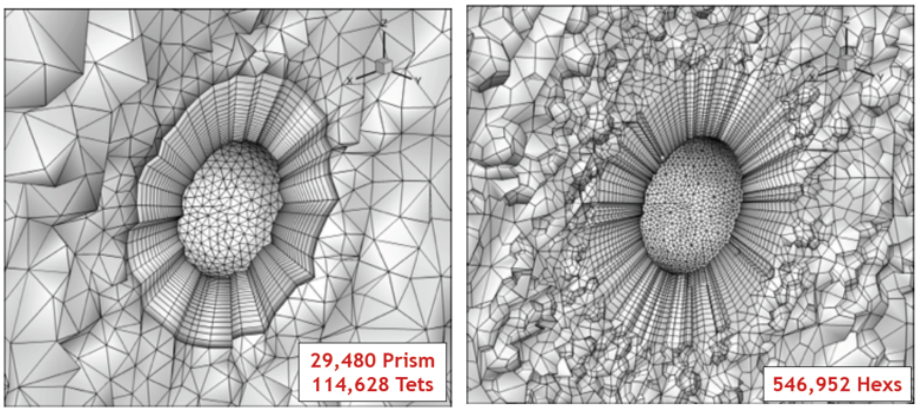


Figure 2.12: Schematic showing Hamiltonian paths on unstructured volume mesh (left: initial prisms and tetrahedrons, right: Hamiltonian paths on hexahedrons).

Figure 2.13 (a) shows an example of current unstructured volume meshing around a sphere. As a typical unstructured mesh, the viscous domain extrudes a short distance from the surface using prism layers with 29,480 elements and it is transitioned to tetrahedrons with 114,618 elements for the space between the outer surface of the prism domain and the far-field boundary. After subdivision (quad-level 0), a pure hexahedral mesh was obtained with 546,952 elements as shown in Fig. 2.13 (b). It is recognized that for the simple geometry of a sphere, the

unstructured volume grid may not be required. However, the goal of the example is to show the capability of the current subdivision and path identification methods in three-dimensions.




(a) Initial prism and tetrahedron volume mesh

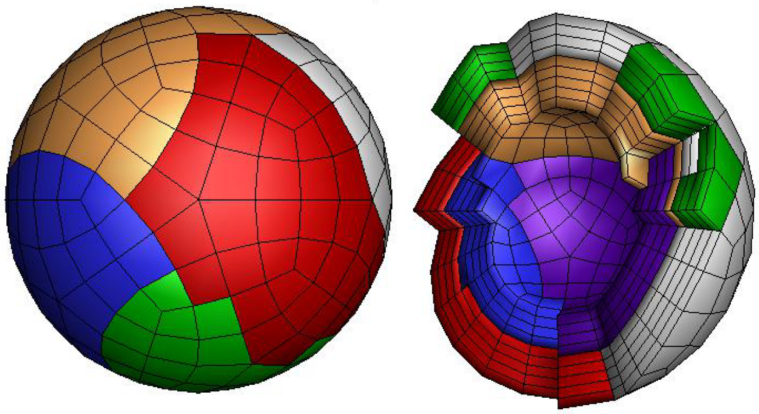
(b) All hexahedral volume mesh

Figure 2.13: Unstructured volume mesh around a sphere.

## 2.3 Domain Decomposition

Typical grid-based CFD meshes contain millions of volume cells making it impractical for the solution to be executed on a single processor. The developed Hamiltonian flow solver and mesh generation code are both parallelized such that it can be executed on distributed computer memory systems. The communication between multiple subdomains is performed using the Message Passing Interface (MPI). Therefore, the overall grid is split into multiple subdomains to be computed in parallel. METIS [20], an open-source program used for graph partitioning, is employed in the current mesh generation code.

The decomposition of the three-dimensional grid system can be performed in two different ways; divide the three-dimensional volume grid as "volume partitioning" or divide the two-dimensional surface mesh and subsequently create the strands within each subdomain as "surface partitioning". Figure 2.14 shows an example of the surface and volume partitioning of a strand grid around a sphere using METIS, with each color indicating a unique subdomain. For better visualization, only a few number of strand layers are used for the volume domain with relatively large wall-normal spacing. Table 2.1 shows the decomposed number of cells and nodes within each subdomain when partitioned either using the surface grid or the volume grid. The overall volume domain contains 2,160 cells and 2,420 nodes. The number of cells is almost evenly balanced using surface partitioning as well as volume partitioning. Although either division of the surface or volume domain will result in the Hamiltonian paths on a surface mesh being broken, surface division can keep each strand grid in a single domain. Therefore, surface division is used for a strand grid volume mesh, otherwise volume division is used for a typical unstructured volume mesh.



(a) Surface partitioning (b) Volume partitioning

Figure 2.14: Representative surface and volume partitioning of a sphere using METIS.

Table 2.1: Distribution of cells across different processors by either volume or surface subdivision.

| | Surface partition | | Volume partition | |
|---|---|---|---|---|
| Subdomain | Cells | Nodes | Cells | Nodes |
| 1 | 360 | 530 | 360 | 526 |
| 2 | 360 | 530 | 360 | 545 |
| 3 | 360 | 540 | 360 | 541 |
| 4 | 369 | 550 | 360 | 531 |
| 5 | 360 | 540 | 361 | 530 |
| 6 | 351 | 530 | 359 | 525 |

The surface domain partitioning can be performed either at the triangular cell stage or the quadrilateral cell stage after quadrilateral subdivision. In this study, the partitioning is performed at the quadrilateral cell stage. This is because more algorithmic work is required to identify neighbor cells between adjacent subdomains if partitioning is performed at the triangular cell stage. Once a domain is partitioned at the triangular cell stage, the subsequent processes, such as quadrilateral subdivision and path identification are performed in each processor, and it requires additional communications between subdomains to identify its neighboring of quadrilateral element. For the same reason, the partitioning for an unstructured volume mesh is also performed at the hexahedral cell stage after the subdivision process from the mixed tetrahedrons and prisms.

It should be noted that Hamiltonian paths on the partitioned surface domain can cross more than two subdomains as the whole domain is divided into more number of subdomains using METIS. As shown in Fig 2.15, an initial closed loop which consisted of 12 cell elements is divided into 4 smaller opened loops through domain partitioning. This will result in the minimum length of the loop, only consisting of 2 elements at second and third subdomains. However, when stencil-based reconstruction is used along the line structures, it usually requires a size of stencil more than 2 for third- or fifth-order schemes. To guarantee a large enough stencil size for the high-order reconstruction methods, a concept using ghost cells which is also known as dummy cells is adopted in both mesh generation and the flow solver. The ghost cells are additional layers of grid points outside the physical domain along the boundaries between subdomains. The ghost cells are only virtual, and geometrical

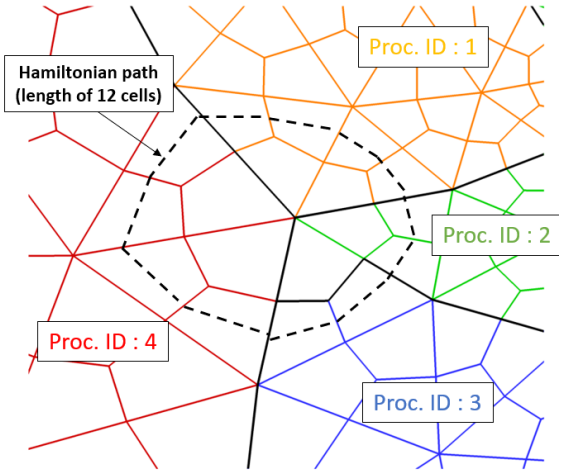



Figure 2.15: Hamiltonian path on partitioned unstructured surface mesh.

quantities like volume and face vectors of ghost cells are not required in the reconstruction methods. Note that the required layers of ghost cell vary depending on the reconstruction methods with a maximum of three layers used in the current method for fifth-order reconstruction. More explanation about reconstruction methods is shown in section 3.2.

## 2.4 Data Structure

Unlike with a classical unstructured grid, unique data structures along each line are provided to the flow solver for its line-based solution algorithm. The face of an element is the basic building block of the current data structure, and faces are arranged in an order along each Hamiltonian path or strand grid. Figure 2.16 shows an example of face indexing for the first two opened-loops on a two-dimensional domain. Between the paths, the face index can be increased without any duplication and the Hamiltonian path and strand grid can be represented as a chain of face

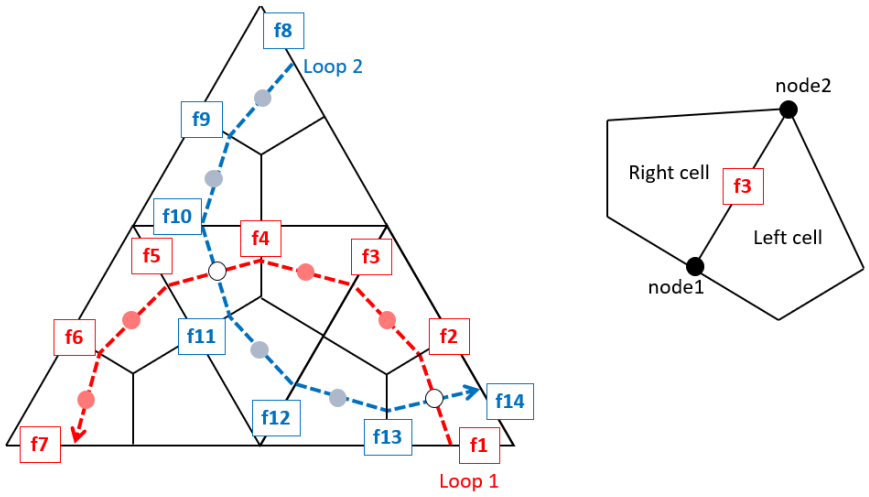



Figure 2.16: Chain-to-face connectivity (left) and required grid data for each face (right).

indices because each face is part of one unique Hamiltonian path.

Once chain-to-face connectivity is created, a group of grid data for each face is stored as an array. In two-dimension, the array has four integers per face. They are the indices of the two nodes that form the face and the indices of the left and right cells of the face. The array can be extended easily for a three-dimensional grid, with a total of six integers required per face by adding two more integers for the additional nodes that form the face. The array for a three-dimensional grid is shown in Eq. 2.1.

$$face[i] = [node1,\ node2,\ node3,\ node4,\ leftCell,\ rightCell] \tag{2.1}$$

For the communication between partitioned domains which run in parallel in the flow solver, the additional grid data structure is required. The array for the communication is made with $[nmpiface \times 13]$ size for each partitioned subdomain

(see Eq. 2.2). $nmpiface$ is the number of boundary faces for each domain along which the partitioned domains are connected to each other. At each boundary face, a total of 6 neighbor cell indices are stored with corresponding processor ID ($proc$): each side of the face has three cells and corresponding processor IDs, respectively. It should be noted that at each side of the face, different subdomains can occur within the successive three cells, thus processor ID information is required for each cell.

$$\begin{aligned} mpiface[i] = [face,\ cell1,\ proc1,\ cell2,\ proc2,\ cell3,\ proc3, \\ cell4,\ proc4,\ cell5,\ proc5,\ cell6,\ proc6] \end{aligned} \tag{2.2}$$

In addition to these arrays, the number of total paths, starting face index for each path, and chain-to-face connectivity are provided to the flow solver. Table 2.2 summarizes the current grid data structures utilized for organizing the data.

## 2.5 Summary

In this chapter, the computational methodology of the mesh generation is presented. First, the generation of Hamiltonian paths has been extended to general unstructured surface grids with both quadrilateral and triangular elements. From an initial mix of elements, each element is subdivided into quadrilaterals to obtain pure quadrilateral mesh. The process of Hamiltonian path identification is also improved to be applied robustly and uniquely in any pure quadrilateral/hexahedral mesh including typical structured grids. Self-crossed paths can occur when paths are identifed on a semi-regular type grid.

For three-dimensions, two different types of volume grid have been employed:

Table 2.2: Grid data structures.

| variables | description |
|---|---|
| nnodes | total number of nodes |
| ncells | total number of quadrilateral cells |
| nfaces | total number of faces |
| nchians | total number of chains |
| faceStartPerChain | starting face index for each path |
| chainConn | chain-to-face connectivity |
| face | list of faces (6 integers per face in three-dimension) |
| mpiface | lift of cells and processor ID (12 integers per mpiface) |
| coord | node coordinates |
| conn | cell-to-node connectivity |

strand based grids and traditional unstructured volume grids. The strand grids are generated from the surface grid by extruding in a wall-normal direction. Therefore, third "out-of-plane" spatial direction is easily identified along each strand. To prevent strands from crossing over each other at concave corner, the strand grids are allowed to be curvilinear which is generated using a advancing front-like technique.

The generation of Hamiltonian path is applicable to an initial mix of unstructured volume elements of tetrahedron and prism. In order to do this, each element is subdivided into multiple hexahedrons as a prerepuisite. Then, the Hamiltonian paths are constructured on the divided volume domain using the same algorithm as

the one used for the surface mesh.

The overall grid is split into multiple subdomains using METIS in order to be computed in parallel. The decomposition of the three-dimensional grid is performed using "surface partitioning" or "volume partitioning" for strand grids and unstructured volume grids, respectively. To guarantee a large enough stencil size for stencil-based reconstructions, a concept of ghost cell with multiple layers is applied along the boundary between subdomains.

# Chapter 3: Computational Methodology of Flow Solver

In this chapter, the fundamental fluid dynamics equations along with the numerical solution algorithms are described. The solution algorithms are focused on the use of Hamiltonian paths and strand grids in the evaluation of numerical fluxes and the application of various line-implicit methods to current mesh system. Following this, turbulence and transition modeling are described briefly. Finally, the integration of current flow solver into the overset framework using multi-mesh/multi-solver paradigm will be discussed.

## 3.1 Governing Equations

The three-dimensional, unsteady, Navier–Stokes equations describe the behavior of fluid flow. In this work, they are used to represent compressible, non-reacting, ideal gas flow across the boundary of a closed domain known as a control volume. They ensure universal laws of conservation of mass, momentum, and energy in the control volume. The integral form of the system of equation in conservation form is given by Eq. 3.1.

$$\frac{\partial}{\partial t}\int_{\mathrm{V}} Q dV + \oint_{\partial \mathrm{V}} F(Q)\cdot \vec{n} ds = \oint_{\partial \mathrm{V}} G(Q)\cdot \vec{n} ds + \int_{\mathrm{V}} S(Q) dV \tag{3.1}$$

where $V$ is control volume and $\vec{n}$ is the unit normal vector to the face. The vector of conserved variables is represented as $Q$, and $F(Q)$ and $G(Q)$ correspond to the convective and viscous fluxes, respectively. $S(Q)$ represents the source terms that have to be included to account for the centrifugal and Coriolis accelerations if the equations are formulated in a non-inertial frame of reference. In the current work, the equations are formulated in an inertial frame of reference for all of the simulations with rotating motion. Therefore, the source terms are zero. The vector of conserved variables is given by:

$$Q = [\rho,\ \rho u,\ \rho v,\ \rho w,\ E]^T \tag{3.2}$$

where $\rho$ is the fluid density, and $u$, $v$, $w$ are components of the fluid velocity along the Cartesian coordinate system $(x,\ y,\ z)$. $E$ is the total energy per unit volume given by:

$$E = \rho \left[ e + \frac{1}{2}(u^2 + v^2 + w^2) \right] \tag{3.3}$$

where, $e$ is the internal energy per unit mass. The inviscid and viscous flux vectors are given by Eqs. 3.4, 3.5.

$$F(Q)\cdot\vec{n} = ((\vec{V}-\vec{V}_g)\cdot\vec{n})\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{pmatrix} + p\begin{pmatrix}0\\ n_x\\ n_y\\ n_z\\ \vec{V}\cdot\vec{n}\end{pmatrix} \tag{3.4}$$

$$\begin{aligned} G(Q)\cdot\vec{n} &= n_x G_1 + n_y G_2 + n_z G_3 \\ G_i &= [0,\ \tau_{i1},\ \tau_{i2},\ \tau_{i3},\ u_j\tau_{ij} - q_i]^T \end{aligned} \tag{3.5}$$

where, $\vec{V}$ is the velocity vector and $\vec{V}_g$ is the grid velocity vector. $p$ is pressure and $q$ is heat conduction term expressed as a function of temperature ($T$) and coefficient of thermal conductivity ($k$) according to Fourier's law.

$$q_i = -\frac{1}{(\gamma-1)}\left(\frac{\mu}{P_r} + \frac{\mu_t}{P_{r_t}}\right)\frac{\partial T}{\partial x_i} \tag{3.6}$$

where, $P_r$ is Prandtl's number for laminar flow as 0.72 and $P_{r_t}$ is for turbulent flow as 0.9 for air. $\mu$ is the molecular viscosity and $\mu_t$ is the eddy viscosity which is determined by a turbulence model. The value of the ratio of specific heats $\gamma$ is 1.4 for air at standard temperature and pressure.

The viscous stress tensor for Newtonian fluids, $\tau_{ij}$, formulated using Stokes' hypothesis is given by:

$$\tau_{ij} = (\mu+\mu_t)\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right] \tag{3.7}$$

The coefficient of molecular viscosity is given by Sutherland's formula.

$$\mu = C_1 \frac{T^{\frac{3}{2}}}{T + C_2} \tag{3.8}$$

where, $C_1 = 1.4 \times 10^{-6}$ kg/(ms$\sqrt{K}$) and $C_2 = 110.4$ K for air at standard temperature and pressure.

To close the system of equations, the *equation of state* for ideal gases is used:

$$p = \rho R T \tag{3.9}$$

where, $R$ is the gas constant. Since all the flows studied in this work involve air at standard temperature and pressure, the calorically perfect gas assumption is valid. A calorically perfect gas is an ideal gas with constant specific heats. Specific heat at constant volume $c_v$ and specific heat at constant pressure $c_p$ are given by:

$$c_v = \frac{R}{\gamma - 1} \qquad c_p = \frac{\gamma R}{\gamma - 1} \tag{3.10}$$

The following relations between thermodynamic quantities are applicable to calorically perfect gases:

$$e = c_v T \tag{3.11}$$

$$p = (\gamma - 1)\rho e \tag{3.12}$$

The total energy per unit volume, $E$, can now be re-written in terms of pressure and velocity components as follows:

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \left(u^2 + v^2 + w^2\right) \tag{3.13}$$

When using mesh deformation techniques within a finite volume method, Geometric Conservation Law (GCL) much be satisfied to ensure free-stream preserva-

tion. In this study, the contravariant grid velocity ($\vec{V}_g \cdot \vec{n}$) of the $i$-th face is computed by calculating sweeping volume of the face.

$$\left(\vec{V}_g \cdot \vec{n}\right)_i^{n+1} = \frac{\Delta V_i^n}{s_i^{n+1} \Delta t} \tag{3.14}$$

where $\Delta V_i^n$ represents the volume sweep by the $i$-th face between the time layer $n$ and $n+1$. $s_i$ represents the area of the $i$-th face.

The validation of GCL is performed through a pitching airfoil simulation in Appendix B.

### 3.1.1 Non-Dimensional Form of Equations

The governing equations are often solved in their non-dimensional form. There are two advantages of doing this: (1) parameters such as Mach number and Reynolds number can be varied independently, (2) all flow variables are normalized to fall in the vicinity of (0,1), thereby reducing numerical inaccuracies that may occur due to mathematical operations between largely different values.

$$x^* = \frac{x}{L}, \quad y^* = \frac{y}{L}, \quad z^* = \frac{z}{L}, \quad t^* = \frac{t}{L/a_\infty} \tag{3.15}$$

$$u^* = \frac{u}{a_\infty}, \quad v^* = \frac{v}{a_\infty}, \quad w^* = \frac{w}{a_\infty}, \quad \mu^* = \frac{\mu}{\mu_\infty} \tag{3.16}$$

$$\rho^* = \frac{\rho}{\rho_\infty}, \quad p^* = \frac{p}{\rho_\infty a_\infty^2}, \quad T^* = \frac{T}{T_\infty} \tag{3.17}$$

where the nondimensional variables are denoted by an asterisk, free stream conditions are denoted by $\infty$, and $L$ is the reference length used in the Reynolds number:

$$Re_L = \frac{\rho_\infty V_\infty L}{\mu_\infty} \tag{3.18}$$

Substituting the above relations into the governing equations in Eq. 3.1 gives a new set of equations in terms of the non-dimensional variables.

### 3.1.2 Reynolds-Averaged Navier–Stokes Equations

The Reynolds-Averaged Navier–Stokes (RANS) equations is an affordable alternative to Direct Numerical Simulation (DNS) or Large-eddy simulation (LES) for turbulent flow simulation in engineering applications. Therefore, the RANS approach is used in this study. In the RANS approach, dependent variables in the baseline governing equations shown in Eq. 3.1 are decomposed into their mean and fluctuating components and the resulting equations are averaged over an appropriate time interval.

In the Reynolds-averaging procedure, the mean or time-averaged quantity $\overline{f}$ is defined as:

$$\overline{f} \equiv \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} f dt \tag{3.19}$$

where $\Delta t$ is large compared to the period of the random fluctuations associated with the turbulence, but small compared to the time scales of mean flow variation in unsteady flows. By definition, the time-averaged value of a fluctuating quantity is zero:

$$\overline{f'} \equiv \frac{1}{\Delta t}\int_{t_0}^{t_0+\Delta t} f' dt = 0 \tag{3.20}$$

The following relations hold for sum and product of any two fluctuating quantities:

$$\overline{\overline{f}g'} = 0 \qquad \overline{\overline{f}g} = \overline{f}\,\overline{g} \qquad \overline{f+g} = \overline{f} + \overline{g} \tag{3.21}$$

The most important identity is that the time-average of the product of two fluctuating quantities is not zero:

$$\overline{f'f'} \neq 0 \qquad\qquad \overline{f'g'} \neq 0 \tag{3.22}$$

In the Reynolds decomposition approach, the randomly changing flow variables are replaced by time averages plus fluctuations as shown below:

$$u = \overline{u} + u' \quad v = \overline{v} + v' \quad w = \overline{w} + w' \quad \rho = \overline{\rho} + \rho' \quad p = \overline{p} + p' \tag{3.23}$$

The turbulence field is said to be isotropic when $u' = v' = w'$. Turbulent intensity of turbulence level ($Tu$) is defined as the ratio of the root-mean-square of turbulent velocity fluctuations ($U'$) and the mean velocity magnitude ($U$):

$$Tu = \frac{U'}{U} \tag{3.24}$$

$$U' = \sqrt{\frac{1}{3}[(u')^2 + (v')^2 + (w')^2]} \qquad U = \sqrt{(\overline{u})^2 + (\overline{v})^2 + (\overline{w})^2} \tag{3.25}$$

Substitution of the Reynolds-decomposed dependent variables in Eq. 3.23 into the instantaneous, unsteady Navier-Stokes equations in Eq. 3.1, followed by time-averaging of the equations gives rise to a new set of governing equations. These are known as the Reynolds-Averaged Navier-Stokes equations. These are almost identical in form to the unsteady Navier-Stokes equations except that the time-averaged

quantities $\overline{f}$ have replaced the instantaneous quantities $f$. The additional term is also derived which behaves as an apparent stress tensor due to the transport of momentum by turbulent fluctuations. Hence it is commonly known as the Reynolds Stress Tensor $\rho\overline{u_i'u_j'}$, given by:

$$(\overline{\tau}_{ij})_{turb} = -\rho\overline{u_i'u_j'} \tag{3.26}$$

With the introduction of Reynolds-stress terms, six additional unknowns are obtained in the Reynolds-averaged momentum equations. In order to close the RANS equation, the Reynolds-stress term is approximated using a turbulence model. Details of turbulence modeling will be discussed in Section 3.4.

## 3.2 Evaluation of fluxes

### 3.2.1 Finite Volume Method

The governing equations are discretized by using a cell-centered finite-volume method. The flow domain is divided into a finite number of control volumes. Each one of the arbitrary volumes is $V_i$ and closed by a boundary $\partial V_i$. Since each control volume $V_i$ shares common boundaries with its neighbors, this approach retains the conservative property inherent to the integral equations. This feature ensures that the contributions from the fluxes across all of the interior boundaries within the global domain will exactly cancel each other. The control volume $V_i$ can be represented by any arbitrary cell elements, i.e. hexahedron, tetrahedron, etc.

The cell-centered approach in the current study contrasts with the more com-

monly used node-centered approach for an unstructured grid based solver. The primary reason for choosing the cell-centered approach is to use a line-based method on the unstructured grid along the Hamiltonian paths and strand grid. A node-centered approach uses dual control volumes which are created by connecting midpoints of the cells having the respective node in common. In this case, it cannot guarantee that the dual control volume has an even number of mutual faces, which is a prerequisite for Hamiltonian path identification. Also, the cell-centered approach is more straightforward to program because the control volumes are identical with the grid cells.

The conserved variables $Q$ are the volume-averaged values in the finite volume method and defined by:

$$\overline{Q}_i = \frac{1}{V_i} \int_{\mathrm{V}} Q dV \tag{3.27}$$

By using a semi-discrete approximation, the unsteady Navier-Stokes equations in Eq. 3.1 can be written as:

$$\frac{\partial (Q_i V_i)}{\partial t} + \sum_{j \in \text{ nfaces}} F(Q)_{i,j} s_{i,j} = \sum_{j \in \text{ nfaces}} G(Q)_{i,j} s_{i,j} \tag{3.28}$$

### 3.2.2 Inviscid Fluxes

The spatial discretization and time integration procedures are fully decoupled in the finite volume approach. Evaluation of inviscid fluxes involves two steps: (1) reconstruction of the conservative variables at cell faces, and (2) evaluation of the fluxes at interfaces using reconstructed conservative variables. Reconstruction schemes for systems with hyperbolic properties are often based on some form of

upwinding. This ensures that the numerical scheme respects the direction of wave propagation and uses information only from the upstream direction.

In this study, both stencil-based and gradient-based reconstructions are used. It should be noted that stencil-based reconstructions cannot be used on a general unstructured grid. However, the current method can use it on both structured and unstructured grids along the paths and strand grids.

## Stencil-Based Reconstruction

Figure 3.1 shows a schematic of loops passing through a quadrilateral cell along with the associated cell and face numbers. The quadrilateral cells can be obtained from either a structured grid or an unstructured grid with subdivided triangle elements. The reconstruction of left states ($q^L$) and right states($q^R$) are accomplished along the cells of a given loop from the cell-averaged values. Because each of the paths is treated like a coordinate direction of a structured grid, any stencil-based reconstruction methods used in a structured grid based solver can be applied along each of the paths.

In the current study, the third-order Monotone Upstream-Centered Scheme for Conservation Laws (MUSCL) [3] and fifth-order Weighted Essentially Non-Oscillatory (WENO) scheme [4] are used for reconstruction. With a MUSCL scheme, the reconstructed conservative variables at each interface is expressed as a function of the cell-averaged values with a three-point stencil:

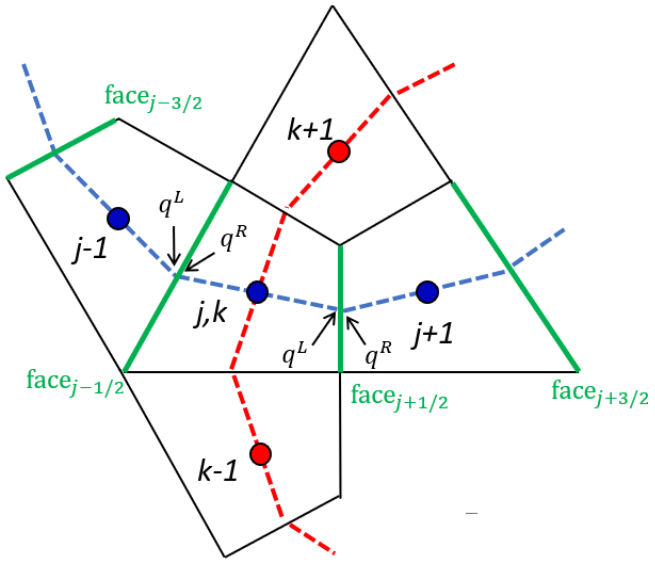



Figure 3.1: Schematic showing the numbering scheme along Hamiltonian loops used for reconstruction strategy.

$$
\begin{aligned}
q^L_{i+\frac{1}{2}} &= \overline{q}_i + \phi_i \left[ \frac{1}{3} \left( \overline{q}_{i+1} - \overline{q}_i \right) + \frac{1}{6} \left( \overline{q}_i - \overline{q}_{i-1} \right) \right] \\
q^R_{i-\frac{1}{2}} &= \overline{q}_i - \phi_i \left[ \frac{1}{3} \left( \overline{q}_{i+1} - \overline{q}_i \right) + \frac{1}{6} \left( \overline{q}_i - \overline{q}_{i-1} \right) \right]
\end{aligned}
\tag{3.29}
$$

where $\phi$ is the Koren's differentiable limiter given by.

$$
\phi_i = \frac{3 \Delta \overline{q}_i \nabla \overline{q}_i + \epsilon}{2 \left( \Delta \overline{q}_i - \nabla \overline{q}_i \right)^2 + 3 \Delta \overline{q}_i \nabla \overline{q}_i + \epsilon}
\tag{3.30}
$$

where $\epsilon$ is a small number used to prevent division by zero, and $\Delta$ and $\nabla$ are the forward and backward difference operators defined by $\Delta \overline{q}_i = \left( \overline{q}_{i+1} - \overline{q}_i \right)$ and $\nabla \overline{q}_i = \left( \overline{q}_i - \overline{q}_{i-1} \right)$.

The fifth-order finite difference WENO scheme is briefly explained in Eq. 3.31. The same weights, $\omega_1$, $\omega_2$ and $\omega_3$, are used as proposed by Jiang and Shu [4].

$$
\begin{aligned}
q^L_{i+\frac{1}{2}} &= \frac{\omega_1}{3}\overline{q}_{i-2} - \frac{1}{6}(7\omega_1+\omega_2)\overline{q}_{i-1} + \frac{1}{6}(11\omega_1+5\omega_2+2\omega_3)\overline{q}_i \\
&\quad + \frac{1}{6}(2\omega_2+5\omega_3)\overline{q}_{i+1} - \frac{\omega_3}{6}\overline{q}_{i+2} \\
q^R_{i-\frac{1}{2}} &= \frac{\omega_1}{3}\overline{q}_{i+2} + \frac{1}{6}(7\omega_1+\omega_2)\overline{q}_{i+1} + \frac{1}{6}(11\omega_1+5\omega_2+2\omega_3)\overline{q}_i \\
&\quad + \frac{1}{6}(2\omega_2+5\omega_3)\overline{q}_{i-1} - \frac{\omega_3}{6}\overline{q}_{i-2}
\end{aligned}
\tag{3.31}
$$

## Gradient-Based Reconstruction

The gradient-based reconstruction is also applied for inviscid flux evaluation in addition to stencil-based reconstructions. It is assumed that the solution is piecewise linearly distributed over the control volume. Then, the left and right state for a cell-centered scheme are estimated from Eq. 3.32.

$$
\begin{aligned}
q^L_{i+\frac{1}{2}} &= \overline{q}_i + \phi_i(\nabla q_i \cdot \vec{r}_L) \\
q^R_{i-\frac{1}{2}} &= \overline{q}_i + \phi_i(\nabla q_i \cdot \vec{r}_R)
\end{aligned}
\tag{3.32}
$$

where $\nabla q_i$ is the gradient of $q$ ($\left[\frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}, \frac{\partial q}{\partial z}\right]^T$) at the cell center $i$ and $\phi$ denotes a multi-dimensional limiter function [2]. The vectors $\vec{r}_L$ and $\vec{r}_R$ point from the cell-centroid to each face-midpoint.

To compute the solution gradient ($\nabla q_i$), two approaches are usually used on an unstructured grid: Green-Gauss and least-squares. In this study, least-squares approach is applied because the least-squares gradient estimation provides more reliable results than Green-Gauss on poor quality meshes [21]. The linear least-squares (LLS) approach is based upon the use of a first-order Taylor series approximation

from each cell-centroid. The gradient can be computed by a summation of field variables at neighboring cell-centroids ($q_j$) and their corresponding weights ($\beta_j$) as shown in Eq. 3.33.

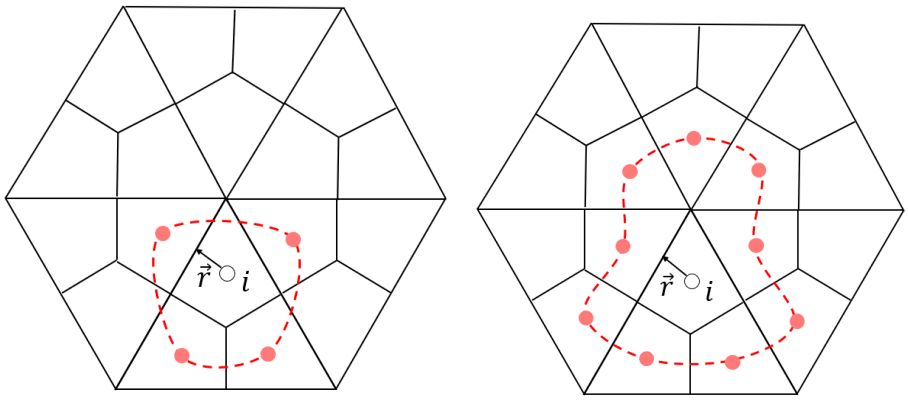
$$\nabla q_i = \sum_{j=1}^{N} \beta_j q_j \tag{3.33}$$

where $N$ is the number of neighbor cells of the cell $i$.

To find weights of each neighbor, a system of equations is solved as shown in Eq. 3.34. The dimension of the matrix, $(x_j - x_i)^T$, is $N$ by 2 for two-dimensions ($N$ by 3 for three-dimensions). The dimension of vector, $C$, is $N$ by 1 with all elements of 1. More details of the least-squares procedure may be found in the reference [22].

$$\begin{gathered} q_j = q_i + \nabla q_i (x_j - x_i)^T \\ \left[ (x_j - x_i)^T ,\ C \right] \left[ \nabla q_i,\ q_i \right]^T = q_j \end{gathered} \tag{3.34}$$

The number of neighbors, $N$, can vary for each cell on an unstructured grid system. Figure. 3.2 shows two different sets of neighbor cell stencils for the linear least-squares method; standard stencil and wrapping stencil. For standard second-order, the computational stencil composed of face-neighbors is a natural choice. However, a wrapping stencil composed of not only the face-neighbors but also node-neighbors was also considered as proposed by [23]. This is because the wrapping stencil provides a better reconstruction and ensures an adequate number of neighbor cells even along the boundary.




(a) Standard stencil (b) Wrapping stencil

Figure 3.2: Gradient based reconstruction stencil.

## Flux-Difference Splitting Schemes

The reconstructed left and right states are used to compute inviscid fluxes at each interface using Roe's flux difference splitting scheme with an entropy fix, as shown below:

$$F\left(q^L,\, q^R\right)_{=} \frac{1}{2}\left\{\left(F(q^L) + F(q^R)\right) - |\mathbf{A}_{\text{Roe}}(q^L,\, q^R)|(q^R - q^L)\right\} \tag{3.35}$$

where, $F(q^L)$ and $F(q^R)$ are the left and right state fluxes, and $\mathbf{A}_{\text{Roe}}$ is the Roe-averaged Jacobian matrix. The second term on the right-hand side of the above equation represents numerical dissipation. The dissipation term is scaled by normal velocity to prevent excessive dissipation in low Mach number flow condition using the all-Mach correction proposed by Rieper [24]. The entropy correction of Harten [25] is used to modify the eigenvalues of the flux Jacobian to prevent non-physical phenomena at a stagnation point or sonic point.

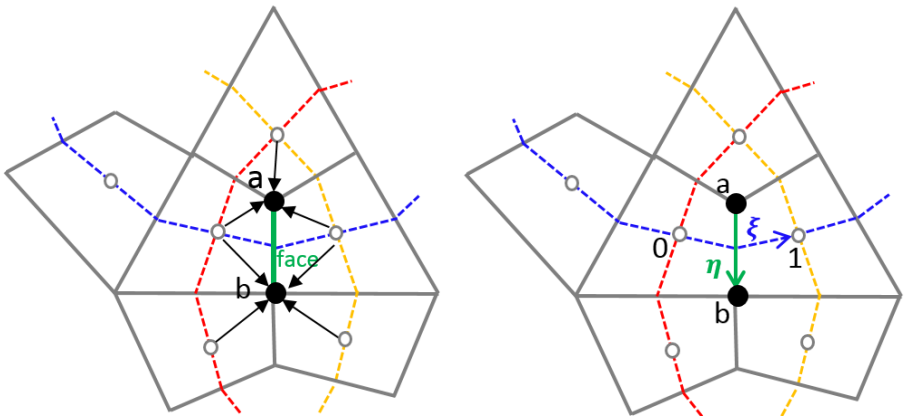
### 3.2.3 Viscous Fluxes

The viscous fluxes are calculated using velocity and temperature gradients obtained from either a finite central difference scheme or linear least-squares, which are both second-order accurate. Figure 3.3 explains the schematic procedure of the contribution of the cells toward the streamwise and cross term evaluations of the gradients using a finite difference method. Using two values at cell centered (0,1) and two values at nodes (a,b), Eq 3.36 shows the gradient computation in two-dimensions at each interface. $\phi$ is the velocity component or temperature and $\xi$, $\eta$ are the spatial directions in two-dimensions.

$$\begin{aligned} \frac{\partial \phi}{\partial x} &= \xi_x \frac{\partial \phi}{\partial \xi} + \eta_x \frac{\partial \phi}{\partial \eta} \\ \frac{\partial \phi}{\partial y} &= \xi_y \frac{\partial \phi}{\partial \xi} + \eta_y \frac{\partial \phi}{\partial \eta} \end{aligned} \tag{3.36}$$

where $\phi_\xi = \phi_1 - \phi_0$, $x_\xi = x_1 - x_0$, $y_\xi = y_1 - y_0$, $\phi_\eta = \phi_b - \phi_a$, $x_\eta = x_b - x_a$, $y_\eta = y_b - y_a$.

Similarly, the gradient at a mid-point edge can be computed using a linear least-squares approach. Compared to the finite different method, the least-squares approach can compute the gradient using field values only at cell-centers, thus node value evaluation is not required in the least-squares approach. For the gradient at mid-point edge the wrapping stencil is used as shown in Fig. 3.4.




(a) Node value evaluation

(b) Stream-wise and cross-term evaluation of flux

Figure 3.3: Schematic showing the finite different method for gradient evaluation.

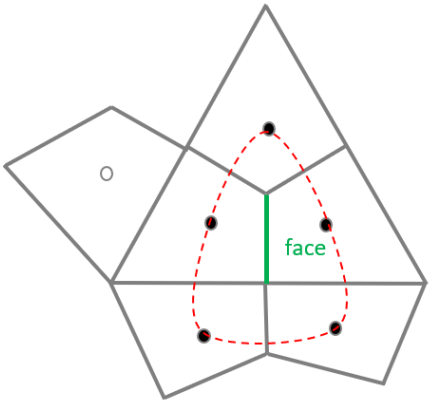



Figure 3.4: Wrapping stencil of least-squares approach for gradient estimation at mid-point edge.

### 3.2.4 Initial Conditions

To perform the time integration of the Navier–Stokes equations, an initial flow field consisting of the primitive variables ($\rho$, $u$, $v$, $w$, $p$) must be specified at each grid point. In simulations of external aerodynamic flows, the entire flow field is typically initialized with the freestream values.

### 3.2.5 Boundary Conditions

#### Wall Boundary

In the case of an inviscid wall, the fluid slips over the surface. The velocity vector must be tangent to the surface. This is equivalent to the condition that there is no flow normal to the surface.

$$\vec{V} \cdot \vec{n} = 0 \tag{3.37}$$

where $\vec{V}$ is the local flow velocity vector and $\vec{n}$ denotes the unit normal vector at the surface. Within the cell-centered scheme, multiple layers of ghost cells are employed at the wall boundary (e.g. MUSCL uses two layers of ghost cells and WENO uses three layers of ghost cells).

The velocity components in the ghost cells are obtained by reflecting the velocity vectors in the boundary cells at the wall (See Fig. 3.5(a)).

$$\begin{aligned} \vec{V}_{-1} &= \vec{V}_1 - 2V_1\vec{n} \\ \vec{V}_{-2} &= \vec{V}_2 - 2V_2\vec{n} \end{aligned} \tag{3.38}$$

where $V_1 = u_1 n_x + v_1 n_y + w_1 n_z$ is the contravariant velocity and $\vec{n}$ stands for the wall unit-normal vector. The pressure and density in the ghost cells are set equal to the values in the corresponding boundary cells.

For a viscous wall, the relative velocity between the surface and the fluid at the surface is assumed to be zero (no-slip boundary condition). Within the cell-centered scheme which utilizes ghost cells, an adiabatic wall boundary condition is

set as shown in the equation below (see Fig. 3.5(b)).

$$\rho_{-1} = \rho_1, \quad E_{-1} = E_1, \quad u_{-1} = -u_1, \quad v_{-1} = -v_1, \quad w_{-1} = -w_1 \tag{3.39}$$

Since the pressure gradient normal to the wall is zero, the pressure in the boundary is prescribed in the ghost cells ($p_{-1} = p_1$).

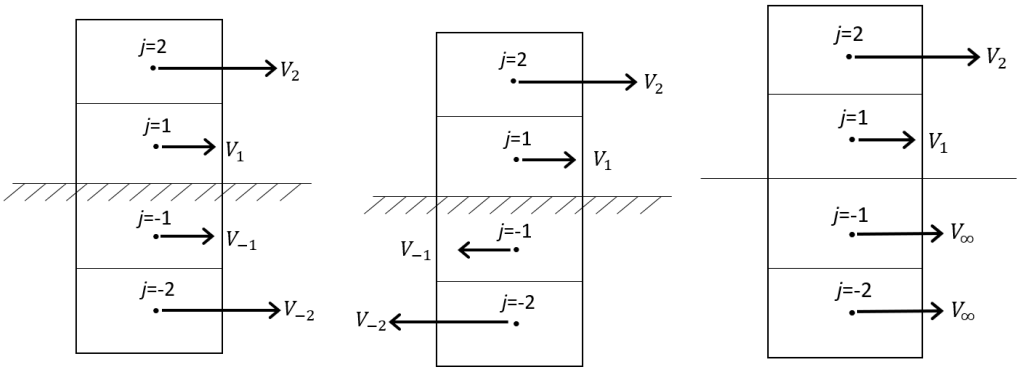
### Far-field Boundary

In external flow simulations, the computational domain is truncated to a finite size, thus creating an artificial boundary. In this study, a Dirichlet boundary condition using freestream flow conditions is imposed on far-field boundaries of the domain (see Fig. 3.5(c)). Therefore, meshes for external flow problems are typically generated with the far-field boundary located at large distances from regions of flow activity, such as flow past a solid body. Mesh stretching is also employed towards the far-field boundary to numerically dissipate strong flow gradients.

## 3.3 Time Integration

### 3.3.1 Implicit Operator

The simplest methods of time integrating the semi-discrete form of the unsteady Navier-Stokes equations (Eq. 3.28) is the use of the explicit schemes. In the explicit schemes, the time derivative is discretized using a finite difference formula at time step $n$, and the residual $R$ is evaluated at the same time step $n$. Therefore,




(a) Inviscid wall/ symmetric B.C.

(b) Viscous wall B.C.

(c) Far-field B.C.

Figure 3.5: Schematic showing ghost cell set up for solid wall and free-stream boundary conditions.

unknown quantities in the difference equation can be directly evaluated in terms of known quantities. For example, forward differences in time can be written as

$$\frac{(QV)^{n+1} - (QV)^n}{\Delta t} = -R(Q^n) \tag{3.40}$$

The primary disadvantage of the explicit method is a restriction on the time step size for the stability of difference schemes. Courant, Friedrichs, and Lewy (CFL) convergence condition for hyperbolic equations limit the time step. An implicit scheme requires the simultaneous solution of a system of equations involving all of the unknowns at the new time level, $n+1$ as shown in Eq 3.41. The resulting non-linear equation obtained using the backward difference formulations, first-order (BDF1) or second-order (BDF2) methods are shown below:

$$\frac{\partial (QV)^{n+1}}{\partial t} = -R(Q^{n+1}) \tag{3.41}$$

where BDF1 method is shown below,

$$\frac{\partial(QV)^{n+1}}{\partial t} = \frac{(QV)^{n+1} - (QV)^{n}}{\Delta t} \tag{3.42}$$

where BDF2 method is shown below,

$$\frac{\partial(QV)^{n+1}}{\partial t} = \frac{3(QV)^{n+1} - 4(QV)^{n} + (QV)^{n-1}}{\Delta t} \tag{3.43}$$

On the right-hand side of Eq. 3.41, $R(Q^{n+1})$ is the solution residual at time level $n+1$, which is computed as below,

$$R(Q^{n+1}) = \sum_{i \in \text{nfaces}} F_i(Q^{n+1}) \cdot ds_i \tag{3.44}$$

The above non-linear equation is linearized in time using a Taylor series expansion about $Q^n$ as follows:

$$F^{n+1} = F^n + \mathbf{A}\Delta Q^n + O(\Delta t^2) \tag{3.45}$$

where term $\mathbf{A}$ represents the Jacobian matrices of the flux vectors with respect to the conservative variables, $\frac{\partial F}{\partial Q}$.

Linearizing the nonlinear residual function $R^{n+1}$ yields a linear algebraic system of equations as shown in Eq. 3.46, which is solved using an iterative technique.

$$\left[\frac{I}{\Delta t} + \left(\frac{\partial R}{\partial Q}\right)^{n}\right] \Delta Q^n = -R(Q^n) \tag{3.46}$$

In this study, the implicit operator is constructed based on a first-order linearization of the nonlinear residual function, and therefore, entails contributions

only from the neighboring cells. With the computational coordinates, the implicit operator can be written as

$$\left[\frac{I}{\Delta t}+\frac{\partial R}{\partial Q}\right]=\left[\frac{I}{\Delta t}+\frac{\partial R^{\zeta}}{\partial Q}+\frac{\partial R^{\eta}}{\partial Q}+\frac{\partial R^{\xi}}{\partial Q}\right] \tag{3.47}$$

where the operators are given by:

$$\begin{aligned}
\frac{\partial R^{x}}{\partial Q} &= [\mathbf{A}_x]\Delta Q_{x-1}+[\mathbf{B}_x]\Delta Q_{\mathrm{jkl}}+[\mathbf{C}_x]\Delta Q_{x+1} \\
\mathbf{A}_x &= \frac{\partial F_x}{\partial Q_R} \\
\mathbf{B}_x &= \frac{\partial F_x}{\partial Q_L}-\frac{\partial F_{x+1}}{\partial Q_R} \\
\mathbf{C}_x &= -\frac{\partial F_{x+1}}{\partial Q_L}
\end{aligned} \tag{3.48}$$

where $x$ denotes each of the three spatial directions, $\zeta$, $\eta$ and $\xi$, and $j$, $k$, and $l$ are their indices, respectively. The three spatial directions directly correspond to the Hamiltonian paths and strand grids passing through it. Terms $\partial F/\partial Q_L$ and $\partial F/Q_R$ are the linearizations of the Roe flux function with respect to the left and right states, respectively.

### 3.3.2 Approximate Factorization

The implicit operator shown in Eq. 3.47 can be factorized in the coordinate direction given by Hamiltonian paths. Three different line-implicit methods are applied in the current study: ADI, DDADI and DDLGS. Details of each method can be found in Ref [26]. Unlike with grid coordinate directions in a structured grid, the coordinate direction along Hamiltonian paths is a local direction at each element. This means spatial direction, $\zeta$ of cell $i$ does not necessarily need to be the

same direction as for neighbor cell $j$. Because each face is part of a unique path, the factorized implicit operator can be solved for the whole domain by sweeping each loop from first to last cell regardless of spatial direction.

## Alternating Direction Implicit (ADI)

In the case of ADI, the implicit operator in Eq. 3.47 is simply factorized for each direction given by the Hamiltonian paths and strand grid lines. The factorized system can be expressed as:

$$\left[\frac{I}{\Delta t}+\frac{\partial R^{\zeta}}{\partial Q}\right]\left[\frac{I}{\Delta t}+\frac{\partial R^{\eta}}{\partial Q}\right]\left[\frac{I}{\Delta t}+\frac{\partial R^{\xi}}{\partial Q}\right]\Delta Q=-R\left(Q^{n}\right) \tag{3.49}$$

## Diagonally Dominant ADI (DDADI)

In the case of DDADI, the factorized system is written such that there is a more diagonally dominant term that aids in solution convergence and adds to the numerical stability of the scheme when compared to the ADI scheme. The DDADI factorized scheme is expressed as:

$$\left(\mathbf{D}+\mathbf{O}_{\zeta}\right)\mathbf{D}^{-1}\left(\mathbf{D}+\mathbf{O}_{\eta}\right)\mathbf{D}^{-1}\left(\mathbf{D}+\mathbf{O}_{\xi}\right)=-R\left(Q_{n}\right) \tag{3.50}$$

where the diagonal term $\mathbf{D}$ and the $\mathbf{O}$ matrices are given by

$$\mathbf{D} \;=\; \left[\frac{I}{\Delta t}+\left[\mathbf{B}_{j}\right]+\left[\mathbf{B}_{k}\right]+\left[\mathbf{B}_{l}\right]\right] \tag{3.51}$$

$$\mathbf{O}_\zeta = ([\mathbf{A}_j], 0, [\mathbf{C}_j]) \qquad \mathbf{O}_\eta = ([\mathbf{A}_k], 0, [\mathbf{C}_k]) \qquad \mathbf{O}_\xi = ([\mathbf{A}_l], 0, [\mathbf{C}_l]) \tag{3.52}$$

## Diagonally Dominant Line Gauss Seidel (DDLGS)

In the case of DDLGS, the factorization is not performed and a Gauss-Seidel is performed on a line basis, i.e. implicit inversion is performed for each line and the changes in the conservative variables ($\Delta Q$) are updated to the right-hand-side of subsequent lines as soon as a new update is available. The DDLGS method is given by:

$$\begin{aligned}[\mathbf{A}_j]\,\Delta Q_j + [\mathbf{D}]\Delta Q_{\mathrm{jkl}} + [\mathbf{C}_j]\Delta Q_{\mathrm{j+1}} = -R(Q^n) - [\mathbf{A}_k]\Delta Q^*_{\mathrm{k-1}} - [\mathbf{C}_k]\Delta Q^*_{\mathrm{k+1}} \\ -[\mathbf{A}_l]\Delta Q^*_{\mathrm{l-1}} - [\mathbf{C}_l]\Delta Q^*_{\mathrm{l+1}}\end{aligned} \tag{3.53}$$

where the diagonal term $\mathbf{D}$ is given by

$$\mathbf{D} \;=\; \left[\frac{I}{\Delta t} + [\mathbf{B}_j] + [\mathbf{B}_k] + [\mathbf{B}_l]\right] \tag{3.54}$$

The values with the asterisk (*) indicate intermediate values obtained during the Gauss-Seidel sweep. To eliminate the sweep bias, symmetry is obtained by sweeping once in a prescribed direction through the Hamiltonian loops and strand grid and then sweeping again in the reverse direction.

Using one of the above line-implicit methods, the left hand side of an implicit operator becomes a block-tridiagonal system for each path, and it can be either a periodic system for the closed loop or a non-periodic system for the open loop. Then, each block-tridiagonal system is inverted directly using a modified Thomas algorithm.

For the comparison with DDLGS method, a point Gauss-Seidel (PGS) method is implemented in the current flow solver, which is a well-known implicit method typical for an unstructured grid flow solver. As a point relaxation method, the PGS method iteratively solves the implicit operator (Eq. 3.47) by multiplying all the non-main block diagonals by the conservative variables ($\Delta Q^*$) and moving this to the RHS. This is done for each cell, sweeping from first to last cell index and then sweeping again in the reverse direction.

### 3.3.3 Generalized Minimum Residual Method

GMRES is a minimization algorithm for the norm of the residual vector of a linear equation system, which is initially developed by Saad and Schultz [27]. Given the equation system $Ax = B$, an initial guess $x_0$, and a preconditioner matrix $M$, the process leads to an approximate solution vector $x$ which minimizes the residual of the initial system. The residual is minimized by using orthogonal search directions in a Krylov subspace. The rate of convergence of Krylov subspace methods is influenced by the condition number of the matrix A. Therefore, in most cases, a preconditioning of the original system is required to achieve reasonable convergence rates. The current study uses a right-preconditioned GMRES which is given by:

$$AM^{-1}(Mx) = B \tag{3.55}$$

The right-preconditioned GMRES is preferred over the left-preconditioned GMRES because it preserves the magnitude of the residual within the linear it-

erations. Once the solution $Mx$ of Eq. 3.55 is obtained, the solution of the original system, i.e., the $x$ vector itself, is easily computed. This is because applying $M^{-1}$ is typically matrix-free, therefore it does not require storing a matrix. Algorithm 1 outlines the GMRES algorithm as presented by Behr [28]. The inner iteration loop constructs the Krylov space and projects the original equation system to this space. As the number of Krylov vector $m$ increases, both required storage and the number of operations increases linearly and quadratically. Therefore, outer iterations are often required to minimize the required number of Krylov vector. Given a fixed number of Krylov iterations, the process can be repeated using the outer iterations which is referred to as restarts. The restarted GMRES procedure improves the quality of the initial guess $x_0$.

In the current study, the GMRES method is applied to the linear system of equations shown in Eq. 3.46. As a preconditioner, one of the approximate factorization methods (ADI, DDADI, DDLGS) is applied. To save computational cost, the matrix-vector product process within the GMRES algorithm is approximated with finite difference as shown in Eq. 3.56.

$$\left[\frac{\partial R}{\partial Q}\right] x \approx \frac{R(Q+\varepsilon x) - R(Q)}{\varepsilon} \tag{3.56}$$

where $R(Q+\varepsilon x)$ is the residual evaluated by using perturbed state quantities. In this study, the $\varepsilon$ is a small scalar chosen based on the magnitude of $Q$ and it is independent of the size of the mesh. This resulting implementation of GMRES is termed matrix free. This approximaion has been validated in both inviscid and

**Algorithm 1** Right-Preconditioned GMRES Algorithm [28]

**for** $l = 1, n_{outer}$ **do**

Compute initial residual $r_0 := B - Ax_0$

Compute initial residual norm $\beta := r_0$

Define first Krylov vector $v_1 = r_0/\beta$

**for** $j = 1, m$ **do**

preconditioning $z_j := M_j^{-1} v_j$

matrix-vector product $w := Az_j$

**for** $i = 1, j$ **do**

$h_{i,j} := (w, v_i)$

$w := w - h_{i,j} v_i$

**end for**

$h_{j+1,j} := \|w\|$

Define next Krylov vector $v_{j+1} := w/h_{j+1,j}$

**end for**

Define reduced system matrix $H := h_{i,j}$

Solver reduced system matrix $y := argmin \, \|\beta e_1 - H\hat{y}\|$

Form approximate solution $x := x_0 + \sum_{i=1}^{m} y_i z_i$

**if** $\|\beta e_1 - Hy\| \leq \varepsilon$ **then**

exit

**else**

Restart $x_0 := x$

**end if**

**end for**

viscous flow simulations [29, 30].

### 3.3.4 Dual Time Stepping

In the use of an implicit time marching method, many simplifications are used in order to make it more computationally efficient: factorization, low order spatial discretization, linearization, etc. Therefore, minimizing the errors from the simplifications is conducted by iterating at each time step using a dual time stepping method. To carry out these iterations, Eq. 3.41 can be modified to consider a term that contains a fictitious pseudo time, $\tau$:

$$\frac{\partial (QV)^{p+1}}{\partial \tau} + \frac{\partial (QV)^{p+1}}{\partial t} = -R(Q^{p+1}) \tag{3.57}$$

where $p$ represents the solution at the $p_{\text{th}}$ subiteration.

Once BDF2 method is applied, the above equation is expressed as below,

$$\frac{Q^{p+1} - Q^{p}}{\Delta \tau} + \frac{3Q^{p+1} - 4Q^{n} + Q^{n-1}}{2\Delta t} = -R(Q^{p+1}) \tag{3.58}$$

It should be noted that in the subiteration process in pseudo time, one starts by setting $Q^{p=0} \equiv Q^{n}$ and when one has finished the process one should have $Q^{p+1} = Q^{n+1}$.

The generalized dual time stepping approach using delta form ($\Delta Q^{p} \equiv Q^{p+1} - Q^{p}$) results in:

$$\left[\frac{I}{h} + \left(\frac{\partial R}{\partial Q}\right)^{p}\right] \Delta Q^{p} = -\frac{3Q^{p} - 4Q^{n} + Q^{n-1}}{2\Delta t} - R(Q^{p}) \tag{3.59}$$

where, $h = \frac{\frac{2}{3}\Delta t}{1+\frac{2}{3}\frac{\Delta t}{\Delta \tau}}$

The above equation has a similar form as Eq. 3.46 and therefore can be solved using the above line-based methods. The unsteady residual at each sub-iteration time step ($p$) is given by:

$$\frac{3Q^p - 4Q^n + Q^{n-1}}{2\Delta t} - R(Q^p) \tag{3.60}$$

## 3.4 Turbulence and Transition Modeling

### 3.4.1 Turbulence Modeling

The turbulence modeling is to close the RANS equation by approximating the Reynolds-stress term (Eq. 3.26). With the assumption of isotropic eddy viscosity, the Reynolds-stress can be represented by:

$$\tau_{ij}^R = \mu_t \left[ \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right] \tag{3.61}$$

where $\mu_t$ is the turbulent viscosity. The evaluation of turbulent viscosity can be performed using various turbulence models. In the current study, either the Spalart–Allmaras (SA) turbulence model or the Menter Shear Stress Transport (SST) turbulence model was used.

#### Spalart–Allmaras (SA) Turbulence Model

In the SA model, a single transport equation is solved for the eddy viscosity, $\nu_t$. The model is designed for external aerodynamic flows, such as transonic flow

over airfoils, including boundary-layer separation. The model is widely used in aerospace applications because of its accuracy and numerical robustness. However, it is incapable of accounting for the decay of $\nu_t$ in isotropic turbulence.

The transport equation is of the form

$$\frac{D\tilde{\nu}}{Dt} = c_{b1}\tilde{S}\tilde{\nu} + \frac{1}{c_\sigma}\left[\nabla\cdot\left(\left(\nu+\tilde{\nu}\right)\nabla\tilde{\nu}\right) + c_{b2}\left(\nabla\tilde{\nu}\right)^2\right] - c_{w1}f_w\left[\frac{\tilde{\nu}}{\tilde{d}}\right]^2 \tag{3.62}$$

The eddy viscosity $\nu_t$ is computed from $\tilde{\nu}$ as shown in Eq. 3.63,

$$\nu_t = \tilde{\nu}f_{v1} \quad with \quad f_{v1} = \frac{\chi^3}{\chi^3 + {c_{v1}}^3} \quad and \quad \chi = \frac{\tilde{\nu}}{\nu} \tag{3.63}$$

The left hand side of Eq. 3.62 accounts for the convection of $\tilde{\nu}$ at the local mean flow velocity. The first term of the right hand side represents the diffusion, followed by the production and destruction terms. Further details and expressions are provided in the reference [31].

## Menter Shear Stress Transport (SST) Turbulence Model

As a two-equation model, the Menter SST turbulence model is utilized, which uses a blending function to combine $k-\omega$ and $k-\epsilon$. Close to the walls the blending function is zero (leading to the standard $\omega$ equation), whereas remote from the walls the blending function is unity (corresponding to the standard $\epsilon$ equation).

The two equation model is given by:

$$\frac{Dk}{Dt} = P - \beta^*\rho\omega^2 k + \frac{\partial}{\partial x_j}\left[\left(\mu+\sigma_k\mu_t\right)\frac{\partial k}{\partial x_j}\right] + \beta^*\rho\omega_{amb}k_{amb} \tag{3.64}$$

$$\frac{D\omega}{Dt} = \frac{\gamma}{\nu_t} P - \beta\rho\omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_j}\right] + 2(1 - F_1)\frac{\rho\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} + \beta\rho{\omega_{amb}}^2 \tag{3.65}$$

The details of the current form of the SST model can be found in the reference [32].

## Hybrid RANS-LES method

A hybrid RANS-LES method is used to extend the applicable flow conditions to high Reynolds number with massively separated flows as well as attached and mildly separated regions. Detached-eddy simulation (DES) as originally proposed is designed to treat the entire boundary layer using a RANS model and to apply an LES treatment to separated regions. Delayed-detached-eddy simulation (DDES) approach was proposed to better preserve the RANS mode, or delay LES mode, under the use of more general grids which have large cell aspect ratio [33]. In the current study, the formulation resulting in the DDES approach based on the SA turbulence model is used by modifying the length scale in the SA model. The modified length scale definition in DDES is given by:

$$\tilde{d} \equiv d - f_d \max(0, d - C_{DES}\Delta) \tag{3.66}$$

where $d$ is the wall normal distance, $\Delta$ is the local grid spacing, $f_d$ is the boundary layer shielding function, and $C_{DES} = 0.65$.

The local grid spacing is computed using a technique proposed by Scotti et al.

to account for anisotropy in the grid [34]. Below is the shielding function, which is given by the reference [33].

$$f_d = 1 - tanh\left(8r_d\right)^3 \tag{3.67}$$

In the above shielding function, the modified $r_d$ is defined as:

$$r_d \equiv \frac{\nu_t + \nu}{\sqrt{U_{i,j}U_{i,j}}\kappa^2 d^2} \tag{3.68}$$

where $U_{i,j}$ are the velocity gradients and $\kappa$ is the Von Karman constant, 0.41.

### 3.4.2 Laminar-Turbulent Transition Model Formulation

In the current study, $\gamma - \overline{Re_{\theta t}}$−SA transition model is used and is coupled with the one equation SA turbulence model. Similar with the $\gamma - \overline{Re_{\theta t}}$ transition model [35], the current model can predict Tollmien-Schlichting instability, separation-induced transition, and bypass transition using empirical criteria deduced from experimental data. The baseline formulation of the $\gamma - \overline{Re_{\theta t}}$−SA transition model has a limitation on predicting the transition onset due to crossflow instabilities in the three-dimensional boundary layer. Recently, the baseline model has been extended to account for instability due to crossflow by incorporating an existing crossflow model [36, 37]. In the current framework, the extended transition model has been implemented for more accurate simulation inside the boundary layer. The form of the extended $\gamma - \overline{Re_{\theta t}}$−SA transition model is presented in Appendix A. A more detailed description of the model can be found in the references [36–38].

The current transition model is based on two transport equations: one is for

intermittency and the other one is for transition momentum thickness Reynolds number. The transport equation for the intermittency, $\gamma$, is given by:

$$\frac{D(\rho\gamma)}{Dt} = P_\gamma - D_\gamma + \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\frac{\partial\gamma}{\partial x_j}\right] \tag{3.69}$$

The transport equation for transition momentum thickness Reynolds number, $\overline{Re_{\theta t}}$, is given by:

$$\frac{D\left(\rho\overline{Re_{\theta t}}\right)}{Dt} = P_{\theta t} + P_{CF} + \frac{\partial}{\partial x_j}\left[2.0(\mu + \mu_t)\frac{\partial\overline{Re_{\theta t}}}{\partial x_j}\right] \tag{3.70}$$

Once the transport equations for intermittency, $\gamma$, and transition momentum thickness Reynolds number, $\overline{Re_{\theta t}}$, are solved separately using DDADI, the solution of the intermittency transport equation is used to control only the production term of eddy viscosity in the SA turbulence model as follows:

$$\frac{D\tilde{\nu}}{Dt} = \gamma P_\nu - D_\nu + \frac{1}{\sigma}\left[\nabla\cdot((\nu + \tilde{\nu})\nabla\tilde{\nu}) + c_{b2}(\nabla\tilde{\nu})^2\right] \tag{3.71}$$

## 3.5 Overset Technique

Practical problems rely heavily on multiple mesh systems (one for each component of the geometry or multiple nested systems) which interact with each other. Therefore, the capabilities of the current mesh generator and solver have been extended to handle multiple mesh systems using an overset technique. Overset meshes can be particularly useful when dealing with strand grids because the near-body domain can transition to the background Cartesian mesh before the strands cross each

other at concave corners. The overset methodology typically involves three major steps: (1) identification of hole, fringe, and field cells, (2) finding donor cells and interpolation weights, and (3) performing interpolation on a data set from donor cells to a fringe cell.

The hole cells are defined as the grid elements which lie inside solid walls of another domain. These hole cells are excluded from being a part of the flow solution. Field cells are those cells in which the flow equations should be solved. Fringe cells have their conservative variables which are interpolated from field cells in another domain. The donor cells are chosen from the other domains using a search algorithm. In the case of multiple other domains overlapping, the donor cells with smaller cell volumes have been chosen.

This identification of grid elements is done using an integer *iblank* array. Each hole, fringe, and field element is tagged using an *iblank* of 0, -1, and 1, respectively. In this study, the Topology Independent Overset Grid Assembler (TIOGA) [39], is used to generate the *iblank* map between overset meshes. Once the *iblank* map is generated, the flow data is passed to TIOGA for the interpolation and exchange between the overset meshes. TIOGA performs second order transfinite linear interpolation for a given data from all interpolation donor points to fringe points.

The current solver is based on a cell-centered scheme; however, TIOGA requires grid coordinates and solutions be stored at the same location. Therefore, flow solutions are interpolated between cell-center and node points whenever the flow solutions are exchanged through TIOGA. The interpolation is performed using either a simple average or a least-squares approach.

## 3.6 Solver Architecture

Programming for the current solution algorithm is performed based on the Alg. 2 and 3. Algorithm 2 explains the residual calculation process which corresponds to net fluxes of each control volume. Algorithm 3 explains the inversion process after constructing the system of equations for each loop. Because the detailed inversion processes are different between the factorization methods (ADI, DDADI, DDLGS), only common executions are shown in the Alg. 3.

## 3.7 Python Integration Framework

The current method (HAMSTR) is developed as an alternative for near-body unstructured grid flow solvers within the multi-mesh/multi-solver paradigm. Thus, the flow solver is wrapped in Python to allow for ease of integration with other codes within a framework. HAMSTR is coupled with other in-house flow solvers which are written in different languages (FORTRAN, CUDA) and also wrapped in Python. In an overset framework, each solver is initialized separately and passes a dictionary of data pointers to Python. The grid information is used in the initialization of TIOGA for overset grid connectivity. In this implementation, TIOGA has direct access to the same memory passed through Python. Thus, the communication between flow solvers at a sub-iteration level can be performed efficiently by eliminating data transfer using file I/O. The flowchart for the current framework is shown in Fig. 3.6. This light-weight Python framework has been used for the simulations

**Algorithm 2** Residual calculation

```
for i = 1 to nchains do
    Collect all the faces forming the chain
    Collect all the cells forming the chain
    for j = 1 to chainsize do
        Copy conservative variables to one-dimensional array for the chain
        Apply boundary conditions (wall, far-field, MPI, etc.)
    end for
    Reconstruct left and right states using selected reconstruction scheme
    for j = 1 to chainsize do
        Use Riemann solver to find interface inviscid fluxes and Jacobians
        Compute interface viscous fluxes and Jacobians
        Add interface fluxes at each face to corresponding cells
        if turbulent or transition model then
            Compute residuals and Jacobians of transport equations
        end if
    end for
end for
```

**Algorithm 3** Inversion

**for** $i = 1$ to nchains **do**

Collect all the faces forming the chain

Find left and right state Jacobians along the faces

**if** Chain is closed **then**

Construct a periodic block tri-diagonal system

Invert periodic block tri-diagonal system using Sherman-Morrison algorithm

**else if** Chain is opened **then**

Construct a non-periodic block tri-diagonal system

Invert block tri-diagonal system using Thomas algorithm

**end if**

Update right hand side with result from the inversion

**end for**

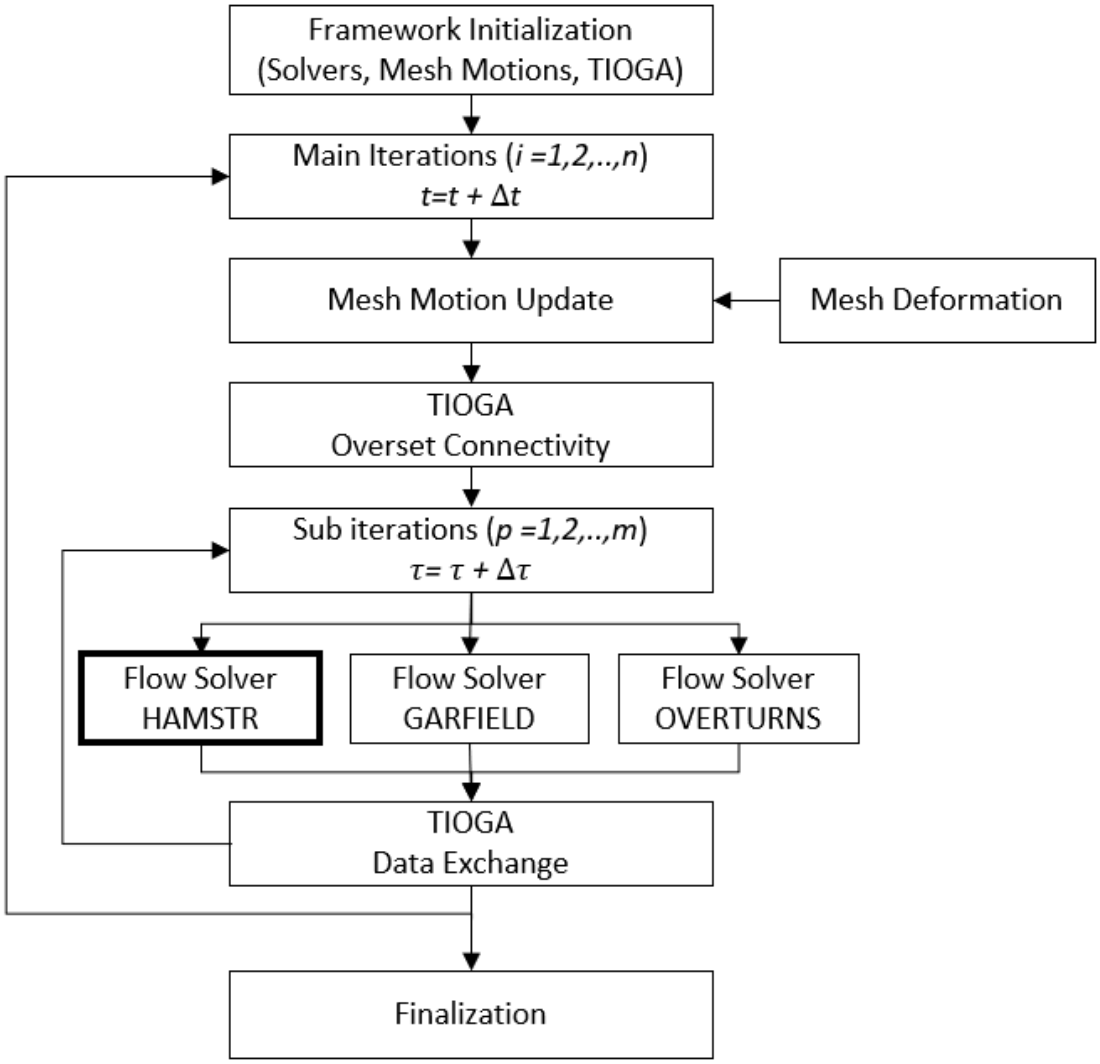



Figure 3.6: Flowchart of Python framework for time-accurate simulations.

of complete configurations, such as complete rotorcraft/wind turbine configurations (see references [40, 41]).

## 3.8 Summary

In this chapter, the computational methodology of the flow solver is presented. The capability of the flow solver is extended for rotaty wing system simulations.

- The inviscid fluxes are reconstructed using either stencil- or gradient-based reconstruction depending on the type of mesh. For example, the stencil-based reconstruction such as third-order MUSCL or fifth-order WENO can be used for a domain of structured or semi-regular surface mesh. The gradient-based reconstruction which uses linear least-squares to estimate gradient can be applied in a domain of unstructured type mesh. For a strand volume grid, a combined method can be used; the stencil-based reconstruction is used in a wall-normal direction, otherwise the gradient-based reconstruction is used in wall-tangential directions for a unstructured type surface grid.

- The viscous fluxes are computed using either finite difference or least-squares method which are both second-order accurate on any type of grid.

- One equation Spalart-Allmaras (SA) and two equations Menter Shear Stress Transport (SST) turbulence models are integrated to the current method for Reynolds Averaged Navier–Stokes (RANS) simulations.

- Various line-implicit methods are applicable along a line structure: Alternating Direction Implicit (ADI), Diagonally Dominant ADI (DDADI), and Diagonally Dominant Line Gauss Seidel (DDLGS). Implicit operator becomes block-tridiagonal system for each path and it can have periodic system for closed loop or non-periodic system for open loop.

- A right-preconditioned Generalized Minimum Residual (GMRES) method is implemented to improve solution convergece rate. Preconditioned step is per-

formed using DDLGS method.

- Time-accurate methods with dual-time-stepping strategies are implemented for time-dependent problems. Non-uniform grid motion is accounted for the simulation with rotating objects. In a computation of grid velocity, Geometric Conservation Law (GCL) is considered to ensure free-stream preservation in the case with deformed grids.

- The flow solver is parallelized using a Massage Passing Interface (MPI).

- The capabilities of the mesh generator and flow solver have been extended to handle multiple mesh system using an overset technique. The TIOGA, a overset grid assembler, is used to generate *iblank* map and interpolate flow variables at an overset boundary.

- The flow solver is wrapped in Python to allow for ease of integration with other codes within a framework. The current solver is coupled with other in-house flow solvers which are written in different languages (Fortran, CUDA) and architecture (GPU) within the Python framework.

# Chapter 4: Validation

In this chapter, the current method will be validated in terms of both numerical accuracy and efficiency. The validations are conducted using relatively simple and well-known problems before applying the current method to complex rotaty wing systems. First, formal spatial order of accuracy is evaluated on both structured and unstructured grids. Following this, the implementations for turbulent flow simulation are validated. Finally, the performance of current flow solver is evaluated by comparing the results with either well-established flow solvers or a traditional unstructured grid based method.

## 4.1 Solution Accuracy Analysis

### 4.1.1 Method of Manufactured Solution

The formal order of solution accuracy is tested using a Method of Manufactured Solution (MMS) [42]. With MMS, the "manufactured solution" can be prescribed arbitrarily, which is not physically realistic but should be simple, smooth, and exercise all terms in the governing equations. By substituting the arbitrary solution into the governing equations, analytic source terms can be obtained. Then,

the discretized equation is solved with the addition of the source terms. Finally, the solution error can be estimated by comparing with the exact manufactured solution. In this study, the manufactured solution from reference [13] is used, which is given by:

$$\begin{aligned}
\rho(x,y,z) &= \rho_0 + \rho_1 exp\left(\frac{-x^2-y^2-z^2}{L^2}\right) \\
u(x,y,z) &= u_1 sin\left(\frac{x^2+y^2+z^2}{L^2}\right) \\
v(x,y,z) &= v_1 cos\left(\frac{x^2+y^2+z^2}{L^2}\right) \\
w(x,y,z) &= w_1 sin\left(\frac{xy+xz+yz}{L^2}\right) \\
p(x,y,z) &= p_0 + p_1 sin\left(\frac{x}{L}\right) sin\left(\frac{y}{L}\right) sin\left(\frac{z}{L}\right)
\end{aligned} \tag{4.1}$$

where $\rho_0$ is 1.0, $p_0$ is $\frac{1}{\gamma}$, $\rho_1$,$u_1$,$v_1$,$w_1$ and $p_1$ are set to 0.1 and $L$ is the size of the domain, which is set to 1.0.

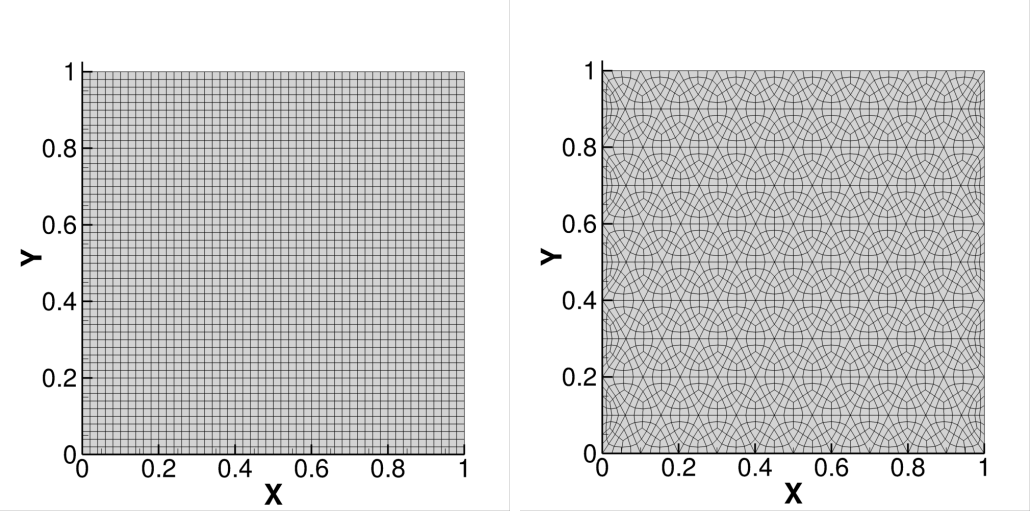
The MMS analysis is performed using either a stencil-based reconstruction or a gradient-based reconstruction on the representative two-dimensional Cartesian and unstructured domains. For a stencil-based reconstruction, both MUSCL and WENO schemes are tested. For a gradient-based reconstruction, two different stencils are used; standard stencil and wrapping stencil. The solution errors are computed by refining the mesh in both directions. When the meshes are refined, the size of domain is fixed as 1 by 1.

Figure 4.1 shows one of the test grids from each Cartesian and unstructured grid. In the case of an unstructured grid, the initial triangles are subdivided into 12 quadrilaterals using quad-level 1. While computing the discrete solution, a Dirich-

let boundary condition is imposed along the outer boundary, which is the exact manufactured solution. The L2 norm of solution error (density, $\rho$) is defined by:

$$L^2_{error} = \left[ \frac{\sum_{i=1}^{N} error^2 |J|}{\sum_{i=1}^{N} |J|} \right]^{1/2} \tag{4.2}$$

where $J$ corresponds to the cell volume and N is the number of cells.

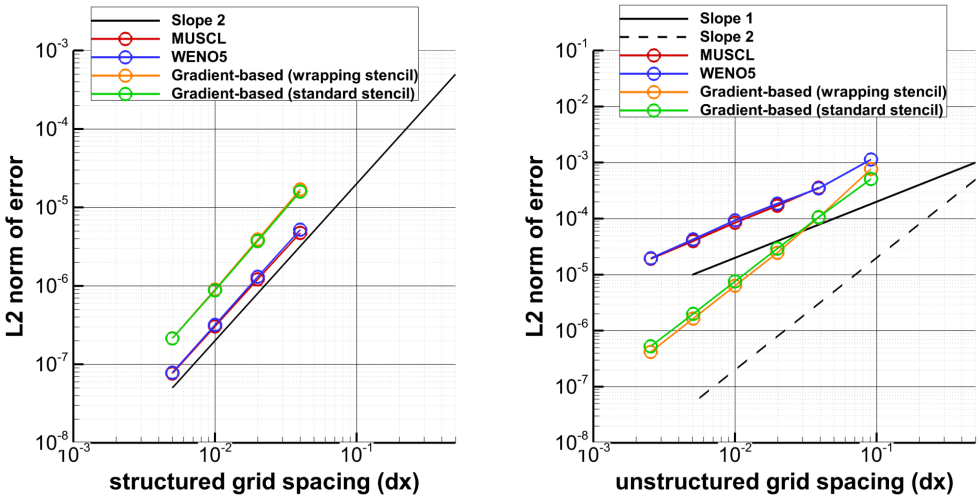



(a) Cartesian grid (b) Unstructured grid

Figure 4.1: Representative 2D meshes for MMS analysis.

As shown in Fig. 4.2 (a), the errors from both reconstruction methods show second order of accuracy on the Cartesian grid. The results are expected because the current finite volume method evaluates the fluxes at each face-midpoint using one quadrature point (as many other finite volume based flow solvers do). It should be noted that the line reconstruction methods can provide formal order of accuracy (third and fifth order for MUSCL and WENO, respectively) using a finite different formulation. However, it is limited to a simple Cartesian grid. On an unstructured grid as shown in Fig. 4.2 (b), both MUSCL and WENO show only the first order

of accuracy due to the varying cell size and curvature along the loops. This result highlights the limitation of the finite volume approach in the use of line reconstruction method on a general unstructured grid. However, gradient-based methods show its formal order of accuracy (second order) on the unstructured grid as we expected. When the results using least-squares method are compared with each other, the solution accuracy is rarely improved by using the wrapping stencil on either the Cartesian or unstructured grid.




(a) Cartesian grid (b) Unstructured grid

Figure 4.2: Solution error convergence using MMS analysis.

### 4.1.2 Isentropic Vortex

The convection of a two dimensional inviscid isentropic vortex is simulated to examine the accuracy of stencil-based reconstruction methods on an unstructured grid. During the time-accurate simulation, the intermediate solutions are extracted and these solutions are compared against the exact solution. Considering there is no

dissipation of this canonical isentropic vortex, the exact solution at any time is the same as the initial condition shifted in space. The freestream flow values are set as $(\rho_\infty, u_\infty, v_\infty, p_\infty, T_\infty) = (1, 0.5, 0, 1, 1)$. The solution domain is $[-7, 7] \times [-3.5, 3]$, and all boundary conditions are periodic in nature. The solution domain is composed of 11,700 quadrilateral cell elements which are generated from the 975 equilateral triangles using quad-level 1 subdivision. The non-dimensional time step size is set as 0.2. At $t = 0$, the flow is perturbed by an isentropic vortex $(\delta u, \delta v, \delta T)$ centered at $(x_0, y_0)$ given by

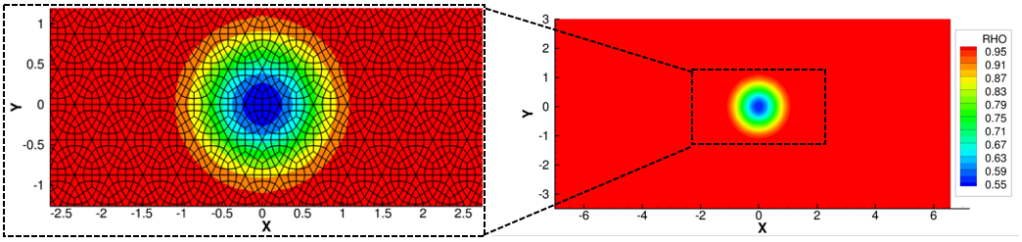
$$\begin{aligned} \delta u &= -\frac{\alpha}{2\pi}(y - y_0)e^{\phi(1-r^2)} \\ \delta v &= \frac{\alpha}{2\pi}(x - x_0)e^{\phi(1-r^2)} \\ \delta T &= \frac{\alpha^2(\gamma - 1)}{16\phi\gamma\pi^2}e^{2\phi(1-r^2)} \end{aligned} \tag{4.3}$$

where $\phi = 1.0$, $\alpha = 4.0$ and $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$.

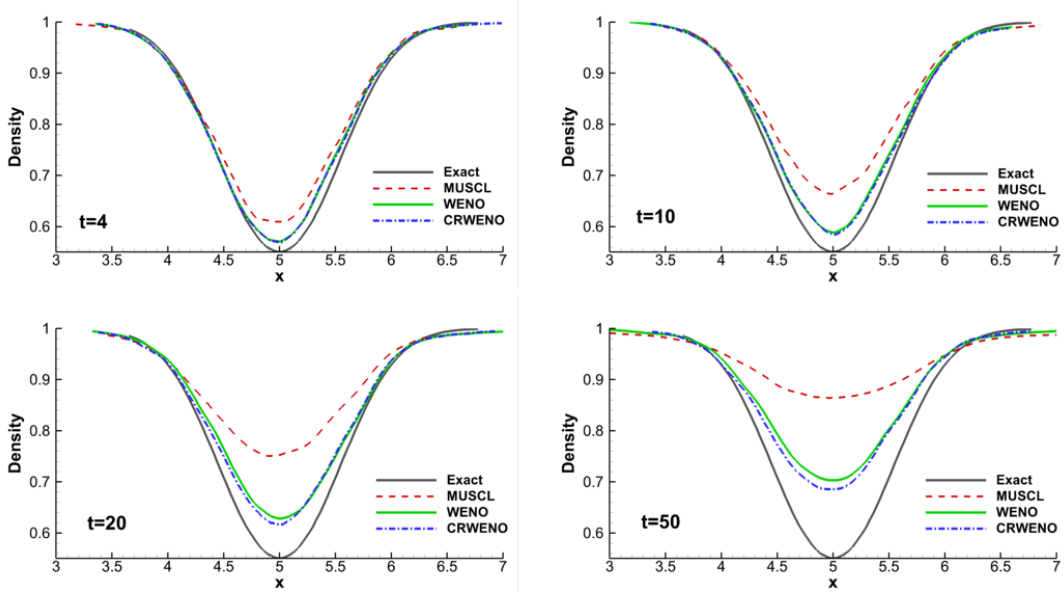
The solutions are extracted at times $t = 4$, $t = 10$, $t = 20$ and $t = 50$ using the second order accurate backward difference formula (BDF2) time marching scheme and three different reconstruction schemes: MUSCL, WENO, CRWENO [43].

Figure 4.3 (a) illustrates the computational mesh and the initial density contours in the domain. For quantitative comparison, the computed density profiles along the horizontal centerline are compared against the exact solution at times $t = 4$, $t = 10$, $t = 20$ and $t = 50$ as shown in Fig. 4.3 (b). For ease of comparison, the density profiles are aligned at the same vertical line $(x = 5)$ for all cases. From the figure, it can be seen that the vortex core is better conserved using a higher order scheme, such as fifth-order CRWENO or fifth-order WENO scheme, than using a

third-order MUSCL scheme as the solution evolves.




(a) Grid and initial density contours for the case of isentropic vortex convection.




(b) Comparison of the density distribution across the vortex core.

Figure 4.3: Grid and density profiles across the vortex core at different solution times for the case of isentropic vortex convection.

From the solution accuracy tests, although the stencil-based reconstructions do not guarantee its formal order of accuracy along the Hamiltonian path on an unstructured grid, the higher-order type method provides less dissipation error then a lower-order method.

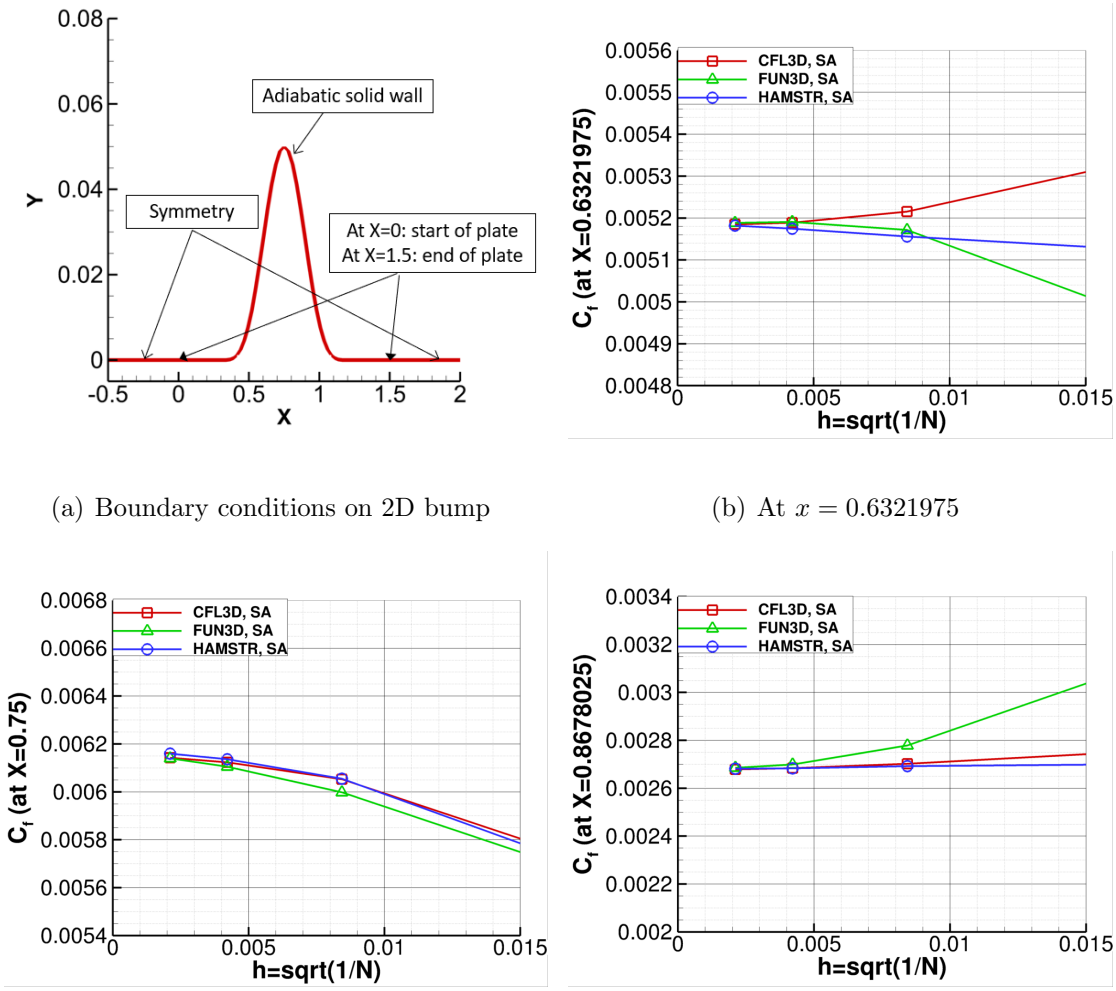
### 4.1.3 Two-Dimensional Bump in Channel

The 2-D bump-in-channel case is conducted as a turbulence flow validation of the current method [44]. This case assumes fully turbulent flow and the SA turbulence model is used. As shown in Fig. 4.4, the top of the bump is located at $x = 0.75$, and the adiabatic solid-wall boundary condition is imposed from $x = 0$ to $x = 1.5$; otherwise, the symmetric boundary condition is imposed along the surface at $y = 0$. The freestream Mach number of 0.2 and Reynolds number of 3 million per grid unit are used for this case. The grid refinement study is performed using the structured 2-D grid family obtained from the TMR website [44] and the detailed grid information is shown in Table 4.1.

Table 4.1: Two-dimensional structured mesh information for Bump simulation.

| | Tiny | Coarse | Medium | Fine |
|---|---|---|---|---|
| Size(points on the wall) | 89x41(41) | 177x81(81) | 353x161(161) | 705x321(321) |
| Initial normal spacing($y^+$) | 8e-6(0.95) | 4e-6(0.47) | 2e-6(0.24) | 1e-6(0.12) |

The grid refinement study is performed by comparing the surface skin friction value against the values from the other flow solvers as shown in Fig. 4.4. The three different test points are located near the top of the bump. The current results are converged to the same value with the reference results from CFL3D and FUN3D flow solvers. It should be also noted that the current skin friction values are converged relatively faster using the fifth-order WENO scheme for the reconstruction than the reference values (especially from FUN3D which uses the unstructured MUSCL scheme).




(a) Boundary conditions on 2D bump

(b) At $x = 0.6321975$

(c) At $x = 0.75$

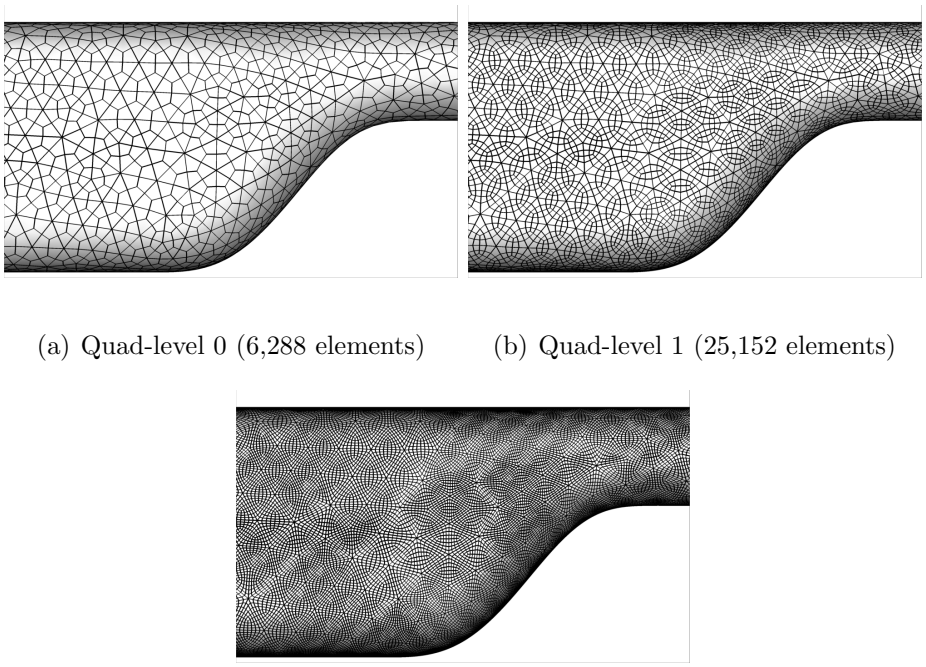
(d) At $x = 0.8678025$

Figure 4.4: Comparison of skin friction at three different locations on the bump.

### 4.1.4 Three-Dimensional Robin-Mod7 Fuselage

A grid refinement study for the Robin-Mod7 fuselage configuration is also performed using the current method. The boundary layer is assumed as fully turbulent flow at the freestream Mach number of 0.1 and Reynolds number of 1.6 million based on the fuselage length.

Three different resolution surface grids are generated from the same 2,096



(a) Quad-level 0 (6,288 elements) (b) Quad-level 1 (25,152 elements)

(c) Quad-level 2 (100,608 elements)

Figure 4.5: Three different resolution surface meshes for Robin-Mod7 fuselage.

triangle elements by applying additional quad-levels. Figure 4.5 shows the resultant quadrilateral elements on the surface near the fuselage ramp. The grid spacing in the wall-normal direction is kept sufficiently fine as $5 \times 10^{-6}$ fuselage length, which corresponds to a $y^+ = 0.3$.

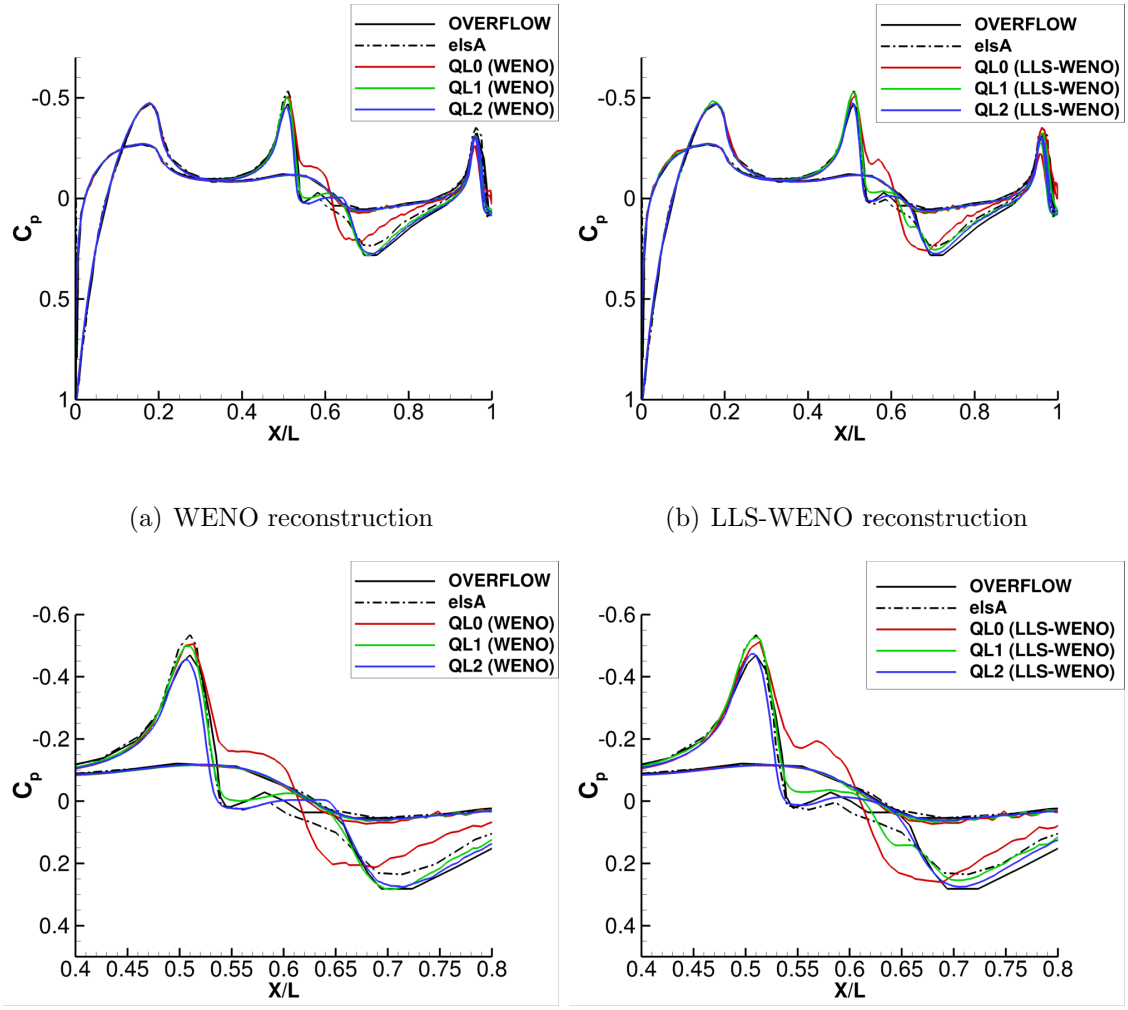
In this grid refinement study, the inviscid fluxes of the wall-tangential directions are reconstructed using two different schemes; WENO or LLS. Otherwise, the inviscid fluxes of the wall-normal direction are reconstructed using only WENO scheme along each strand. For the fully turbulent flow simulation, the SA turbulence model is used.

The surface pressure distributions on longitudinal plane ($y/L = 0$) are compared between the different meshes at $0^\circ$ angle of attack as shown in Fig. 4.6. Cur-

rent predictions are also compared against results from OVERFLOW and elsA [45]. Both OVERFLOW and elsA solvers used the same multi-block structured meshes which include a total of about 128,000 grid points on the surface. The current predictions from both WENO and LLS-WENO schemes approach the reference results especially from OVERFLOW solver as the surface grid is refined. In Figs. 4.6 (c) and (d), the flow separation location on the rear ramp region is well captured using the quad-level 2 surface mesh, otherwise the separation is predicted earlier using the quad-level 0 surface mesh.

The span-wise surface pressure distributions are extracted halfway down the ramp at $z/L = -0.0375$ as shown in Fig. 4.7, where the surface pressure taps were installed along the line in the experiment [45]. To ensure a fully-developed turbulent boundary layer, boundary-layer trips were installed near the nose of the fuselage in the experiment. The current predictions using the different meshes are compared against the result from OVERFLOW (128,000 surface grid points). The current predictions using the quad-level 1 and 2 meshes are generally matched with the result from OVERFLOW over the current span which includes the separated flow region. It is observed that the differences between the results using either WENO or LLS-WENO schemes are less as the surface grid is further refined.

Figure 4.8 shows the predicted fuselage drag coefficients using the different resolution surface meshes. The current results are also compared with the available reference simulation results [45]. In Fig. 4.8 (a), the current predictions from both schemes are converged asymptotically to the reference result. It should be noted that some differences between the solvers are observed because the pressure drag



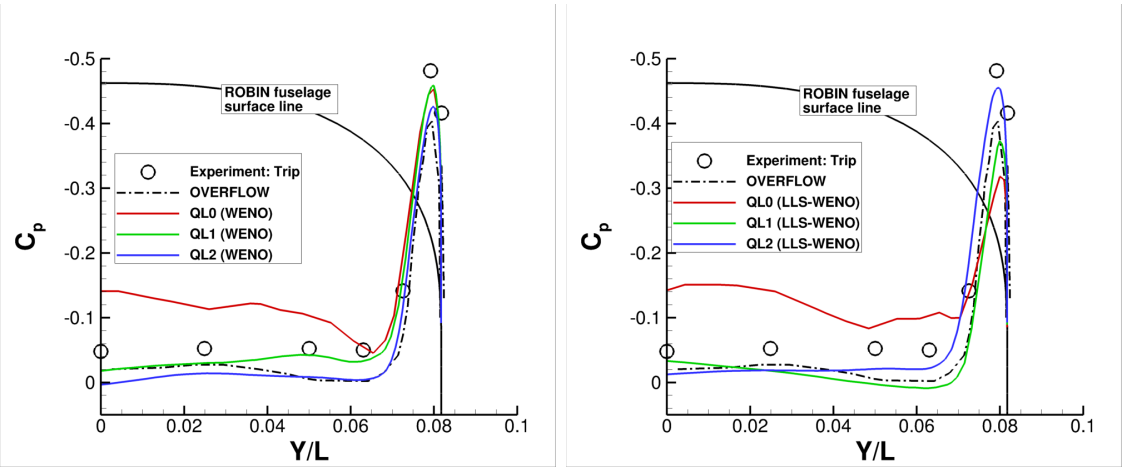

(a) WENO reconstruction

(b) LLS-WENO reconstruction

(c) WENO reconstruction (zoomed-in)

(d) LLS-WENO reconstruction (zoomed-in)

Figure 4.6: Robin-Mod7 fuselage surface pressure distribution on longitudinal plane ($y/L = 0$) at AoA=0°.

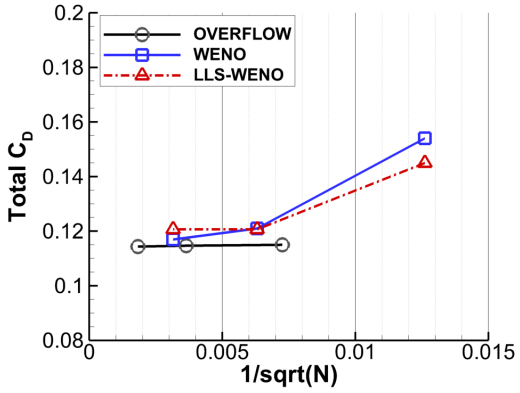



(a) WENO reconstruction

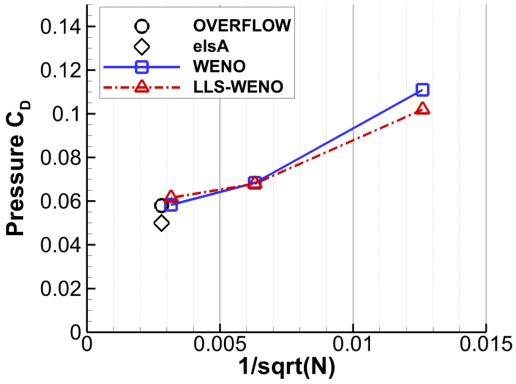
(b) LLS-WENO reconstruction

Figure 4.7: Robin-Mod7 fuselage span-wise surface pressure distribution ($z/L = -0.0375$) at AoA=0°.

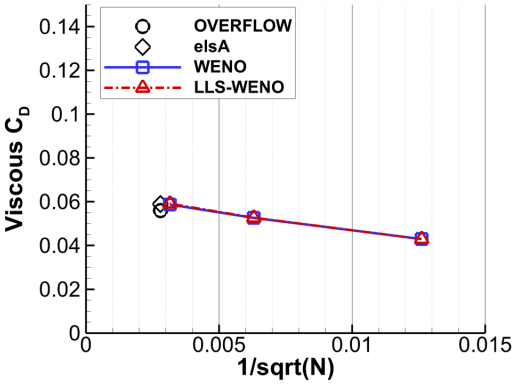
is highly dependent on the predicted separation location at the ramp region (see Fig. 4.8 (b)). Otherwise, less deviation occurs between the solvers for the viscous drag component as shown in Fig. 4.8 (c). Figure 4.8 (d) shows the fuselage drag coefficient break down of the viscous and pressure contributions. In this comparison, both OVERFLOW and elsA results were obtained from the surface mesh including about 128,000 points which is similar to the current quad-level 2 mesh. The current results using the quad-level 2 mesh are well matched with the reference results especially from OVERFLOW solver. Also, minor differences are observed between the current results using either WENO or LLS-WENO schemes.




(a) Total drag coefficient




(b) Pressure drag coefficient




(c) Viscous drag coefficient

| | Viscous $C_D$ | Pressure $C_D$ |
|---|---|---|
| OVERFLOW | 49 % | 51 % |
| elsA | 54 % | 46 % |
| QL 0 | 28 % (30 %) | 72 % (70 %) |
| QL 1 | 43 % (44 %) | 57 % (56 %) |
| QL 2 | 50 % (49 %) | 50 % (51 %) |

(d) Drag ratio: WENO (WENO-LLS)

Figure 4.8: Robin-Mod7 fuselage drag comparison at AoA=0° (N is the number of surface node points).

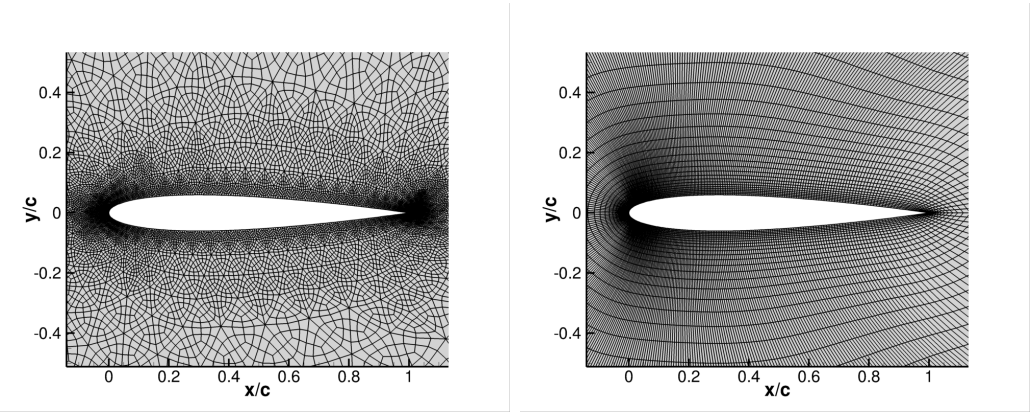
## 4.2 Performance Analysis

### 4.2.1 Transonic Flow Past NACA0012 Airfoil

Stability and accuracy of the current method on both a structured and an unstructured grid are evaluated using a standard test case; inviscid transonic flow over a NACA0012 airfoil at $M = 0.8$ and $\alpha = 1.25°$. For the unstructured grid, the initial grid has 1,732 triangles, and total of 20,784 quadrilateral elements are obtained after quad-level 1 subdivision (see Fig. 4.9 (a)). The domain is extended until 50 chords length from the airfoil surface and the far-field boundary condition is imposed along the far-boundary. For the structured grid, an O-type topology mesh is obtained with $400 \times 53$ dimensions (see Fig. 4.9 (b)).

For a fair comparison, the number of total elements (20,784), the number of surface grid points (400), and the size of far-boundary (50 chords) are matched between the unstructured and structured grids. The current solver performance is compared with an in-house solver at the University of Maryland, TURNS, which is one of the fast structured grid based flow solver [46]. TURNS uses the same structured grid in this comparison.

HAMSTR flow solver uses either stencil-based reconstruction (MUSCL) or gradient-based reconstruction (LLS) for each structured or unstructured grid, respectively. For implicit inversion, DDLGS approximate line method is compared with point Gauss-Seidel (PGS) which is a traditional method of typical unstructured grid solvers. TURNS flow solver uses MUSCL reconstruction for inviscid fluxes



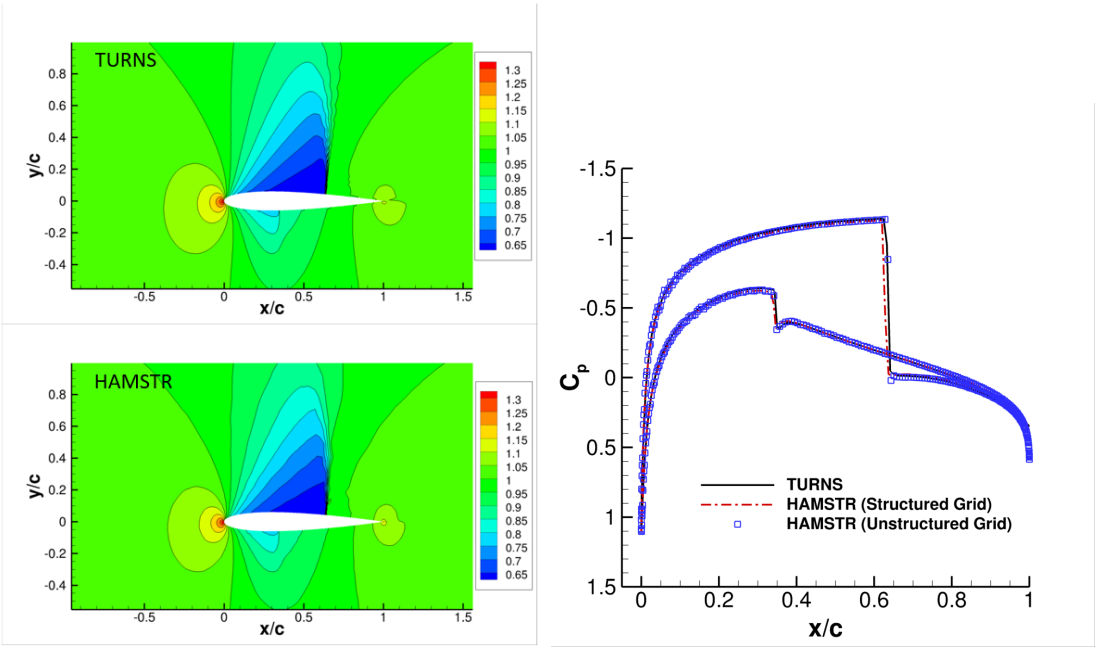

(a) Unstructured grid for HAMSTR (b) Structured grid for both HAMSTR and TURNS

Figure 4.9: Computational mesh for transonic NACA0012 airfoil simulation.

and either LU-SGS (Lower-upper Symmetric Gauss-Seidel) or DDLGS method for implicit inversion. The LU-SGS scheme is a popular inversion algorithm for structured grids and can also be implemented for unstructured grids using a coloring scheme. For a steady simulation, both solvers use a constant CFL number over the domain. The maximum allowable CFL number for convergence is used for each implicit method; CFL number of 20 for DDLGS and PGS, and 50 for LU-SGS.

Figure 4.10 (a) shows the comparison results of density contour around the airfoil using the structured grid for TURNS and the unstructured gird for HAMSTR. In both solutions, the shocks on both the upper and lower surfaces are captured with a good agreement. Figure 4.10 (b) shows the comparison results of surface pressure distribution. In addition to the upper surface shock, the lower surface shock is resolved quite accurately on both structured and unstructured grids using the current method when it is compared with TURNS.

Figure 4.11 shows the comparison of residual convergence rate and drag history




(a) Density contour (b) Surface pressure coefficient

Figure 4.10: Comparison results with TURNS for transonic NACA0012 airfoil simulation.

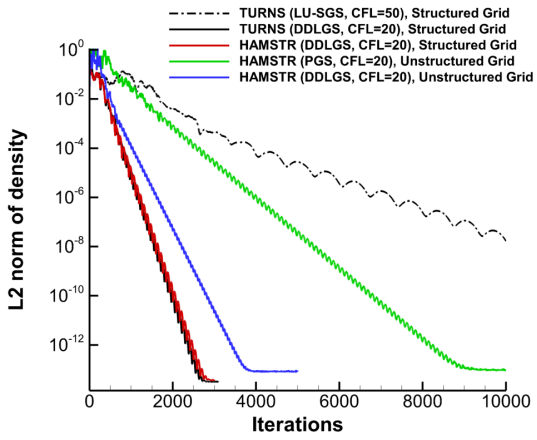
with corresponding solution results from TURNS. The solution residuals drop to machine zero precision in all of simulation results as shown in Fig. 4.11 (a). First, the convergence rates from TURNS are quite different between LU-SGS and DDLGS; DDLGS shows a much faster convergence rate than LU-SGS. The main reason for the difference is due to the different linearization methods. The current DDLGS method is working with the linearization of the Roe flux for the flux Jacobian; on the other hand, the LU-SGS method uses the spectral radius approximation. Although the use of the approximation saves the computational cost per iteration, it generally provides less convergence rate per iteration. Consequently, the DDLGS convergence rate per CPU time is about four times faster than the LU-SGS results as shown in Fig. 4.11 (b).

The convergence rate from HAMSTR for the structured grid is quite simiar to the corresponding result from TURNS in terms of both per iteration and CPU time. This result demonstrates the good efficiency of HAMSTR on structured grids. The convergence rates from HAMSTR are also compared between the structured and unstructured grids. On the unstructured grid, the slower convergence rate per iteration is observed mostly because of the use of a different reconstruction scheme (LLS).

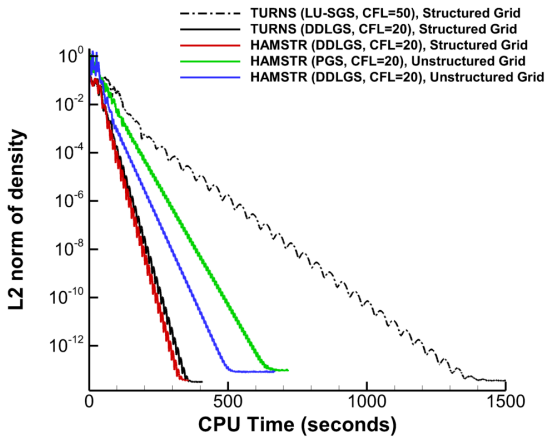
Finally, the solution convergence rates between a line-implicit method (DDLGS) and a point-implicit method (PGS) are compared on the unstructured grid. The convergence rate per iteration is significantly improved by using the DDLGS method as shown in Fig. 4.11 (a). This demonstrates the improved efficiency using line-implicit methods on unstructured grids. The difference becomes less significant when comparing CPU time as shown in Fig. 4.11 (b). This is because the PGS method does not require solving a large block-tridiagonal system but does require solving a $4 \times 4$ matrix at each cell in two-dimensions, which makes it more efficient on a per cycle basis. It should be noted that the difference in convergence rate between PGS and DDLGS becomes more noticeable as the grid size or the average length of loops increases; because the point Gauss-Seidel is a local technique.

The comparison of drag convergence history per CPU time is shown in Fig. 4.11 (c). Overall, a similar trend is observed in the drag as with the result of residual convergence rate. Almost identical results are observed between TURNS and HAMSTR on the same structured grid. The convergence rate using DDLGS is faster than the result using PGS on the same unstructured grid. Table 4.2 shows the required

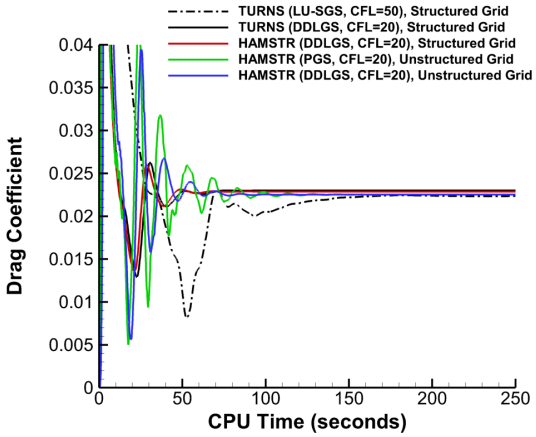
CPU time for each simulaion to obtain the converged drag prediction within 1% error from the converged value. On one hand, a similar CPU time is observed between TURNS and HAMSTR using the same schemes and grid. On the other hand, about 28% more computational efficiency is acheived using DDLGS method than PGS method on the unstructured grid.




(a) Solution residual vs iterations




(b) Solution residual vs CPU time




(c) Drag coefficient vs CPU time

Figure 4.11: Solution convergence comparison with TURNS for transonic NACA0012 airfoil simulation.

Table 4.2: CPU execution time for drag prediction within 1% of converged value (S: structured grid, U: unstructured grid).

| Solver | Grid | Inversion | Reconstruction | CPU time (seconds) |
|---|---|---|---|---|
| TURNS | S | LU-SGS | MUSCL | 140 |
| TURNS | S | DDLGS | MUSCL | 66 |
| HAMSTR | S | DDLGS | MUSCL | 60 |
| HAMSTR | U | PGS | LLS | 100 |
| HAMSTR | U | DDLGS | LLS | 72 |

### 4.2.2 Laminar Flow Past a Sphere

Laminar flow past a sphere for a range of Reynolds number from 25 to 200 is simulated to validate the current viscous flux implementation and to examine the solution convergence rate of the current method.

#### Spherical Grid Generation

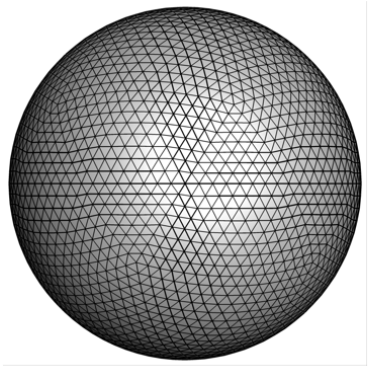
A prerequisite to generate Hamiltonian paths and strand grids around a sphere is to first generate a triangular surface mesh. The surface triangular mesh is generated using repeated subdivision of an icosahedron. The icosahedron is a polyhedron with 20 triangular faces whose nodes all lie on a sphere. At each stage of the subdivision, each triangle is subdivided into four more triangles resulting in a quadruple increase in the number of cells. The newly formed nodes of the triangles are flushed to the surface of the sphere and the node connectivity is recreated. Figure 4.12 (a) shows 5,120 isotropic triangles on the surface of a sphere starting from an icosahe-

dron, and Fig. 4.12 (b) shows the colored paths on the resulting 15,360 quadrilaterals which are obtained using quad-level 0. Strand grids are projected from the surface in the radially outward direction as shown in Fig 4.12 (c). The sphere has a unit diameter and the volume mesh extends until 42 diameters using 55 layers of strand grid, which results in a total of 829,440 hexahedrons. The initial wall spacing is set to $1 \times 10^{-3}$ diameter and a mesh stretching ratio of 1.17 is chosen for the laminar flow simulation.

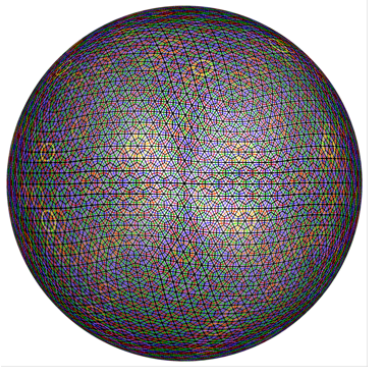
As another option for the volume meshing, the unstructured elements are used around the sphere. From the 9,754 triangular surface elements, 23 prismatic layers are generated using the same initial wall spacing as with the strand grid. The region outwards from the last prismatic layer until the far-boundary surface is then filled with 382,564 tetrahedral elements. Using quad-level 0 subdivision for the volume elements, the surface is discretized with 29,262 quadrilaterals and a total of 2.2 million hexahedra are generated for the simulation (see Fig. 4.12 (d)).

## Simulation Results

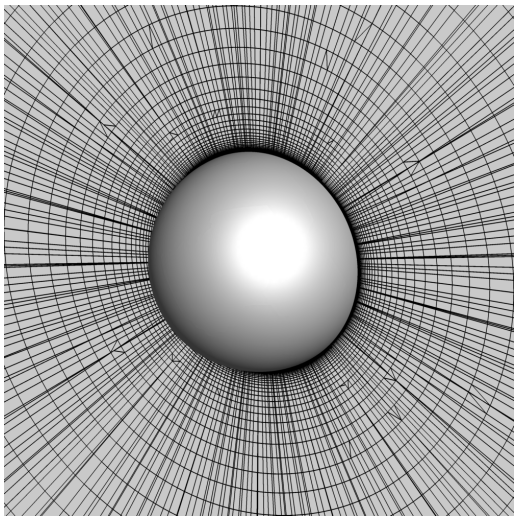
Laminar flow over a sphere is performed at the freestream Mach number of 0.2 and Reynolds number ranging from 25 to 200. Figure 4.13 shows streamlines around the sphere at each Reynolds number. For qualitative comparison of the size of the separation bubble, the results from Johnson and Patel [47] are shown in Fig. 4.13 (a). Current results are shown in Fig. 4.13 (b) and (c) by using the strand grid or the unstructured volume grid, respectively.



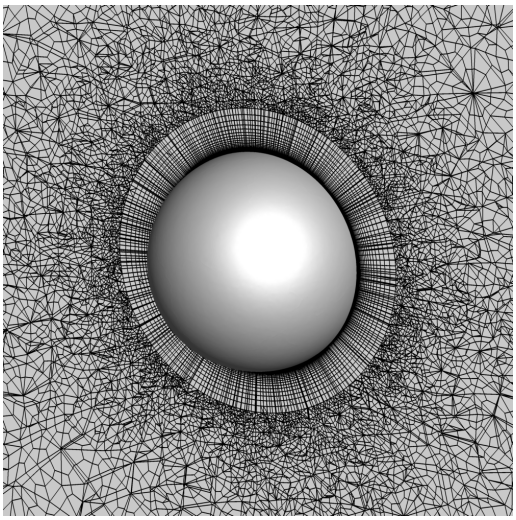
(a) Initial triangles on a sphere



(b) Hamiltonian paths on quadrilaterals



(c) Strand grid sectional view



(d) Unstructured volume grid sectional view

Figure 4.12: Surface and volume mesh for flow over a sphere.

The current solutions are obtained using gradient-based reconstruction in the wall-tangential directions and WENO scheme in the wall-normal direction for the inviscid flux (LLS-WENO). Symmetric wake structures behind the sphere are observed at these low Reynolds numbers and the size of the wake region increases as the Reynolds number increases. The flow structures remain topologically similar

between Reynolds numbers with changes only in the separation distance (separation bubble length) and the center of the separation bubble. The symmetric wake structures are captured better using the strands grid than the unstructured grid. This is because the grid is symmetric itself and the grid quality in the wake region is better using the strand grid.

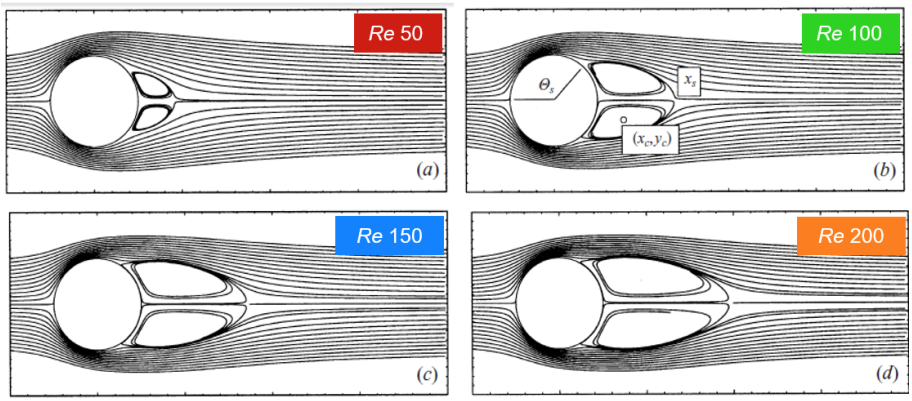
Current predicted drag coefficients are compared with experiment and other simulation results [13, 47] as shown in Fig. 4.14 (a). mStrand results are obtained using a similar number of surface elements (16,180) with the current strand mesh. Overall the current predictions are well matched with both the experimental and mStrand results. Additionally, separation distance and location of the separation bubble are compared with other simulation results as shown in Fig. 4.14 (b) and (c). All of the predicted flow features show a good agreement with results from the references.

Figure 4.14 (d) shows the residual convergence histories at four different Reynolds numbers. The same CFL number of 50 is used for all of the cases. Using the strands grid, the residuals are observed to converge about 9 orders in the cases up to 150 Reynolds number, and 6 orders at 200 Reynolds number within 3,000 iterations. Although reduced convergence rates are observed using the unstructured volume grid, overall similar trends are observed between the different grids. Note that the rapid convergence is shown initially for all cases, which corresponds to the damping of the higher frequencies in the solution.

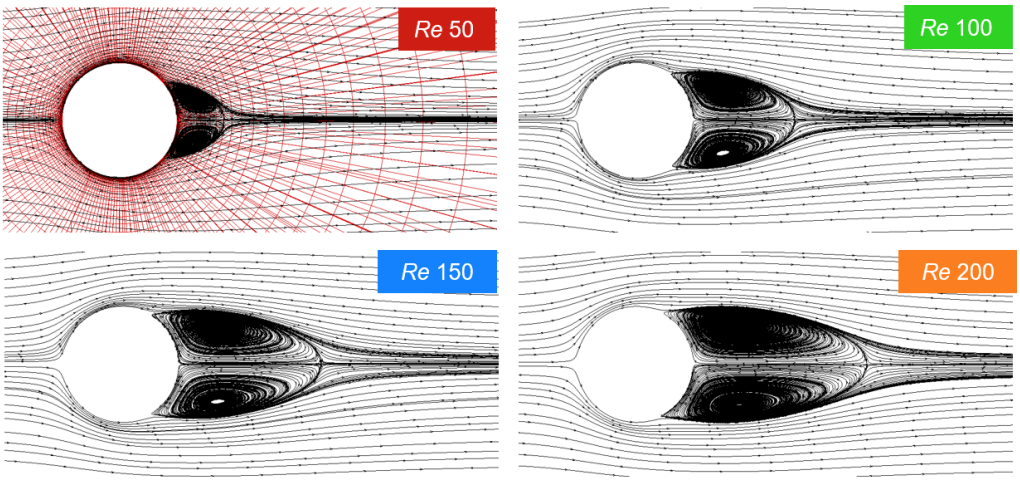
The solution convergence rates are compared between PGS and DDLGS methods at 100 Reynolds number using both strands and unstructured grids. The same

reconstruction method (LLS-WENO) is used during the comparison. Figure 4.15 shows the solution residual convergence along with the drag coefficient history in the case of the strand grid. The maximum allowable CFL number of 20 is used for the PGS method. Overall, a similar trend with the previous two-dimensional results are observed: the DDLGS method outperforms the PGS method in both per iteration and CPU time. However, the effect of line-implicit method on the convergence becomes more significant in this three-dimensional viscous simulation. This is because the line-implicit method is allowed along each strands in the wall-normal direction. To reach machine zero precision residual, about 3 times higher computational efficiency is observed from the DDLGS method.

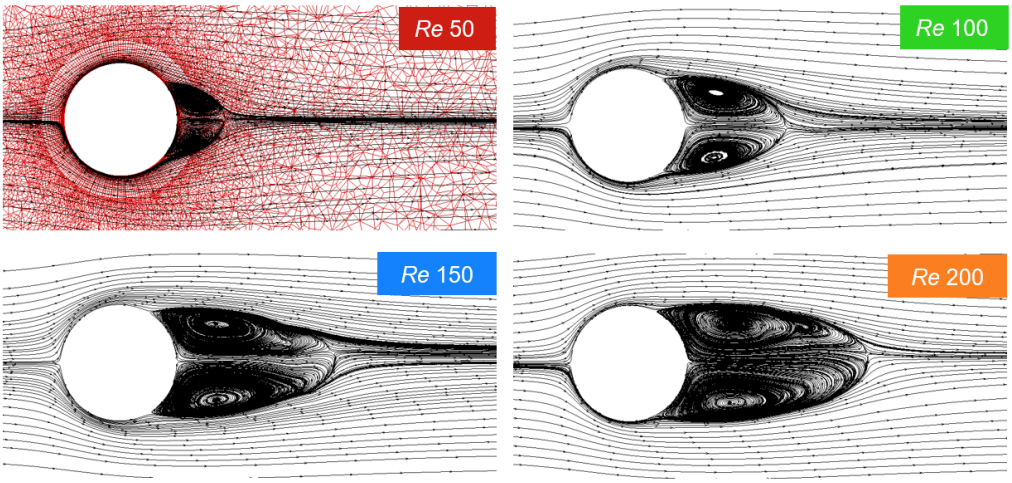
Figure 4.16 shows the solution residual convergence along with the drag coefficient history in the case of the unstructured volume grid. The same CFL number of 50 is able to be used for both the DDLGS and PGS methods. To reach the same solution residual drop (e.g. $2 \times 10^{-8}$), the DDLGS method shows about 2 times higher computational efficiency than the PGS method as shown in Fig. 4.16 (b). The advantage of the DDLGS method on the efficiency is decreased from the results using the strands grid. One of the possible reasons is that the average length of the loops is reduced by using tetrahedral elements instead of strands.




(a) Reference streamlines [47]

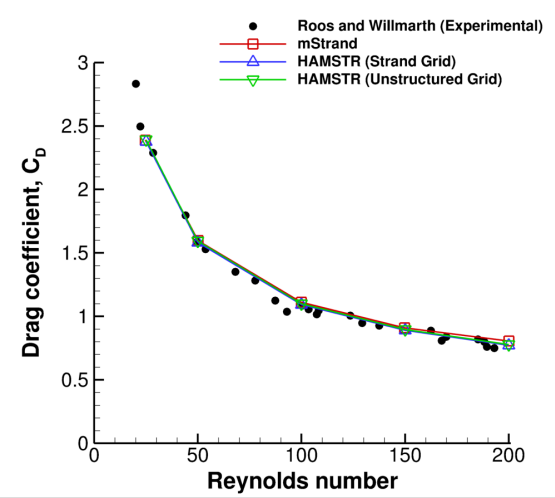



(b) Current results using the strands grid

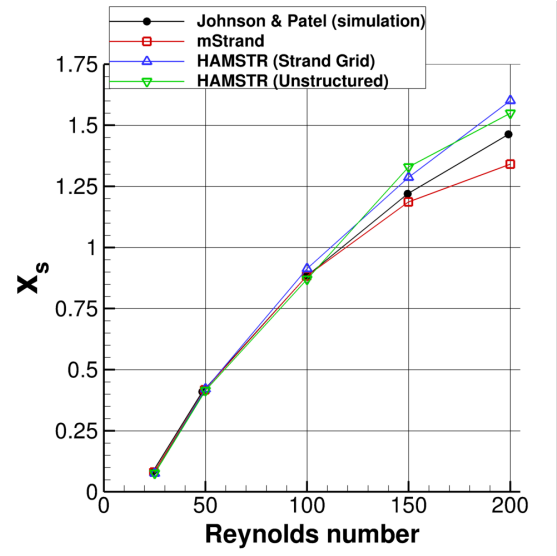



(c) Current results using the unstructured grid

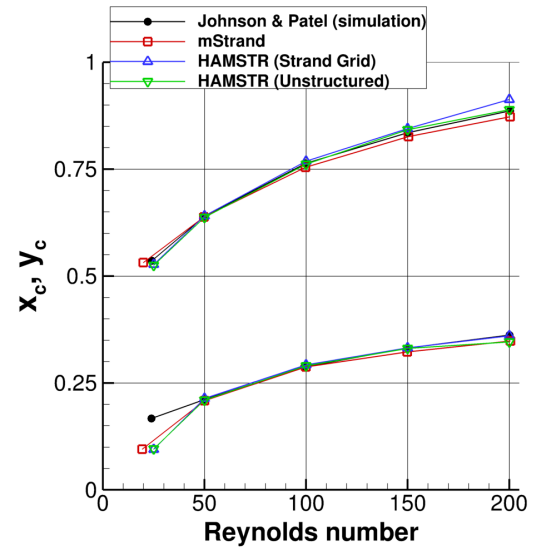
Figure 4.13: Streamline showing separation bubble for flow over sphere.



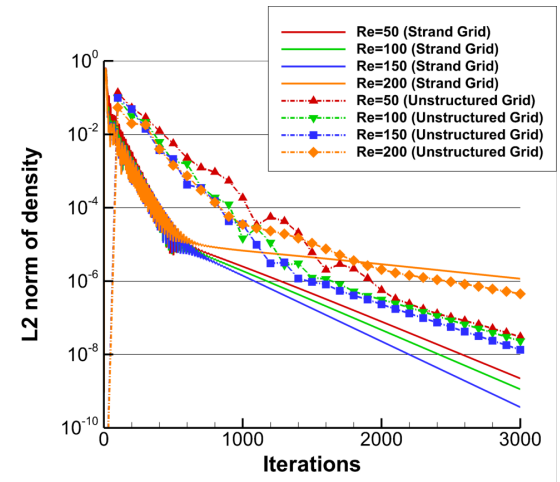

(a) Drag coefficient




(b) Separation distance

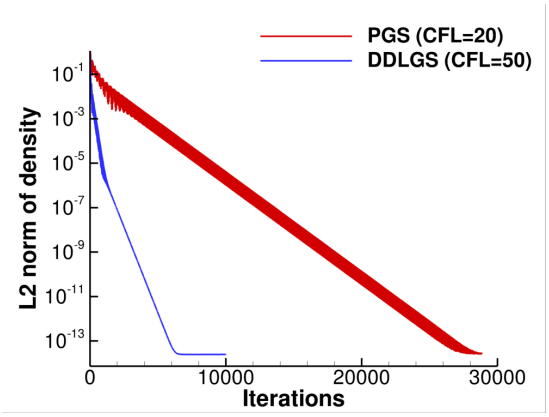



(c) Center of separation bubble

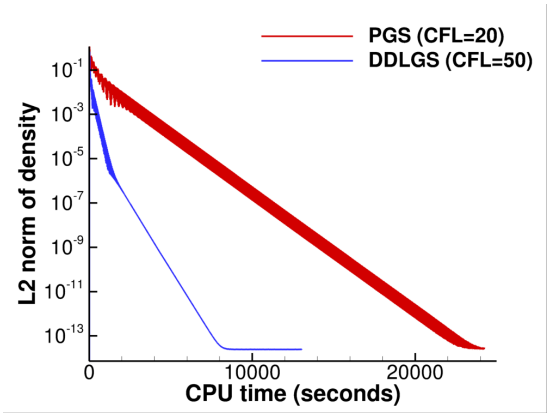



(d) Solution convergence rate from HAMSTR

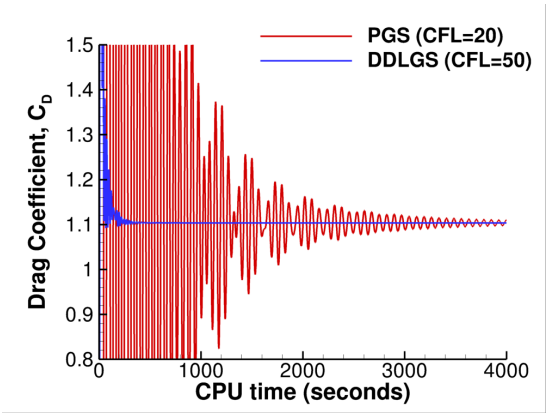
Figure 4.14: Comparison results of laminar flow over sphere with references [13, 47].




(a) Residual convergence per iteration

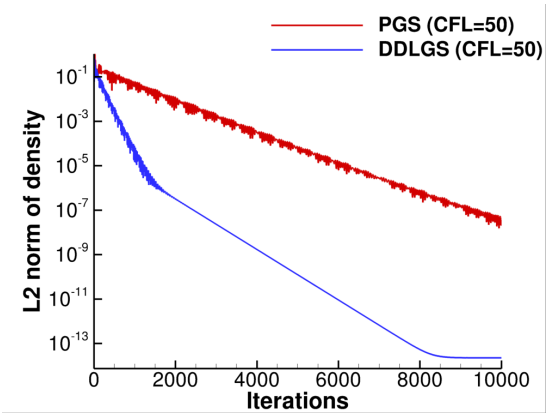



(b) Residual convergence per CPU time

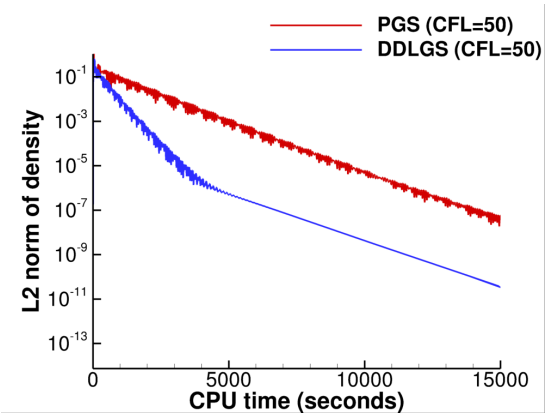



(c) Drag Coefficient per CPU time

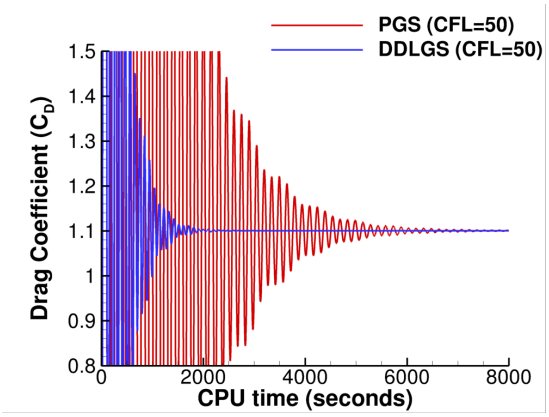
Figure 4.15: Solution convergence rate using the strand grid for laminar flow over a sphere at $Re = 100$.




(a) Residual convergence per iteration




(b) Residual convergence per CPU time




(c) Drag Coefficient per CPU time

Figure 4.16: Solution convergence rate using the unstructured volume grid for laminar flow over a sphere at $Re = 100$.

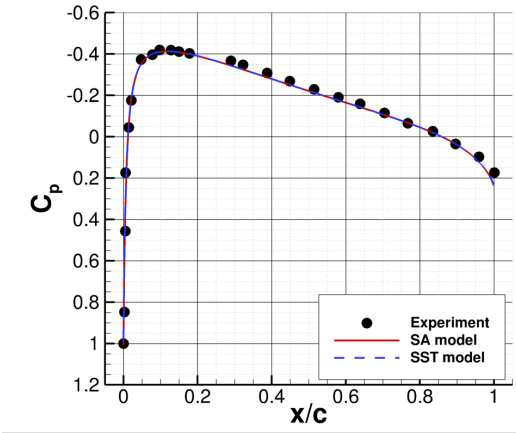
### 4.2.3 Fully Turbulent Flow Past NACA0012 Airfoil

#### Solution Accuracy Analysis

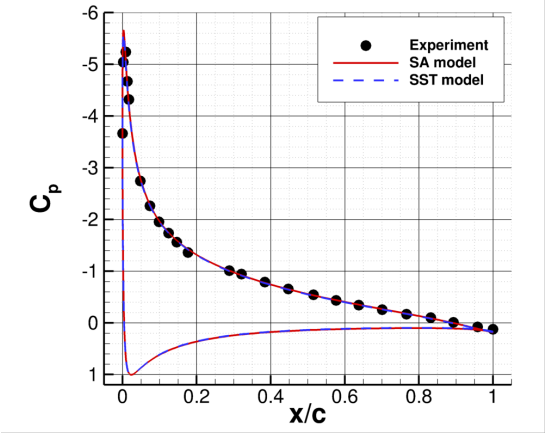
Turbulent flow past a NACA0012 airfoil is computed at a freestream Mach number of 0.15, a Reynolds number of 6 million, and three angles of attack (0, 10, and 15 degrees). To validate the turbulence models implemented in HAMSTR, both Spalart-Allmaras one-equation turbulence model and Menter Shear Stress Transport two-equation turbulence model are used. Current predictions are compared against the results provided by the NASA Turbulence Modeling Resource (TMR) [44]. A structured airfoil C-type mesh ($897 \times 257$) is used for current simulation which is provided by TMR website and the same mesh is used for the other reference simulation results. The mesh has fine enough wall normal spacing at wall of $1 \times 10^{-6}$ chord which corresponds to $y^+ = 0.25$, and the far field extends out to 500 chords. For this structured grid, WENO reconstruction scheme is used in current simulation.

Figure 4.17 (a)-(c) shows the quantitative distribution of the pressure coefficient on the upper and lower surfaces of the airfoil at various angles of attack. For the pressure coefficient, the experimental results [48] could be obtained and compared at only the upper surface of the airfoil. The measurements were obtained by tripping the flow to make it fully turbulent. The current predictions using both turbulence models show excellent agreement in all cases. Figure 4.17 (d) shows the drag histories during current simulations for all cases. All of the drag coefficient values are converged within 10,000 iterations using the same CFL number of 30. The

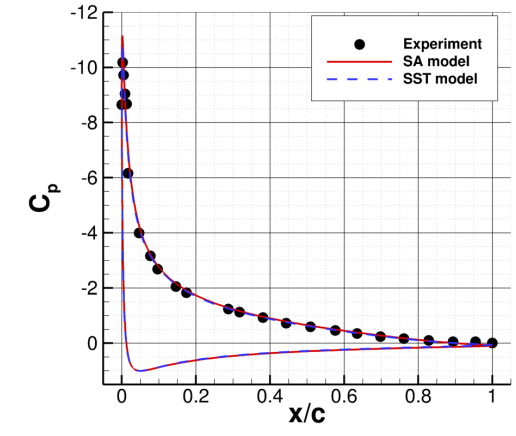
convergence rate of the values is not affected much by the different flow conditions and turbulence models.



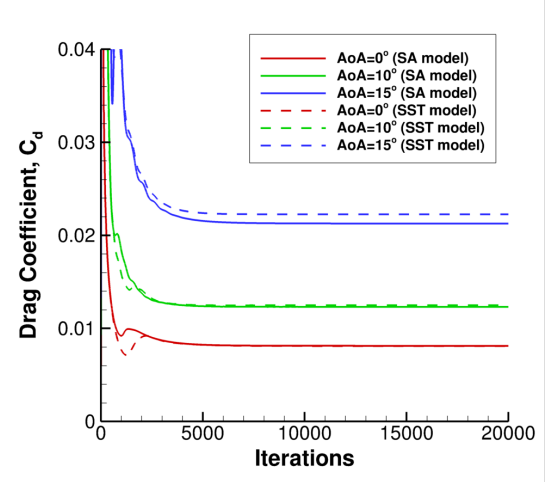

(a) At $\alpha = 5°$




(b) At $\alpha = 10°$




(c) At $\alpha = 15°$




(d) Drag coefficient history

Figure 4.17: Pressure coefficient comparison for NACA0012 airfoil simulation.

NASA TMR website provides the lift and drag coefficients which are predicted by several well-established flow solvers using either the SA turbulence model or the SST turbulence model for the considered angles of attack. As shown in Table 4.3, the current predictions using HAMSTR are compared with the values from the other

flow solvers in the case of the SA turbulence model. The relatively recent prediction from mStrand is also appended to it, which is extracted from the reference [13]. The force coefficients predicted by HAMSTR are comparable to the results predicted by all of the other flow solvers. Table 4.4 shows the comparison of the lift and drag coefficients using the SST turbulence model. Current predictions show a good agreement with other simulation results as well. Similar trend in the predictions between the two turbulence models are predicted with other flow solvers. For example, the SST turbulence model predicts lower lift coefficients and higher drag coefficients compared to the predictions from the SA turbulence model at positive angles of attack.

Table 4.3: Lift and Drag coefficient comparison using SA turbulence model.

| Codes | $C_l$ ($\alpha = 0°$) | $C_l$ ($\alpha = 10°$) | $C_l$ ($\alpha = 15°$) | $C_d$ ($\alpha = 0°$) | $C_d$ ($\alpha = 10°$) | $C_d$ ($\alpha = 15°$) |
|---|---|---|---|---|---|---|
| CFL3D | approx 0 | 1.0909 | 1.5461 | 0.00819 | 0.01231 | 0.02124 |
| FUN3D | approx 0 | 1.0983 | 1.5547 | 0.00812 | 0.01242 | 0.02159 |
| TURNS | approx 0 | 1.1000 | 1.5642 | 0.00830 | 0.01230 | 0.02140 |
| mStrand | approx 0 | 1.0967 | 1.5621 | 0.00804 | 0.01251 | 0.02195 |
| **HAMSTR** | approx 0 | 1.0907 | 1.5459 | 0.00812 | 0.01232 | 0.02127 |

## Solution Convergence Analysis

The current solution convergence rate for the turbulent flow over a NACA0012 airfoil is compared with TURNS at 10 degrees angle of attack as a representative flow condition. Both flow solvers use the same C-type mesh (449x129) which is

Table 4.4: Lift and Drag coefficient comparison using SST turbulence model.

| Codes | $C_l$ ($\alpha = 0°$) | $C_l$ ($\alpha = 10°$) | $C_l$ ($\alpha = 15°$) | $C_d$ ($\alpha = 0°$) | $C_d$ ($\alpha = 10°$) | $C_d$ ($\alpha = 15°$) |
|---|---|---|---|---|---|---|
| CFL3D | approx 0 | 1.0778 | 1.5068 | 0.00809 | 0.01236 | 0.02219 |
| FUN3D | approx 0 | 1.0840 | 1.5109 | 0.00808 | 0.01253 | 0.02275 |
| NTS | approx 0 | 1.0765 | 1.5100 | 0.00809 | 0.01251 | 0.02187 |
| **HAMSTR** | approx 0 | 1.0771 | 1.5057 | 0.00810 | 0.01248 | 0.02225 |

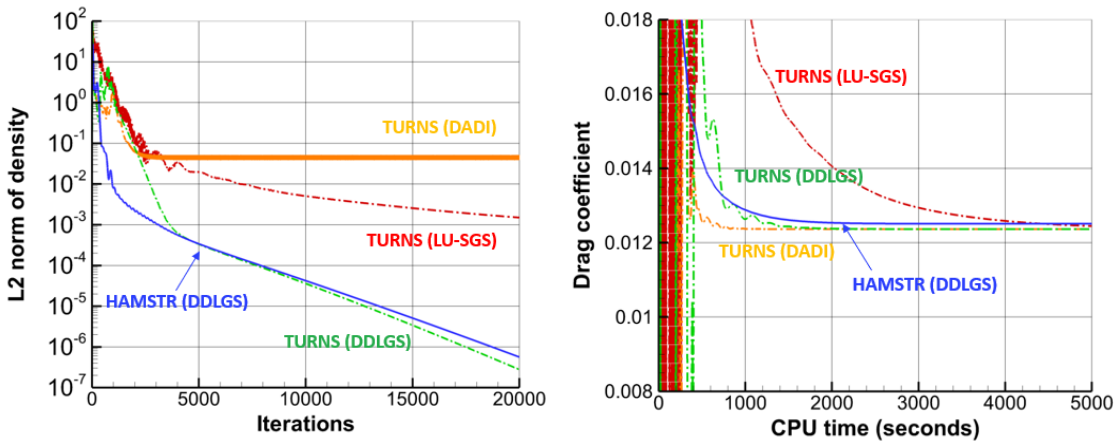
provided by TMR [44]. TURNS requires an averaged boundary condition along the *coordinate cuts*. For the reconstruction, the same WENO methods were used in both flow solvers.

In Fig. 4.18 (a), the solution convergence rate of the current DDLGS method is compared with the approximate inversion methods in TURNS per iteration: LU-SGS, DADI (Diagonalized ADI), and DDLGS. The same CFL number of 30 is used in all methods for the current comparison. It is evident that the current DDLGS method shows faster convergence per iteration than the other line-implicit methods on TURNS (LU-SGS and DADI). When the same DDLGS method on TURNS is compared with the current result, very similar convergence rates are observed between each other. It should be noted that only the DDLGS method on TURNS uses the same linearization of the Roe flux for the flux Jacobian as HAMSTR does.

Table 4.5 compares the CPU time per iteration for the various line-implicit methods on both TURNS and HAMSTR. First, DDLGS method in HAMSTR requires similar CPU time per iteration with DDLGS method in TURNS. The reason for the small difference (about 10 %) is that the HAMSTR flow solver is programmed

based on an unstructured grid data format, thus the access to the structured grid data cannot be exactly the same as with the structured grid flow solver. For example, $i, j$ indices of the structured grid data are not utilized in HAMSTR. This can make difference especially in the viscous flux computation which requires gradient evaluations not only in the streamwise direction but also in the crossterm direction.

Second, LU-SGS and DADI in TURNS require slightly less execute time per iteration than the DDLGS method on both flow solvers. The operators themselves are more efficient than for the DDLGS method and the use of spectral radius approximation for the flux Jacobian can save further CPU time.




(a) Residual convergence per iteration (b) Residual convergence per CPU time

Figure 4.18: Solution convergence comparison with TURNS.

### 4.2.4 Fully Turbulent Flow Past NACA0015 Wing

Fully turbulent flow over a NACA0015 wing is simulated for the accuracy and solution convergence validation of a three dimensional flow solver. While the experiments [49] were performed using a half-span wing, the numerical prediction

Table 4.5: Comparison of CPU execution time per iteration for different implicit inversion methods.

| Scheme | Ratio |
|---|---|
| TURNS (LU-SGS) | 1.0 |
| TURNS (DADI) | 1.03 |
| TURNS (DDLGS) | 2.14 |
| HAMSTR (DDLGS) | 2.35 |

using HAMSTR is conducted for a full-span wing. The wing aspect ratio is 6.6 (measured tip-to-tip) and the wing contains a rounded tip cap at both ends. To be compared with the predictions using structured grid flow solver (OVERTURNS [46]), an O-O type mesh ($195 \times 143 \times 96$) is used for the case. The initial wall normal spacing is $5 \times 10^{-6}$ chord which corresponds to $y^+ = 0.2$, and the far field extends out to 20 chords. Flow solutions are obtained for an angle of attack of 12° at a Reynolds number of $1 \times 10^6$ and Mach number of 0.21. The solution is executed in parallel using 16 processors with both HAMSTR and OVERTURNS. For the reconstruction scheme, both flow solvers use the fifth-order WENO scheme.

Figure 4.19 shows the surface pressure distributions at three spanwise locations near the wing tip (i.e. y/span = 0.676, 0.824, 0.971). The simulation results from both HAMSTR and OVERTURNS show reasonable agreement with the experimental data at all three sections. Both simulation results are exactly aligned with each other at y/span=0.676 and 0.824, otherwise the result from HAMSTR shows better agreement with experiment at y/span=0.971. This is because the wing tip vortex at the tip is better preserved in HAMSTR. It should be noted that HAMSTR is able

to use the same reconstruction method over the whole domain, otherwise OVERTURNS used the wake average boundary condition along the *coordinate cuts* at the wing tip.

Figure 4.20 shows the comparison of solution residual convergence in terms of both iteration and CPU time. For the comparison, DDLGS on HAMSTR is compared with LU-SGS on OVERTURNS at the same CFL number of 50. Similar to the results of the two-dimensional domain, DDLGS on HAMSTR shows faster convergence rates than LU-SGS on OVERTURNS as shown in Fig. 4.20 (a). Current DDLGS method on HAMSTR requires about 2.65 more CPU times per iteration than LU-SGS method on OVERTURNS for the three-dimensional flow simulation, which is increased by about 10 % from the results of the two-dimensional flow simulation (see Table 4.5). As a result, comparable solution convergence rates in terms of CPU time are observed between the solvers as shown in Fig. 4.20 (b). HAMSTR shows slightly better convergence at the end of the simulation, as TURNS shows evidence of the convergence stalling (possibly due to wake average boundary condition).

## 4.3 Validation of Overset Method

The overset capability has been tested and validated using the case of two spheres placed side-by-side to predict interactions between the bodies. The freestream Mach number is 0.2 and the flow is laminar at a Reynolds number of 100. The angle of attack between the two spheres is 0°. Figure 4.21 shows the geometrical place-

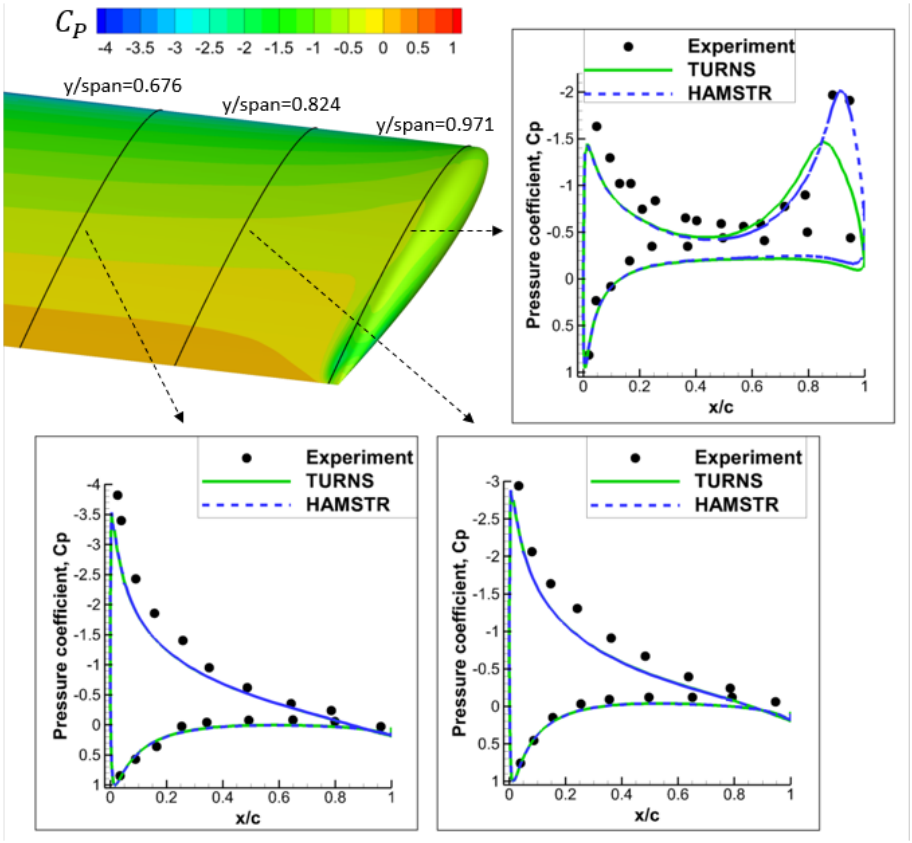
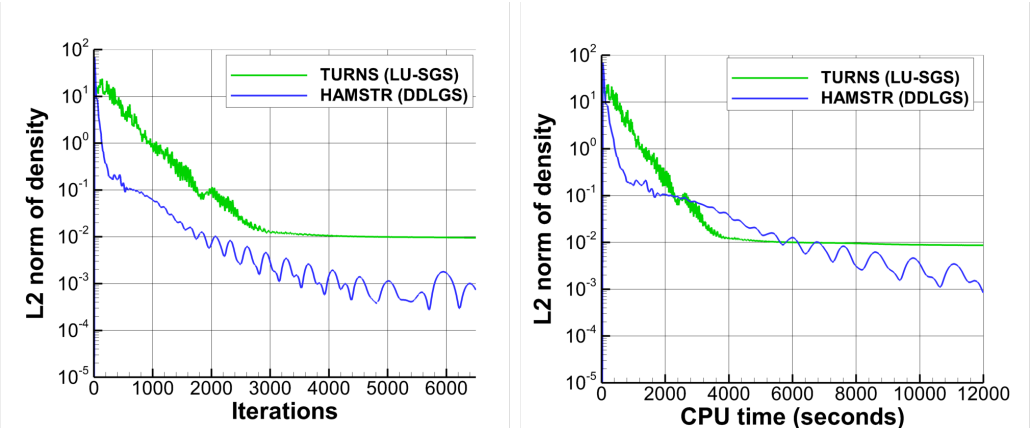



Figure 4.19: Sectional surface pressure comparison for NACA0015 wing simulation.




(a) Residual convergence per iteration (b) Residual convergence per CPU time

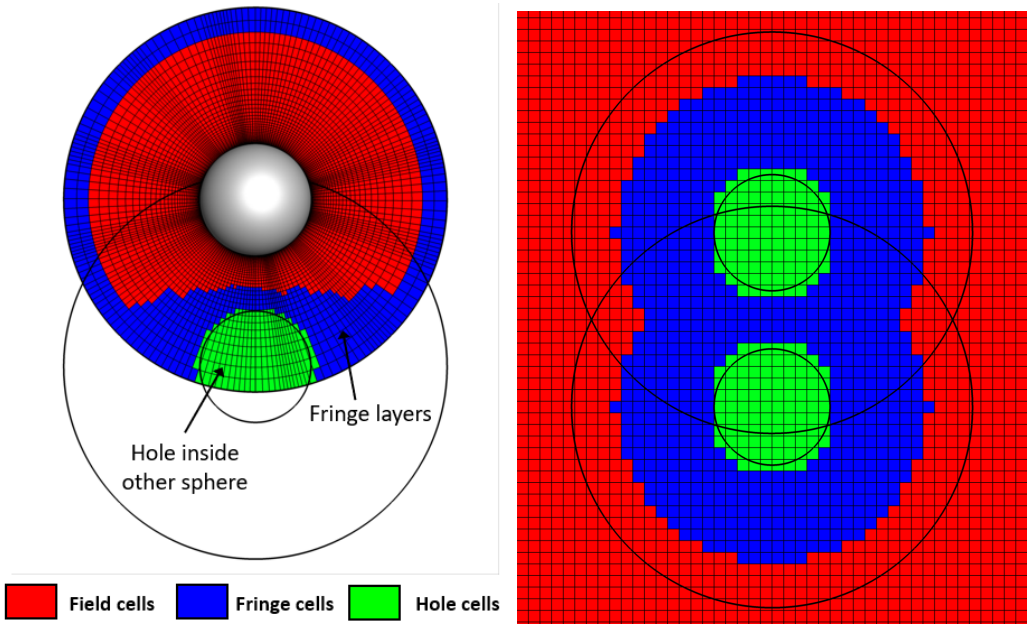
Figure 4.20: Solution convergence comparison with OVERTURNS for NACA0015 wing simulation.

ment of the two spheres relative to each other. The sphere centers are separated by 1.5 sphere diameter.

The surface mesh of the sphere is obtained after subdividing the 5,120 isotropic triangles to 15,360 quadrilaterals using quad-level 0. The initial wall normal spacing is 0.1% of the diameter and 34 strands layers are used for the volume domain. Then, each near-body sphere domain is connected with an off-body Cartesian domain using TIOGA.

Figure 4.21 shows the *iblank* map of the top sphere and the background domain. Each cell is classified into one of three categories; field cell, fringe cell, and hole cell as stated in section 3.5. It should be noted that the number of fringe layers is determined based on the reconstruction scheme stencil. For example, three fringe layers are necessary for the WENO reconstruction scheme. In the top sphere domain, the fringe cells are modified according to the hole cells which are within the bottom sphere as shown in Fig. 4.21 (a). In the off-body domain, the hole cells inside the dual spheres are recognized and the fringe cells are tagged around the hole cells as shown in Fig. 4.21 (b).

Figure 4.22 (a) shows the density contour around the two spheres along the longitudinal plane with the flow coming from the left of the figure. The black lines show the overset boundaries of each near-body and off-body domains. It should be noted that the line contour is shown only in the field cell region (*iblank*=1), thus there are a few void regions in the figure. Overall, smooth line contours are observed across the overset boundaries between either the two near-bodies or the near-body and off-body.



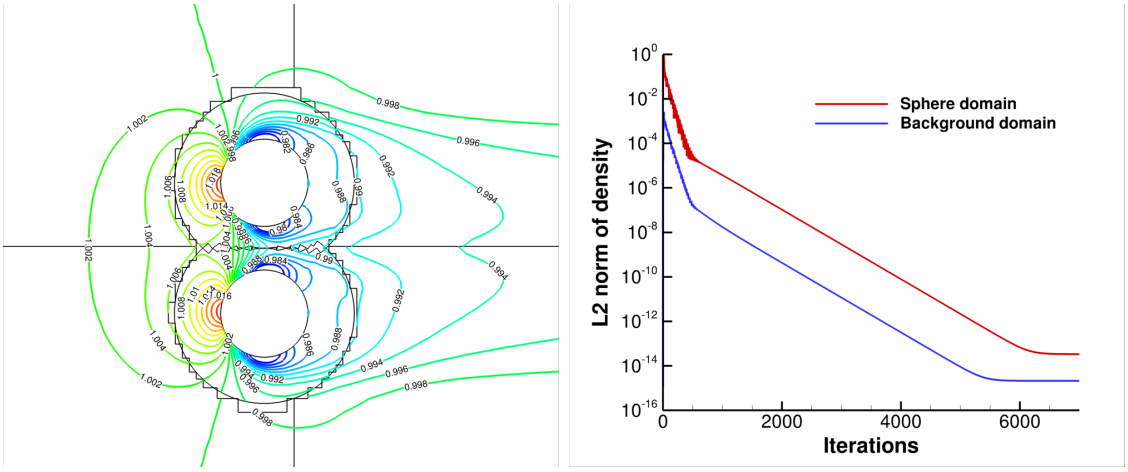

(a) Near-body sphere domain (b) Off-body Cartesian domain

Figure 4.21: *iblank* map for dual sphere nearbody and Cartesian offbody.

The solution residual convergence from each near-body and off-body domains are shown in Fig. 4.22 (b). In both domains, the DDLGS method is used for implicit inversion with CFL of 50. The inviscid fluxes are reconstructed using a combined least-square in the wall-tangential direction and WENO scheme in the strand direction (LLS-WENO). The solution converges to machine precision in the both domains after approximately 6,000 iterations. It should be noted that the current convergence rate of the near-body domain is very similar with the convergence of the single sphere mesh without overset from section 4.2.

In Fig. 4.23, the current surface pressure and skin friction distributions are compared with other simulation results using a structured finite different based Navier–Stokes solver [50]. Due to the interference between the two spheres, asymmetric results were observed between the top and bottom of the surface at 0° angle

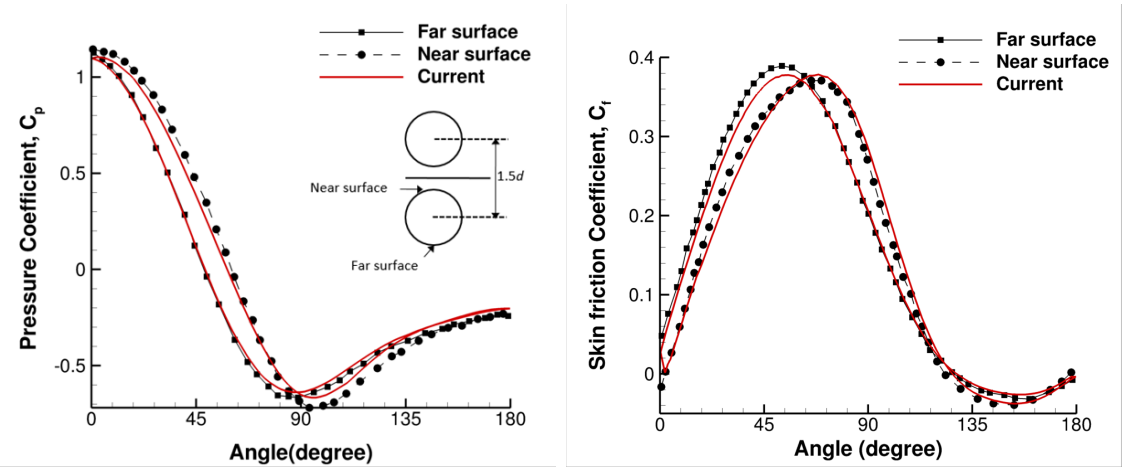
of attack. Although they are not exactly aligned with each other, reasonable agreements are observed between current and reference results by commonly showing a few features. First, the pressure on the near surface is higher than the pressure on the far surface before the minimum pressure occurs. Second, the minimum pressure occurred at an angle greater than 90° on the near surface and the minimum pressure is lower than the minimum pressure on the far surface. Third, the skin friction on the far surface is higher than the near surface before the maximum friction occurs and the trend is opposite after the maximum friction occurs. Fourth, the separation point where the skin friction becomes negative is slightly behind on the near surface than the far surface. The slight differences with the reference results could be attributed to slight differences in the solution strategies or wall normal grid spacing.




(a) Density contour for dual sphere. (black line: domain boundary)

(b) Solution residual convergence from top sphere domain

Figure 4.22: Simulation results for dual sphere case.




(a) Surface pressure distribution

(b) Skin friction distribution

Figure 4.23: Comparison of simulation results for dual sphere.

## 4.4 Summary

In this chapter, the solution accuracy and residual convergence rate are evaluated through canonical problems as validations. The implementations in the flow solver are also validated by comparing the results with well-established other flow solvers. The validations are performed for both two- and three-dimensional flow solvers.

- The accuracy of stencil- and gradient-based reconstruction methods have been evaluated through Method of Manufactured Solution (MMS). On a unstructured surface mesh, the gradient-based method shows its formal order of accuracy of second-order, otherwise the stencil-based methods limit its accuracy to first-order due to varying cell size and curvature along the loop.

- The accuracy of different stencil-based reconstruction schemes are compared

with each other through an isentropic vortex simulation on a unstructured grid: MUSCL, WENO, and CRWENO. It is observed that the higher-order type method provides less dissipation error than the lower-order method.

- The efficiency of line-implicit method (DDLGS) is validated through comparing with a point-implicit method (point Gauss-Seidel, PGS) on unstructured grids. Higher efficiency is observed in three-dimensional viscous flow simulations than a two-dimensional inviscid flow simulation, becuase the line-implicit method is applied to not only the wall-tangential directions but also the wall-normal direction.

- The performance of the current flow solver is validated through comparison studies with in-house structured grid based flow solver, OVERTURNS. The comparable solution residual convergence was obtained for both two- and three-dimensional problems in terms of executive CPU time.

- The implementations of both SA and SST turbulence models are validated through two cases: two-dimensional bump in channel and NACA0012 airfoil. Using the same structured grid with references, detailed comparisons are performed such as grid refinement study.

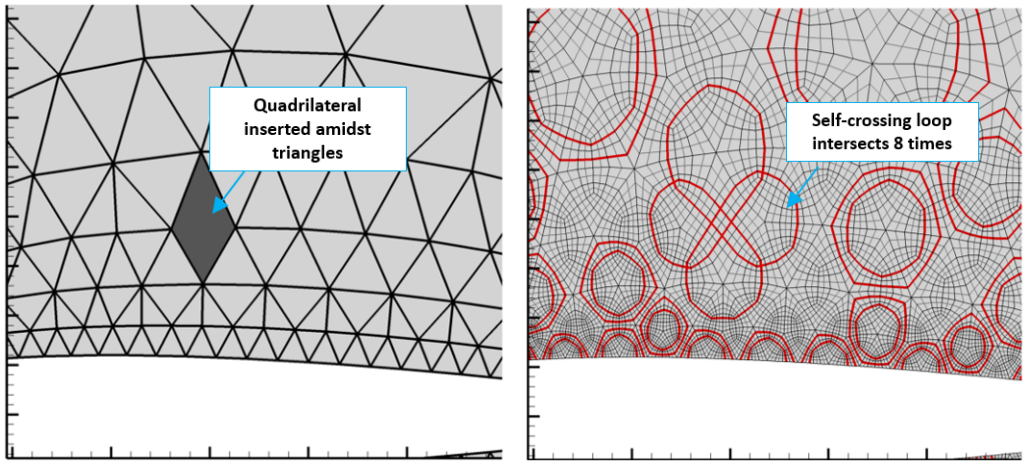
- The overset capability is validated using a case of two spheres placed side-by-side. The interactions between the bodies are captured similarly as the referenced result.

# Chapter 5: Solution Convergence

The preformance evaluations are continuously conducted in this chapter. First, the numerical performance is studied in the case with self-crossing loops. Second, the implementation of precondition GMRES method is validated. Solution convergence rates are compared with the results from a pure line-implicit method for various problems at different flow conditions. Finally, the parallel efficiency of the flow solver is evaluated using a strong scalability test.

## 5.1 Loop Crossing

The Hamiltonian path can cross itself when the paths are generated on unstructured grid with initially mixed elements of triangles and quadrilaterals. It is recognized that one of the ways that Hamiltonian loops can be forced to cross each is by inserting a quadrilateral in the middle of a purely triangular mesh. Figure 5.1 shows an example of such a self-crossing loop by inserting a quadrilateral element on a pure triangle mesh before quadrangulation for a NACA0012 airfoil. After quad-level 1 subdivision, it is observed that two Hamiltonian loops pass through this modified region and self-intersect eight times each. It should be noted that the number of intersections is dependent on the subdivision level. The formulation of the

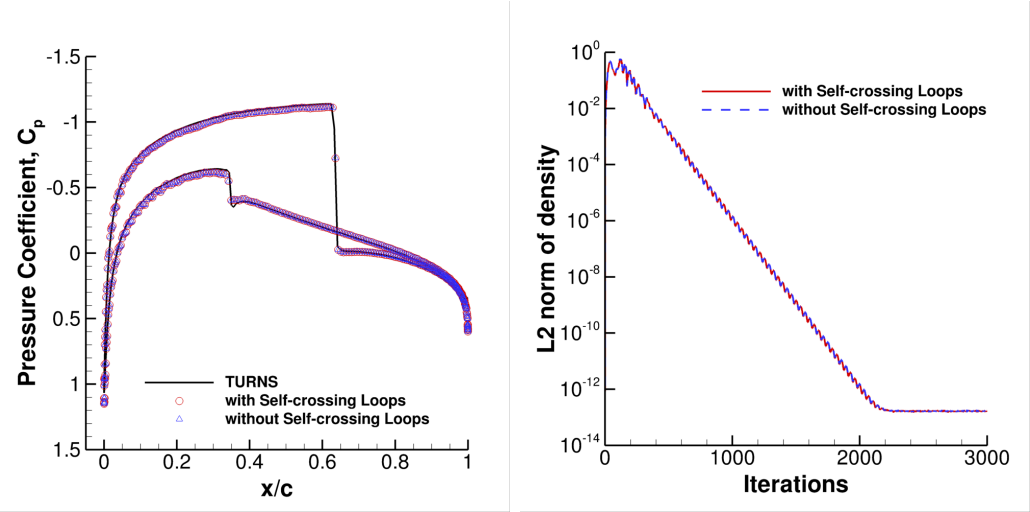



(a) Quadrilateral in a initial triangular mesh (b) Self-crossing Hamiltonian path

Figure 5.1: Self-crossing loops on unstructured mesh for NACA0012 airfoil.

solver can handle this phenomenon without modifications, thus both stencil-based reconstructions and line-implicit methods are still feasible along the self-crossing loop. The simulation results of transonic flow (M=0.8 and AoA=1.25°) are compared between the meshes with the self-crossing loops and without the self-crossing loops as shown in Fig. 5.2. MUSCL reconstruction for the inviscid flux computation and DDLGS for the implicit inversion are used at a CFL number of 40. It is observed that the solution is not affected by the existence of the self-crossing loops above the airfoil, thus very similar surface pressure and solution residual convergence rates are obtained.

The self-crossed loops can also be found when the Hamiltonian paths are constructed on a semi-regular quadrilateral surface (see section 2.1). Thus, a further convergence study with self-crossing loops is conducted for the fully turbulent flow simulation around the Robin-Mod7 fuselage at a flow condition given by: freestream Mach number of 0.1, Reynolds number of 1.6 million based on the fuselage length




(a) Surface pressure distribution (b) Solution residual convergence

Figure 5.2: Comparison of simulation results between with and without self-crossing loops for transonic flow over airfoil.

and 0° angle of attack.

As shown in Fig. 5.3, two different surface meshes are used for the comparison. For the unstructured surface mesh, a total 25,152 quadrilaterals are generated on the surface from the initial 2,096 triangles using quad-level 1 subdivision method. For the semi-regular surface, a total 35,248 quadrilaterals are generated from the initial 2,203 quadrilaterals using the same quad-level 1 subdivision method. From each surface domain, the volume domain of the fuselage is generated using 57 strand layers in the wall-normal direction. The initial wall-normal spacing is $5 \times 10^{-6}$ fuselage length, which corresponds to a $y^+ = 0.3$. The near-body domain is connected with the Cartesian background domain using the overset method. Each near-body domain is partitioned using either 21 processors or 30 processors, which results in a similar number of elements per processor: 67,000 cells for the unstructured mesh

and 66,000 cells for the semi-regular mesh. WENO reconstruction is used for the inviscid fluxes and the DDLGS method is used for the implicit inversion at a CFL number of 30. For the fully turbulent flow assumption, the SA turbulence model is used.

Figure 5.3 (a) and (b) show a portion of the resultant paths on the Robin-Mod7 fuselage surface, from either an unstructured surface mesh or a semi-regular mesh, which are overlaid on surface pressure contours. The resultant loops on the semi-regular mesh are self-crossed as shown in Fig 5.3 (b), otherwise the loops have a circle-shape without self-crossing on the unstructured mesh as shown in Fig 5.3 (a). In the case of the semi-regular mesh, a total 19 loops are generated on the surface where the shortest loop contains 65 cells and the longest loop contains 13,569 cells. In the case of the unstructured mesh, a total 2,099 loops are generated where the shortest loop contains 21 cells and the longest loop contains 29 cells. Clearly, the resultant paths for the semi-regular mesh are quite different in length compared to the loops on the unstructured mesh.

The variations of pressure coefficient along the upper and lower surface of the fuselage in longitudinal plane are compared as shown in Fig. 5.3 (c). Current predictions are also compared against results from experiment and OVERFLOW [45]. Overall, it shows reasonable agreement with the reference results. However, the both simulation results predict a smaller pressure peak than experiment at the ramp region because the current results and OVERFLOW result are obtained in the free-air condition without modeling of the wind tunnel walls. When the current results are compared to each other, the result from the semi-regular mesh has a smoother curve

after the ramp region than the result from the unstructured mesh although they are quite similar to each other.

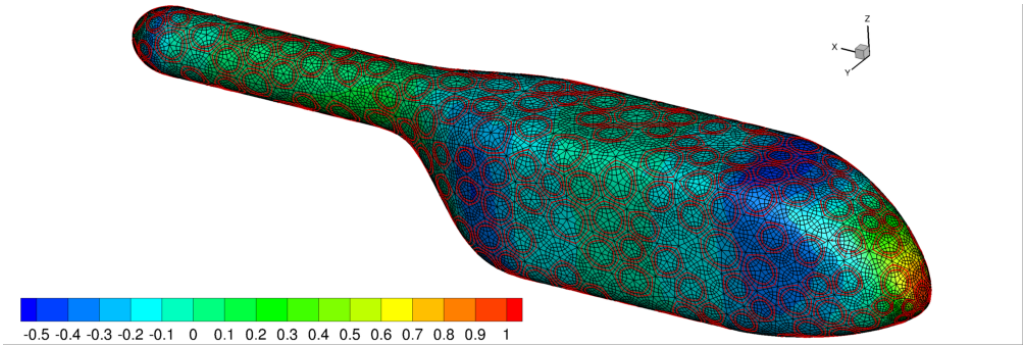
Then, the solution residual convergence rates are compared in terms of both per iteration and CPU time as shown in Fig. 5.4. In both simulations, the solution residual drops by 7 orders of magnitude within 20,000 iterations. Although, a slightly better solution convergence rate is observed using the unstructured surface mesh than the semi-regular one, the difference is not significant.

## 5.2 Solution Convergence of Preconditioned GMRES Method

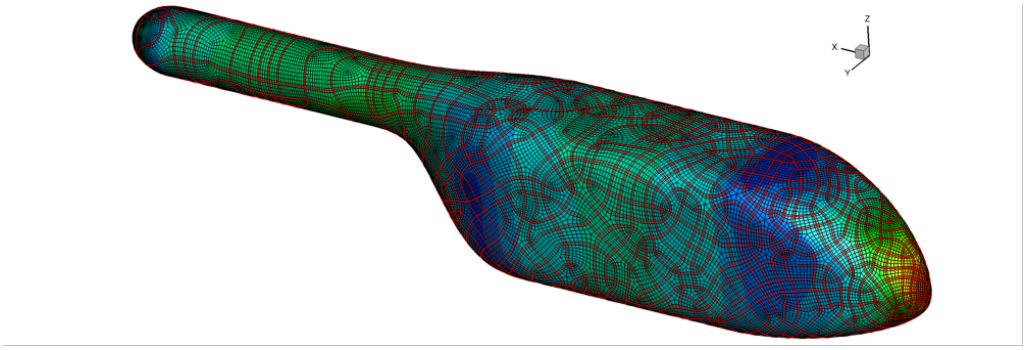
The use of high-order reconstruction and high aspect ratio cells inside boundary layer usually limits the solution convergence rate with a factorized line-implicit method, which is because the convergence of Jacobi or Gauss-Seidel type methods is affected by the size of stiffness of the flux Jacobian matrix. GMRES, based on Krylov subspace methods, can be an option to improve convergence rates with high CFL values. In this section, GMRES preconditioned using DDLGS is used and the solution residuals are compared against the line-implicit method alone (DDLGS).

### 5.2.1 Overset Lifting Rotor

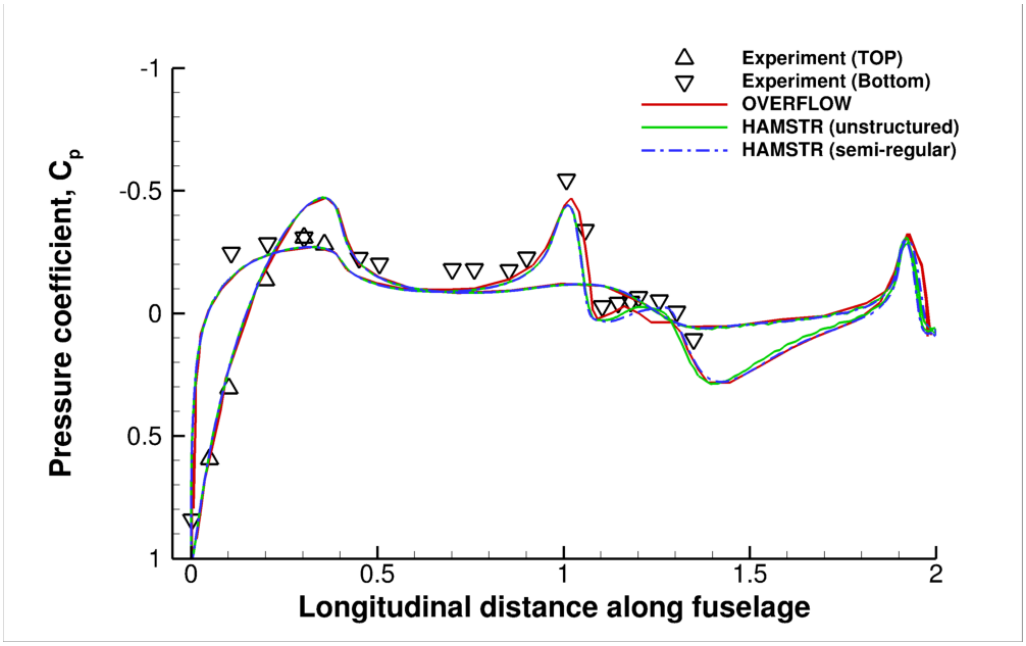
Inviscid flow around the Caradonna-Tung rotor is computed using an overset mesh system, which is a two-bladed rotor with an aspect ratio of 6 [53]. Among the various hovering flight conditions in the experiment, a transonic blade tip Mach number of 0.866 and a collective of 8° is simulated.




(a) Hamiltonian paths without self-crossing on unstructured surface mesh

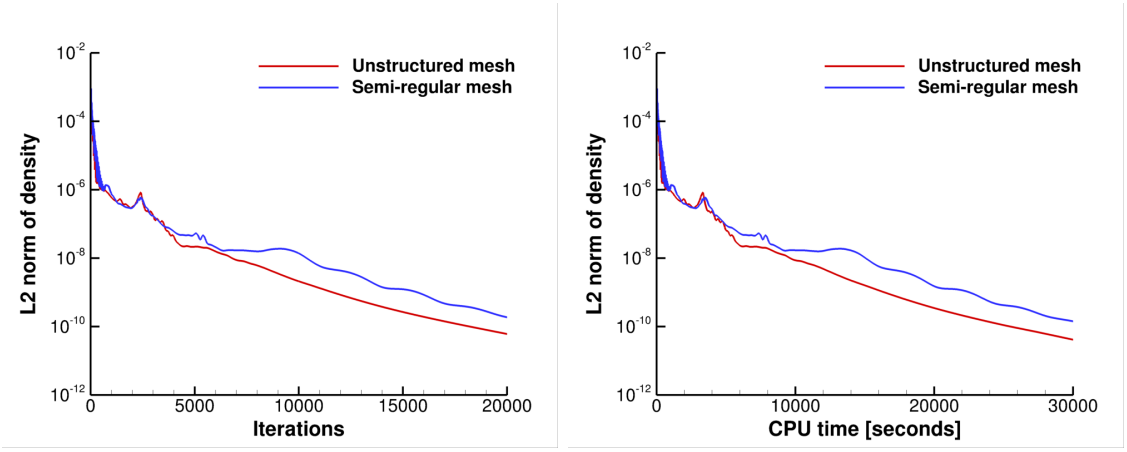


(b) Hamiltonian paths with self-crossing on semi-regular surface mesh




(c) Surface pressure distribution on longitudinal plane ($y = 0$)

Figure 5.3: Hamiltonian paths overlaid on the pressure contour for Robin-Mod7 fuselage simulation.



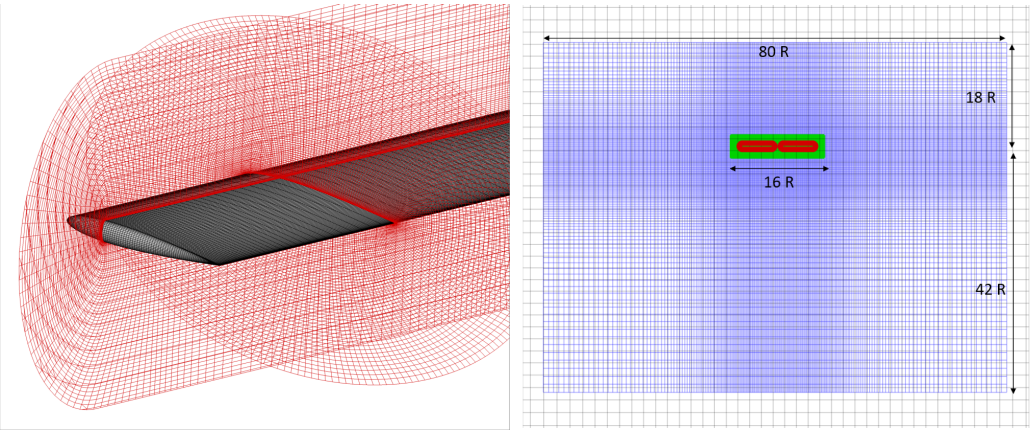

(a) Residual convergence per iterations (b) Residual convergence per CPU time

Figure 5.4: Solution convergence comparison for Robin-Mod7 fuselage simulation.

As shown in Fig. 5.5, an O-O type structured grid is used for the blades which is connected with off-body Cartesian meshes using the overset method. The blade mesh consists of 211 points in the airfoil-wrap direction, 70 points in the normal direction, and 100 points in the span-wise direction. For the inviscid flow simulation, an initial wall-normal spacing is $1 \times 10^{-3}$ chord. The off-body mesh system consists of two nested meshes and a background mesh, which are also connected with each other using the overset method. For the nearest nested mesh, uniform grid spacing of 0.1 chord is used.

The blade domain is computed using HAMSTR and the Cartesian domain is computed using a GPU-based structured grid flow solver [54] within a Python framework. The inviscid fluxes are computed using MUSCL and WENO reconstruction for each near-body domain and off-body domain, respectively. The solution is evolved using time-accurate BDF2 method with sub-iterations.

As shown in Fig. 5.6, the solution convergence rates are compared between




(a) Structured blade mesh ($211 \times 100 \times 70$) (b) Off-body Cartesian meshes with overset

Figure 5.5: Computational mesh for Caradonna-Tung hovering rotor simulation.

DDLGS and preconditioned GMRES. For the off-body domain, a Diagonalized Alternating Direction Implicit (DADI) method is used during the test. The simulation with GMRES is able to use a larger time step compared to the line-implicit method applied alone. Thus, a CFL number of 500 is used for GMRES, otherwise a CFL number of 50 is used for the DDLGS method. In the use of the GMRES method, two different numbers of outer iterations (1 and 3) are used with the same number of Krylov vectors of 5. It should be noted that the number of Krylov vectors of 5 is sufficient for the current inviscid flow simulation; otherwise, more Krylov vectors are typically required for viscous flow simuations.

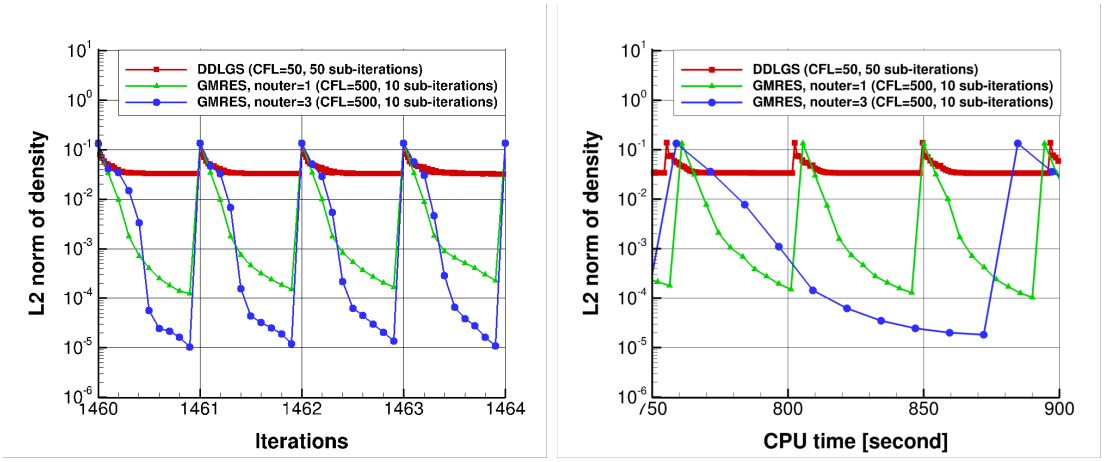
The number of sub-iterations is set as a user input and a specified tolerance is not applied to exit the iteration as soon as the solution is converged. In this study, 50 and 10 sub-iterations are used for each DDLGS and GMRES method, respectively.

In Fig. 5.6 (a), it is observed that a good convergence rate is achieved in

the GMRES results with drops of 3-4 orders of magnitude per iteration. The use of multiple outer iterations helps the residual convergence as well. Otherwise, a relatively poor convergence rate is observed using only DDLGS. The current case is numerically challenging to obtain a good convergence rate because of the flow condition: inviscid transonic flow over the blade. Without viscosity, the numerical convergence rate is typically degraded.

Every iteration of the GMRES method includes multiple preconditioning steps based on the number of Krylov vectors and outer iterations. Because one precondition corresponds to a step of the DDLGS method, the use of 5 Krylov vectors and 3 outer iterations requires 15 DDLGS calls per GMRES iteration. In Fig. 5.6 (b), the residual convergence is compared in terms of CPU time. Although the GMRES method is more expensive than the DDLGS method per each sub-iteration, the GMRES can be an alternative method if the residual is stalled using DDLGS. Typically, two order of convergence per time-step is sufficient for the applications and the results from GMRES satisfy the convergence criteria within 5 sub-iterations.

In Fig. 5.7, the sectional pressure distributions on the blade surface are extracted at spanwise locations of 50%, 68%, 80%, 89% and 96% and are compared against the experimental data. A shock appears at the 80% radial station and gets stronger at 89%. At the 96% radial station, the effect from the shock is smeared out due to the tip effect. Overall, the pressure distribution is in good agreement with those from experiment. However, the pressure distributions on the suction side of blade are somewhat over-predicted and a stronger discontinuity is observed on the predicted shock than that measured in the experiment as a result of the inviscid




(a) Residual convergence per iteration (b) Residual convergence per CPU time

Figure 5.6: Comparison of solution convergence rate for Caradonna-Tung rotor simulation.

flow assumptions. The fully turbulent flow was also simulated for this case using the current flow solver. The observed deviations disappeared in the turbulent flow simulation result, and the agreement with experimental data was improved [55]. The detailed results are not included in this section because it is beyond the scope of the current section. Although the DDLGS method shows poor convergence rate compared against the GMRES method, it is observed that the predicted pressure distribution of the blade is not affected by the less converged solution.

### 5.2.2 Unsteady Laminar Flow over a Sphere

Unsteady laminar flow over a sphere is simulated using an overset mesh system. The mesh system consists of a near-body sphere domain and an off-body Cartesian domain as shown in Fig. 5.8 (a). A strand grid is generated from the surface mesh of the sphere with the initial wall normal spacing of 0.1% of the diameter. The

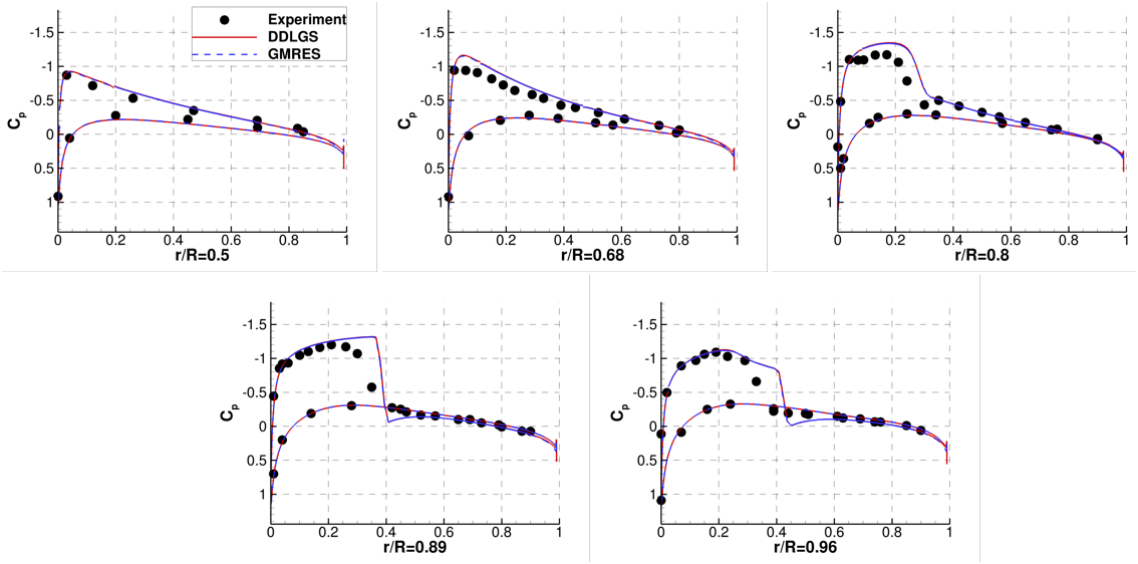
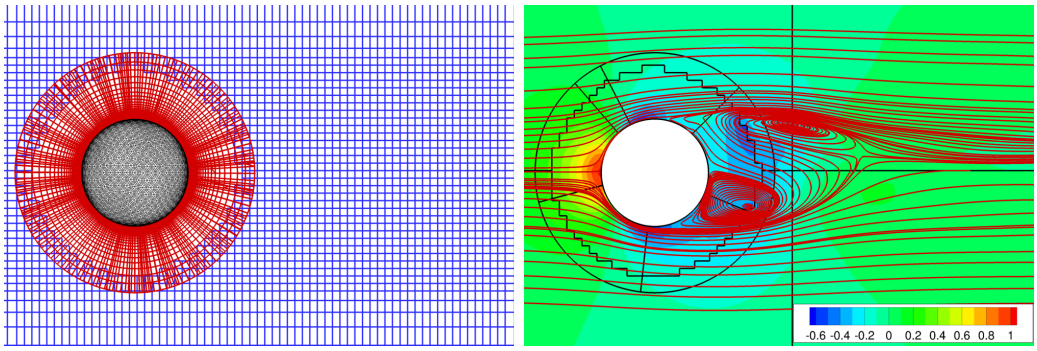



Figure 5.7: Pressure coefficient distributions on the blade at different radial stations for Caradonna-Tung rotor simulation.

nearbody domain is connected with the background domain, which has the minimum grid spacing of 0.07 diameter. Both domains are simulated using HAMSTR flow solver. The inviscid fluxes are reconstructed using the 5th order WENO scheme and either DDLGS or GMRES was used to compare the unsteady solution residual convergence rate. The solution is evolved using BDF2 method with time step size of 0.05.

The freestream Mach number is 0.2 and the Reynolds number is set to $Re = 800$ in order to ensure asymmetric wake shedding behind the sphere. Figure 5.8 (b) shows the instantaneous streamlines overlaid on the pressure coefficient contours in the $x - z$ plane. The black lines show the edge of the domains. It is observed that the vortices are located across the overset boundary.

Figure 5.9 compares the solution convergence rate between DDLGS (30 sub-



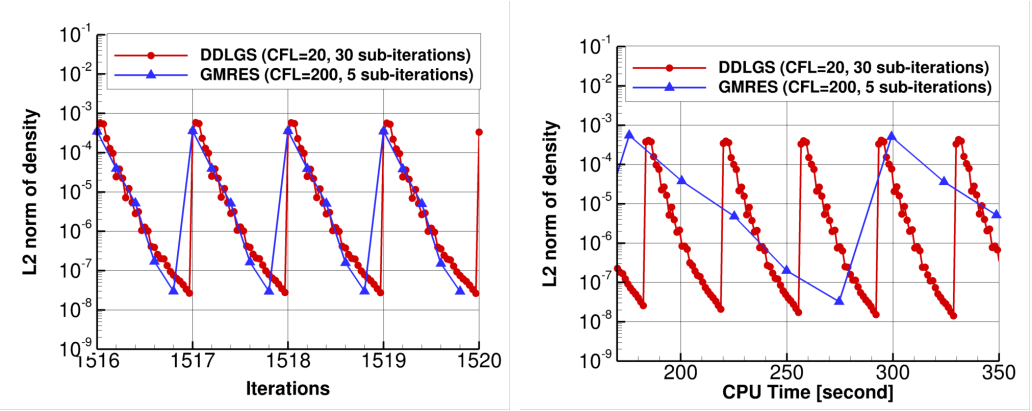

(a) Overset mesh system (b) Unsteady shedding behind a sphere

Figure 5.8: Laminar flow over a sphere simulation at $Re = 800$.

iterations) and GMRES (5 sub-iterations). The GMRES uses 5 Krylov vectors with 5 outer iterations at each sub-iteration. From both DDLGS and GMRES, the convergence rates are quite good, thus the residual drops by more than 4 orders of magnitude during the sub-iterations. Also, the residual stall does not occur in the DDLGS results, which is different from the previous result from the rotor simulation at transonic inviscid flow condition. In Fig. 5.9 (b), the convergence rate is compared in terms of CPU time. As expected, the current GMRES is more expensive than DDLGS per iteration because a total of 25 preconditions are required at every sub-iteration for the GMRES.

### 5.2.3 MD 30P-30N Airfoil

Turbulent flow over an MD 30P-30N airfoil is simulated in this section, which is a McDonnell-Douglas three-element airfoil with a 30° slat deflection and a 30° flap deflection. This configuration was the subject of an experimental study performed by Chin et. al. [56]. Also, various efforts for simulating the flow around the airfoil




(a) Residual convergence per iteration (b) Residual convergence per CPU time

Figure 5.9: Comparison of solution convergence rate for laminar flow ($Re = 800$) over sphere.

were conducted through the high-lift workshop CFD challenge at NASA Langley research center in 1993.

Figure 5.10 shows grid examples for the configuration: multi-block structured grid [57], unstructured grid, and overset structured grid [58]. In the current simulation, an unstructured grid is used which consists of all quadrilateral elements [59]. Current simulations are performed at the flow condition of Reynolds number of 9 million and the freestream Mach number of 0.2. For the fully turbulent flow assumption, the SA turbulence model is used. The inviscid fluxes are reconstructed using WENO scheme.

In this case, a grid convergence study is conducted using three different mesh resolutions. The grid information is summarized in Table. 5.1. It should be noted that the wall normal spacing is reduced by half as the level of grid refinement increases.

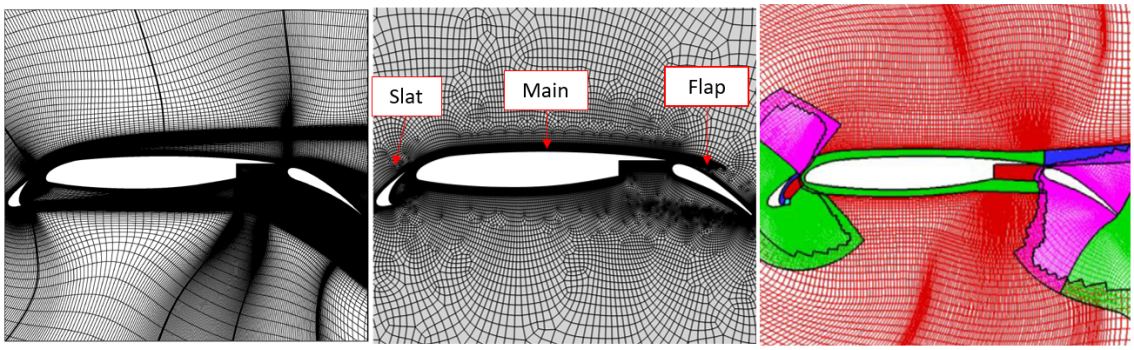


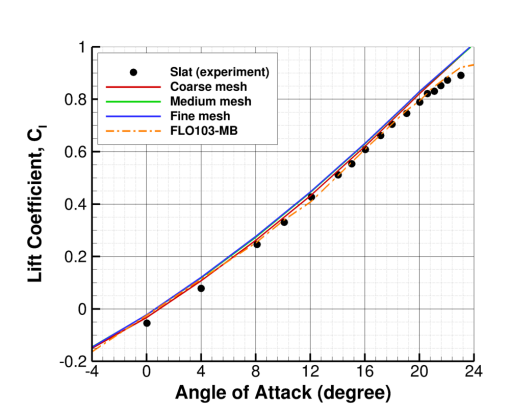

Figure 5.10: Different types of grid for MD 30P-30N airfoil.

Table 5.1: Grid information for MD 30P-30N airfoil.

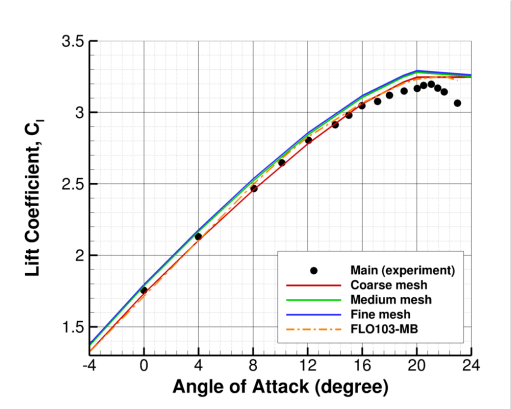
| | Coarse mesh | Medium mesh | Fine mesh |
|---|---|---|---|
| Surface grid points | 434 | 606 | 944 |
| Normal spacing ($y^+$) | $6 \times 10^{-6}$ (2.0) | $3 \times 10^{-6}$ (1.0) | $1.5 \times 10^{-6}$ (0.5) |
| Total grid elements | 56,000 | 80,000 | 124,000 |

Figure 5.11 compares the individual lift from the three components and the total lift in the range of -4 to 24 degrees angles of attack. Besides experimental data, other simulation results are also compared with the current results, which were obtained using well-established RANS solvers: FLO103-MB [57] and INS2D [60]. For the reference simulations, the multi-block structured grids were used for this configuration where the wall normal spacing was less than $2 \times 10^{-6}$. Including the current simulation, all of the simulation results are obtained using the SA turbulence model. Overall, the current results show reasonable agreement with both experimental and other simulation results. It should also be noted that the current solutions were converged over the increasing levels of grid refinement.

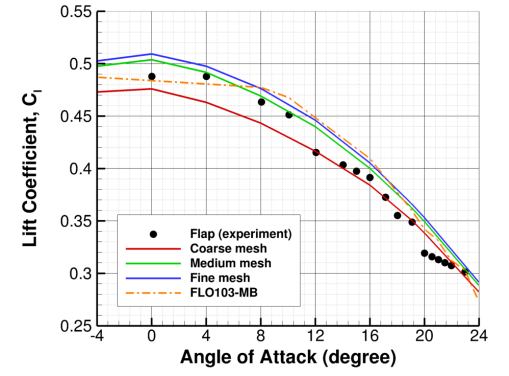
However, some deviations with experimental data are also observed in the



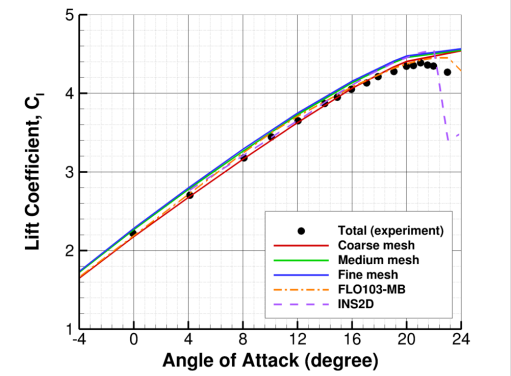

(a) Slat element




(b) Main airfoil element




(c) Flap element

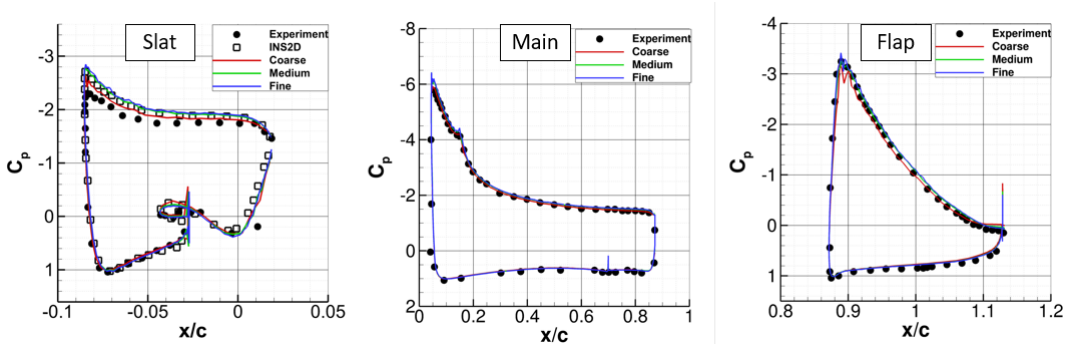



(d) Total airfoil

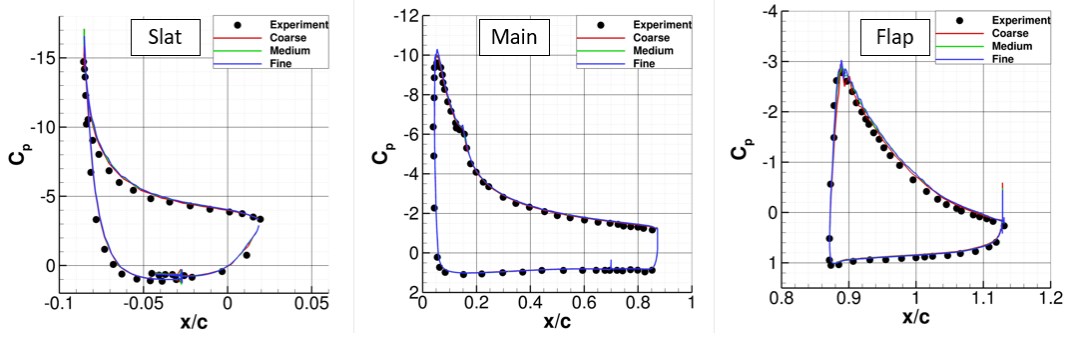
Figure 5.11: Comparison of experimental and computational lift coefficient for 30P-30N airfoil at $Re = 9 \times 10^6$.

current prediction. For example, the lift coefficient is somewhat over-predicted compared to experimental data. There are various possible sources of discrepancy: measuring error, grid quality, limitation of turbulence model, etc. Also, none of the simulation results agree with the experimental value of maximum lift. This is because the experiment starts to experience three-dimensional flow effects at the high value of lift which are ignored in the two-dimensional RANS simulations [60].

Current surface pressure predictions are compared with experimental data at




(a) At 8° angle of attack

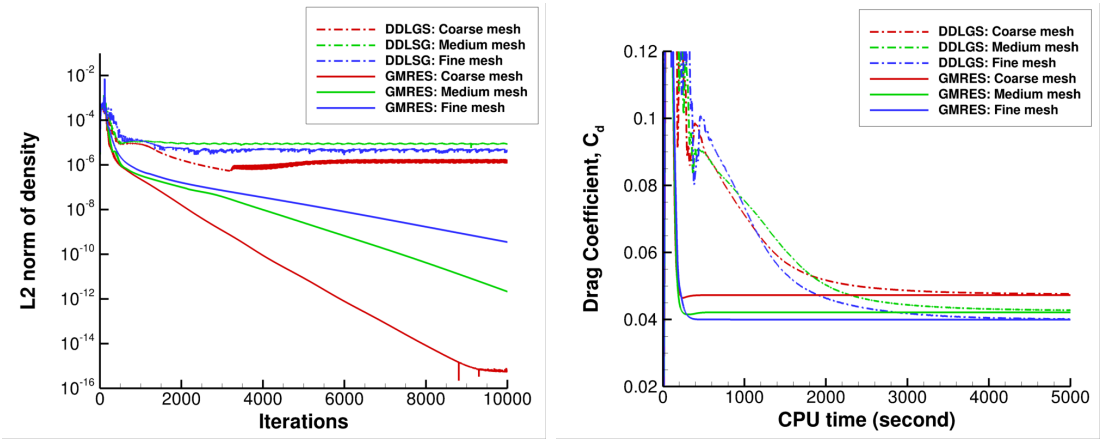



(b) At 19° angle of attack

Figure 5.12: Comparison of surface pressure distribution for MD 30P-30N airfoil at $Re = 9 \times 10^6$.

two different angles of attack of 8° and 19° as shown in Fig. 5.12. The comparison is conducted for each component at each angle of attack. In general, the agreement with experiment is good. The biggest discrepancy occurs at 8° angle of attack on the slat. The suction peak is over-predicted as the level of grid refinement increases. The possible reason for the discrepancy is the use of a fully-turbulent flow assumption in the current simulation. The same trend is captured in the INS2D result [60] as shown in Fig. 5.12 (a).

The solution residual convergence rates are compared between the DDLGS and GMRES methods at the angle of attack of 8° as shown in Fig. 5.13. The residual




(a) Residual convergence per iteration (b) Drag coefficient per CPU time

Figure 5.13: Comparison of solution convergence for MD 30P-30N airfoil at AoA=8°.

is easily stalled using the DDLGS method for all the different resolution grids. The reason for the residual stall can be both from the use of high order reconstruction (WENO) and the first order approximations in the line-implicit method. The use of GMRES method can increase the convergence rate for all the test grids. For the results, 10 Krylov vectors are used for the coarse and medium mesh, and 20 Krylov vectors are used for the fine mesh. It should be noted that the current GMRES is used only for the mean flow equations and the SA turbulence model is solved using DDADI, which is loosely coupled with the mean flow equation.

The advantage in the current GMRES method is that the converged drag values could be obtained much faster than the DDLGS method as shown in Fig 5.13 (b). For the medium mesh, the drag is converged to within 1 % of the final value within 500 seconds using GMRES, otherwise it takes about 5,000 seconds using DDLGS. The similar trend is also observed for both coarse and fine mesh cases.

## 5.3 Scalability

The current flow solver is parallelized using MPI to be executed in parallel for realistic problems which typically include millions of cell elements. In this section, the flow solver scalability is tested through a strong scalability test, which is defined as how the solution time varies with the number of processors for a fixed total problem size. The current parallelized codes are executed on the University of Maryland's Deepthought II cluster, which is a high-performance computing facility that has a peak performance of about 160 TFlops/s.

The case for the scalability test is the time-accurate turbulent flow simulation over a sphere (Re=1.14 million and M=0.2). For the surface mesh, the initial 20,480 isotropic triangles are subdivided into 61,440 quadrilaterals using quad-level 0, and the surface mesh is extruded in the wall normal direction using 74 strand layers. Thus, the total problem size is 4.5 million hexahedra. Initial wall normal spacing is $1 \times 10^{-5}$ of the sphere diameter which corresponds to a $y^+$ of 0.5. MUSCL reconstruction is used for the inviscid flux computation and the DDLGS method is used for the implicit inversion.

The number of processors is varied from 16 to 250 and the averaged CPU time is measured after enough iterations. Figure 5.14 compares the current speed-up result against the ideal speed-up line. The code demonstrates parallel efficiency close to ideal with 64 processors and 90% efficiency with 250 processors. There can be multiple reasons for the decreased parallel efficiency at large number of CPUs. As one of the reasons, the current solver adopts a concept of ghost cells along the

boundary between partitioned domains (see section 2.3), thus the total number of elements increases as the number of partitioning increases, even for the fixed size of the total domain. The use of additional processors also results in additional MPI communications.

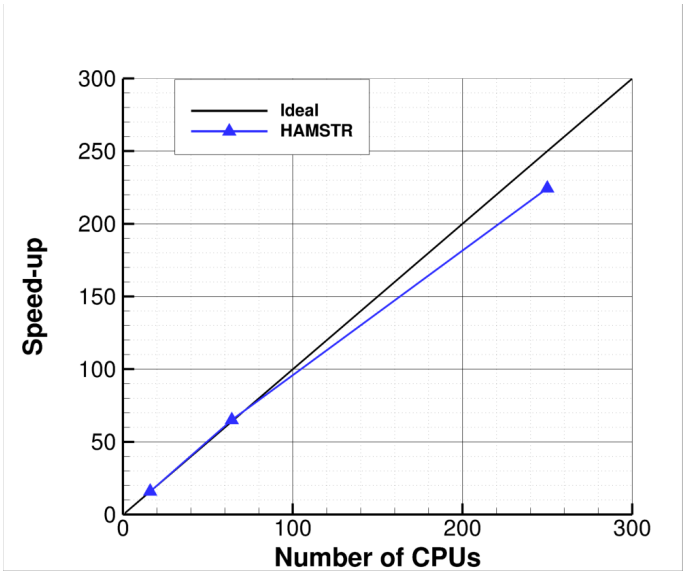
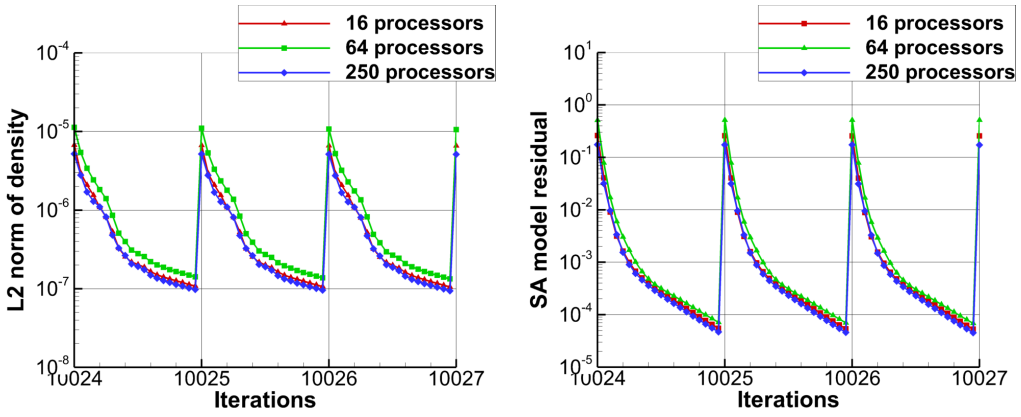



Figure 5.14: Strong scalability test for turbulent flow simulation over sphere.

Figure 5.15 shows the convergence of the mean flow residual and the turbulence model residual for a couple of time steps. In the current simulation, 20 sub-iterations are used at each time step. It is observed that the residual drops by two orders of magnitude in the mean flow and three orders of magnitude in the SA model. The solution convergence rates are not affected by the number of processors in both the mean flow and the SA model.




(a) Mean flow residual convergence

(b) SA turbulence model residual convergence

Figure 5.15: Convergence of residual for turbulent flow simulation over a sphere.

## 5.4 Summary

In this chapter, following topics are covered as a further investigation of the current flow solver performance.

- Solution convergence rate and accuracy are evaluated for the case with self-crossed loops. In both two- and three-dimensional simulations, the results are compared between the grids with and without the self-crossed loops. It is observed that the current line-implicit method and stencil-based reconstruction are not affected much by the existence of the self-crossed loops.

- Solution convergence rates are evaluated for the preconditioned GMRES method with the DDLGS method as a preconditioner. Improved convergence rates are observed compared to the results from the pure DDLGS method. The comparisons are conducted through various test cases: transonic inviscid flow around

hovering rotor, unsteady laminar flow around a sphere, and fully-turbulent flow around MD 30P-30N airfoil. The use of GMRES method alleviates residual stall problems in both steady and unsteady simulations by allowing a larger drop of residual per iteration. Also, aerodynamic coefficients are converged much faster in terms of CPU time compared with the results from the pure DDLGS method. However, more executive CPU time per iteration is required for the GMRES method, which depends on the size of Krylov subspace.

- The scalability test of flow solver is conducted by comparing executive CPU time per iteration at varying number of processors from 16 to 250. Time-accurate turbulent flow simulation around a sphere is used for the test and unsteady residual convergence rates are compared between the results at different number of processors. The code demonstrates 90% of ideal parallel efficiency with 250 processors. The solution convergence rates are not affected by the number of processors in both mean flow and SA model.

# Chapter 6: Applications

In this chapter, the Python-based CFD framework has been applied to helicopter/wind turbine flow simulations. In the framework, the developed method is utilized for either the near-body domains or all of the domains including the off-body region.

First, hovering rotor simulations are conducted including laminar-turbulent boundary layer transition. The effect of boundary layer transition on hovering performance will be discussed. Second, a full wind turbine configuration is simulated as an interactional aerodynamic problem between sub-components. The predicted interations are compared with the experimental data. Third, the flow around complex rotor hub geometries are simulated using unstructured volume meshes. The unsteady hub drag and wake deficits at the wake regions are compared with the experimental data and other simulation results. Finally, forward flight rotor with high advance ratio simulations are performed using the CFD-CSD coupled method. Current simulation is focused on sectional airloads prediction at 30% rotor radius. The accuracy of the coupled method is compared with the results obtained from either CSD alone and CFD alone.

## 6.1 Pressure Sensitive Paint (PSP) Hovering Rotor Simulation

In 2016, a Mach-scale hovering rotor test was conducted in the NASA Langley Research Center to investigate the hover performance as a function of the laminar-turbulent transition state of the boundary layer [61]. The hover performance was measured for natural and forced transition cases. For the natural transition test, boundary layer transition locations were measured on the upper and lower blade surfaces via infrared (IR) thermography. The PSP rotor is a four-bladed configuration and has a radius of 66.5 inches with a chord of 5.45 inches. The blade used RC-series airfoils as shown in Fig. 6.1 and had a linear twist of -14 degrees starting at $r/R = 0.252$ and ending at the rotor tip. The rotor solidity ($\sigma$) is 0.1033. The operating tip Mach number is 0.58, and the tip Reynolds number is 1.7 million based on the reference chord length. The test was conducted with a ROBIN-Mod 7 fuselage beneath the rotor.

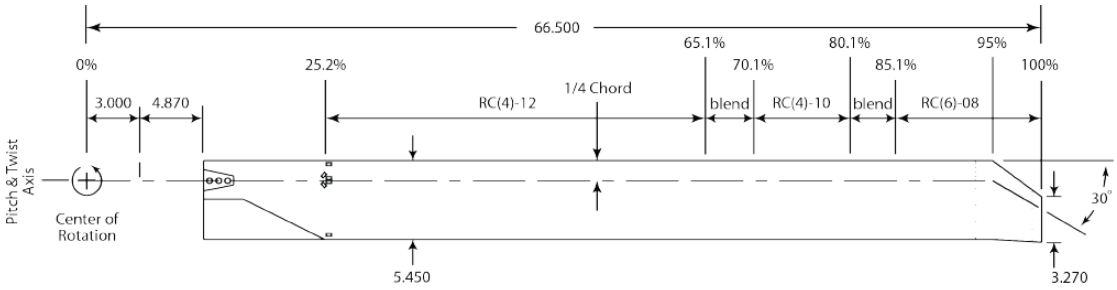
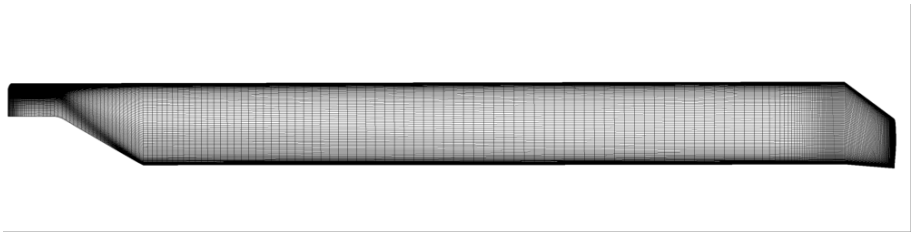



Figure 6.1: PSP blade planform, inches [61].

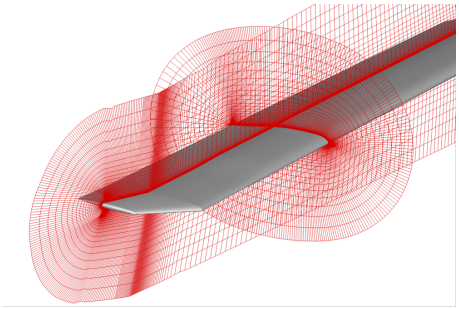
In the current work, both fully turbulent and transitional flow simulations are conducted for the isolated rotor at three different collective angles (6°, 8°, and 10°)

and the predicted hovering performance results are compared with experimental data. For the transitional flow simulations, the $\gamma - \overline{Re_{\theta t}}$−SA transition model is coupled with the SA turbulence model. The freestream turbulence intensity is prescribed as 0.75% according to the previous study [62].

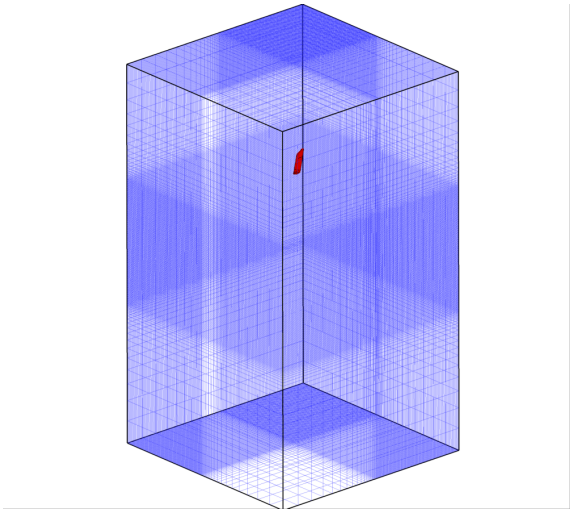
Figure 6.2 shows the computational overset mesh system for the simulation, which consists of a structured O-O type grid for the blade domain and a Cartesian background domain. The current flow solver, HAMSTR, computes the blade domain, otherwise the background domain is computed using OVERTURNS [46]. Owing to the periodic nature of the flow, the simulations are performed only for the single blade of the rotor with an assumed periodic boundary condition.



(a) Blade surface mesh



(b) O-O mesh topology for blade volume mesh



(c) 1/4 Cartesian background mesh

Figure 6.2: Computational overset mesh system for PSP rotor simulation.

The blade mesh has $265 \times 275 \times 70$ points in the chordwise, spanwise, and wall-normal directions respectively. The initial wall normal spacing is $5 \times 10^{-6}$ the reference chord length which corresponds to a $y^+ = 0.5$. For the background domain, the radial outer boundary is at $3.2R$ and the top and bottom boundaries are at $2.0R$ and $3.5R$, respectively. At the outer boundaries, the point-sink/momentum theory boundary condition is applied to account for far-field conditions.

The inviscid fluxes are computed using WENO reconstruction and the SA-DDES model is used for a turbulence closure for both domains. Otherwise, the

transition model is applied only for the blade domain. The connectivity between the overset meshes is computed using TIOGA once before the iterations, and both computational domains are rotated together by 0.5° at every iteration.

Figure 6.3 shows the predicted figure of merit (FM) for both laminar-turbulent transition and fully-turbulent simulations. The results are compared with the experimental data and other simulation results which used either $\gamma - \overline{Re_{\theta t}}$ [63] or the SA amplification factor [64] transition model. In the experiment, trip dots were placed at $x/c = 0.05$ on the upper and lower surfaces to obtain the fully turbulent boundary layer. The current predictions are conducted at three different collective angles and the points are connected as shown in the bold lines.

Overall, there is good agreement between the current predictions and the reference data. In the current results, higher figure of merit are predicted for the fully turbulent flow condition than the results of the transitional flow simulation at all of the collective angles. The same trends are observed in both experimental data and the other simulation results. Although the current prediction for the fully turbulent flow is underpredicted at 10° collective pitch, a similar trend is also observed in the other predictions.

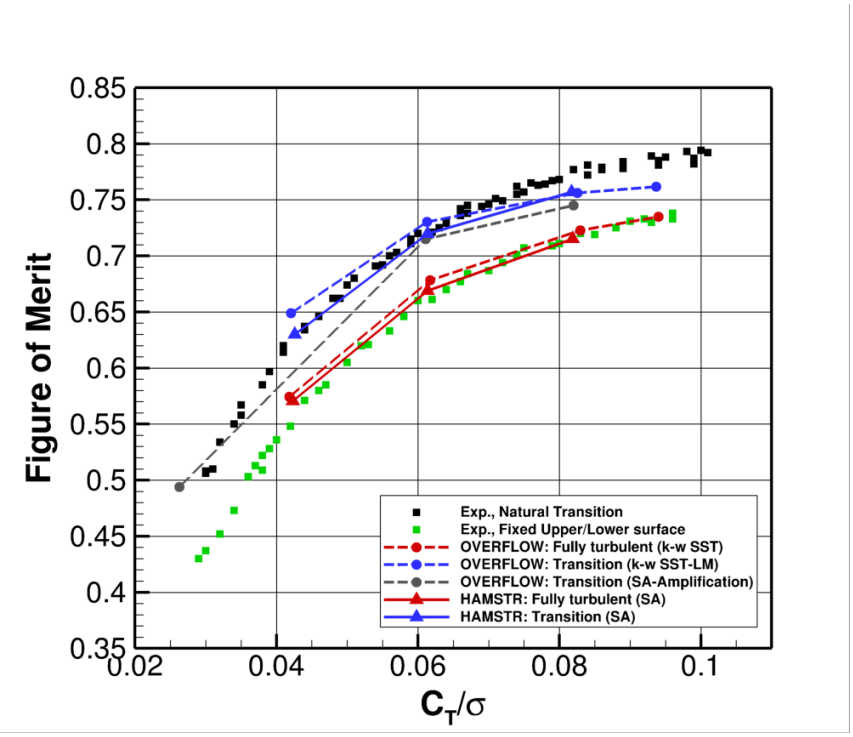
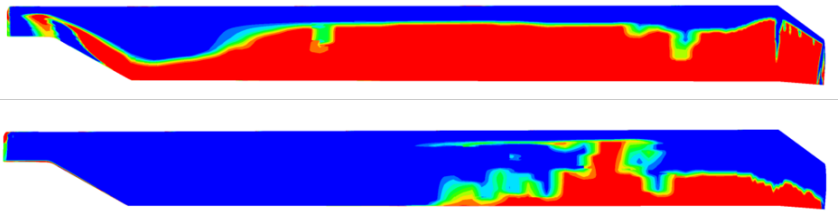



Figure 6.3: Comparison of figure of merit (FM) with experimental and other simulation data.

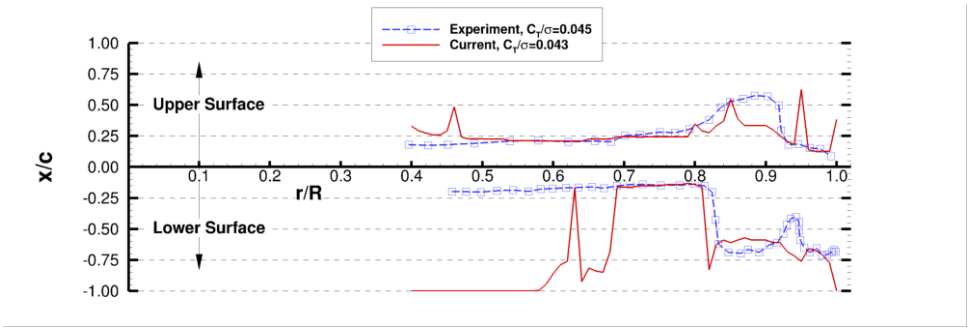
The predicted transition locations on the blade surface are compared with the experimental data for the collective angles of 6°, 8°, and 10° as shown in Figs. 6.4, 6.5, and 6.6, respectively. In the intermittency contours, the blue and red region approximately represents the laminar and turbulent boundary layers, respectively. It should be noted that only the source term of the turbulence model is dependent on the intermittency, otherwise the convection and diffusion terms are not dependent on the intermittency. The transition onset location is determined by picking up the point where the intermittency value reaches 0.5.

Overall, the predicted transition location moves towards the leading edge on the upper surface and towards the trailing edge on the lower surface as the collective

angle increases. This is because the boundary layer becomes less stable under an adverse pressure gradient and more stable under a favorable pressure gradient condition. This trend is also captured in the experimental data. In detail, the predicted locations are somewhat delayed on the mid-span of the lower surface at the collective angles of 6° and 8° compared with the experimental data. This might be due to the use of prescribed freestream turbulence intensity (FSTI) over the domain in the current transition model. In the current SA turbulence model, the local freestream turbulence intensity is not predictable unlike with the $k-\omega$ SST turbulence model. Due to the rotor wake, a higher FSTI can occur underneath the blade, which might cause the transition to occur earlier on the lower surface in the experiment. This interference becomes more severe at the lower collective angles.

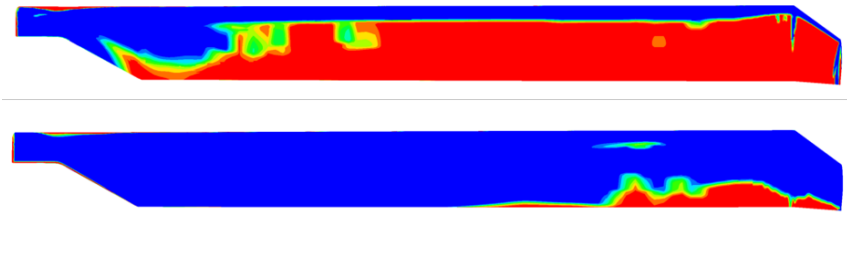


(a) Intermittency contours (top: upper, bottom: lower)

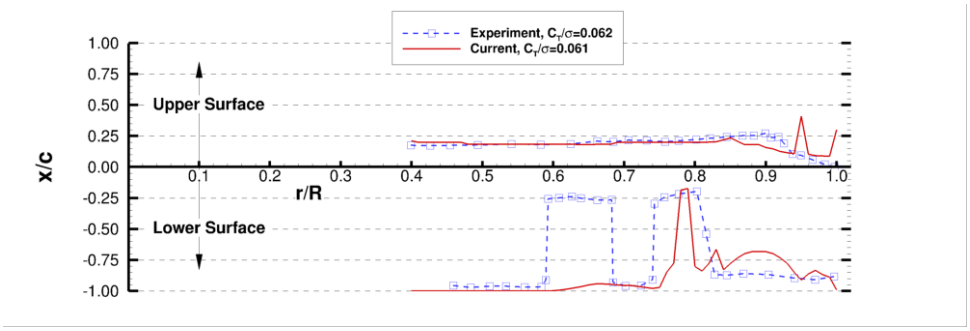



(b) Comparison of experimental and predicted transition location

Figure 6.4: Intermittency contours and transition location for $\theta_{0.75} = 6°$.

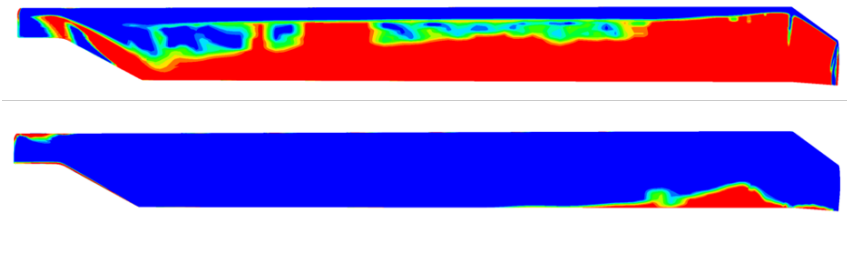


(a) Intermittency contours (top: upper, bottom: lower)

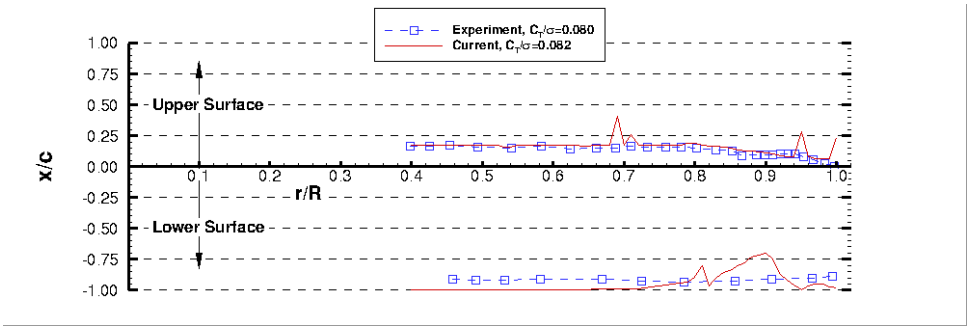



(b) Comparison of experimental and predicted transition location

Figure 6.5: Intermittency contours and transition location for $\theta_{0.75} = 8°$.



(a) Intermittency contours (top: upper, bottom: lower)




(b) Comparison of experimental and predicted transition location

Figure 6.6: Intermittency contours and transition location for $\theta_{0.75} = 10°$.

## 6.2 NREL Phase VI Wind Turbine Simulation

Wind turbines are effective means of obtaining environmental-friendly sustainable energy. In recent years, the number of turbines operated around the world has dramatically increased. Because the aerodynamic efficiency is directly linked to the efficiency of wind power generation, wind turbine aerodynamics research has been conducted using both low-fidelity models such as an actuator disk or a vortex-panel method [65] and high-fidelity CFD simulations [66, 67].

Unlike a low-fidelity model, the CFD simulation allows for accurate prediction of the boundary layer on the rotating blades and various unsteady flow features such as dynamic stall. Also, the use of a transition model improves the performance prediction of a wind turbine by capturing the laminar-turbulent transition point along the boundary layer. In the case of a horizontal-axis wind turbine (HAWT), the unsteady aerodynamic effects associated with the blade tower interference (upwind configuration) or tower shadow effect (downwind configuration) cannot be ignored and must be included in the analysis as well. Thus, the flow simulations around full wind turbine configuration are conducted in the present work.

The computations are performed for the National Renewable Energy Laboratory (NREL) Phase VI turbine rotor using the Python-based framework. The experiment was carried out in the NASA-Ames wind tunnel and the experimental data set has been used extensively in many other works to compare with CFD simulation results [68]. Most CFD studies have been conducted for the isolated rotor case with zero coning and yaw angles at wind speeds between 7 $m/s$ and 20 $m/s$ [66, 67].

This two-bladed turbine has a diameter of 10.06 $m$ and rotates with a constant rotational speed of 72 RPM. As the HAWT configuration, both upwind and downwind configuration experimental data are available. The blade features a cylindrical cross-section at the root and the transition region connects the circular section to the root airfoil section as shown in Fig. 6.7. The blade has a linear taper and has a non-linear twist with the tip pitch angle set as 3°. The pitch and twist axes are located at 30% chord [68].

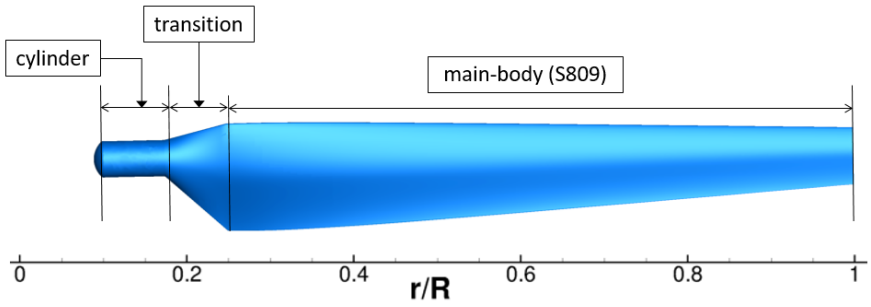
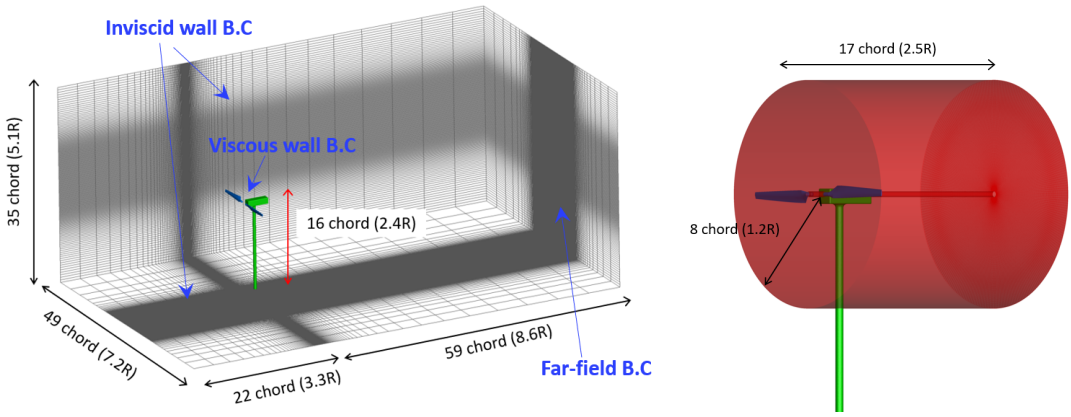



Figure 6.7: NREL Phase VI turbine blade planform.

The Phase VI turbine is mounted on the top of the tower, which has a diameter of 0.4 $m$ and a hub height of 12.2 $m$ from the ground. The turbine has a 1.4 $m$ longitudinal clearance from the tower, which corresponds to 3.5 times the tower diameter. The computational model of the full wind turbine is shown in Fig. 6.8. The nacelle is represented using a simplistic rectangular shape with the dimensions based on the actual size of the model.

The computational mesh for the full configuration consists of five individual overset mesh groups; three Hamiltonian-Strand grids for the two turbine blades and combined tower-nacelle configuration, and two background grids (cylindrical

nested mesh and Cartesian background mesh). The dimensions of the background domain is determined based on the NASA Ames wind tunnel test section as shown in Fig. 6.8. An inviscid wall boundary condition is applied to the test section wall faces and a freestream boundary condition is used for the inlet and outlet faces.




(a) Cartesian background mesh (b) Cylindrical nested mesh

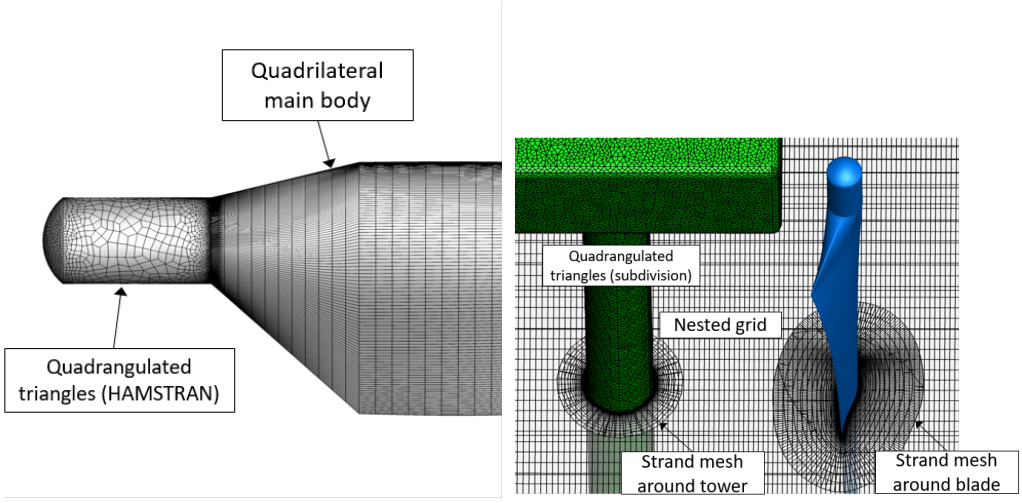
Figure 6.8: Computational model for full NREL Phase VI turbine simulation.

Each blade comprises of 9,082 triangles forming the cylindrical inboard section and the blade tip region, and 27,007 quadrilaterals in the transition and main body regions. In the main body of the blade region, the structured mesh consists of 240 points in the wrap-around direction and 90 points in the spanwise direction. After converting to an all-quadrilateral element mesh, a total of 32,639 quadrilaterals are used for each blade surface as shown in Fig. 6.9 (a). The blade grid contains 52 strands layers for the volume mesh with an initial wall spacing of $1 \times 10^{-5}$ of the root chord, which corresponds to a $y^+ = 0.5$.

In order to better capture the blade tip vortex, a grid spacing of about 4% root chord is used along the blade tip path in the cylindrical nested mesh. A grid

spacing of about 6% root chord is used between the blade and tower to capture possible interference. A total of 5.8 million and 4 million elements are used in the nested mesh and the Cartesian background mesh, respectively. Within the Python framework, the near-body domains including the blades, nacelle, and tower are computed using HAMSTR and the off-body domains are computed using the GPU-based solver (Garfield) [54].

To simulate the relative motion between the blade and non-rotating components, the blades are rotated at a time step size of 0.5° around the rotational axis. WENO reconstruction is used for the inviscid flux in all of the domains. For the implicit inversion, DDLGS and DADI methods are used for the near-body and off-body domain, respectively.




(a) Blade surface mesh at root region

(b) Overset connections between blade and tower at 63% R plane

Figure 6.9: Computational mesh for blade and overset system for NREL Phase VI turbine simulation.

### 6.2.1 Isolated Rotor Computation

The simulations for an isolated rotor are performed to compare against experimental data at various flow conditions and to capture the effect of the laminar-turbulent transition model on turbine performance. For the transition simulation, the freestream turbulence intensity (FSTI) is set as 0.1%. Owing to the largely periodic nature of the flow without tower interaction, the calculations are performed only for a single blade of the rotor and a periodic boundary condition is applied at the plane between the blades.

As shown in Fig. 6.10, the instantaneous chordwise surface pressure distributions are obtained with and without the transition model and the current predictions are compared against the experimental data at wind speeds of 7, 10, and 20 $m/s$. The pressure distributions are compared at five spanwise sections where the experimental data is available; $r/R$ of 0.3, 0.47, 0.63, 0.8, and 0.95. The sectional pressure coefficient is defined as given by:

$$C_p = \frac{P - P_\infty}{0.5\rho({W_\infty}^2 + (r\omega)^2)} \tag{6.1}$$

where $W_\infty$ is freestream windspeed and $\omega$ is the rotational speed.

At a wind speed of 7 $m/s$, the flow is fully attached over the blade. The current predictions from both fully turbulent and transitional flow simulations show a good agreement with the experiment. At 10 $m/s$, the flow separation appears at mid-chord on the suction side. Some deviations with experiment are observed on the suction side of the blade; this phenomenon is similar to that observed in other

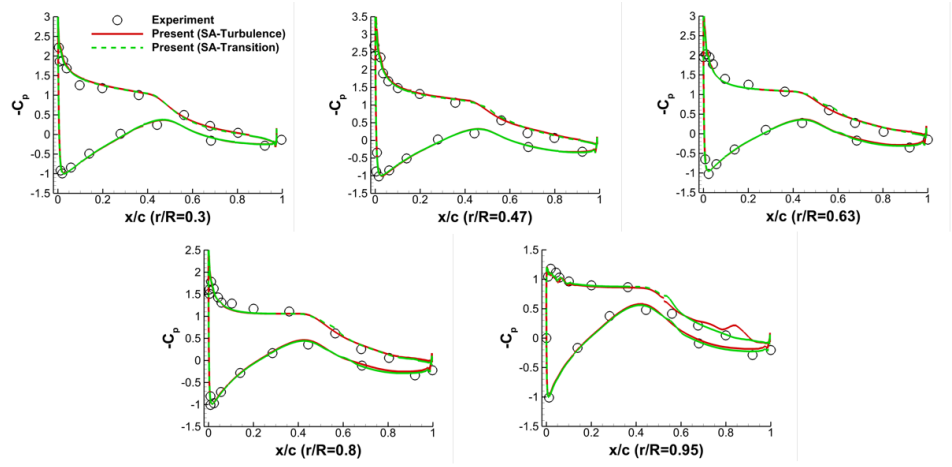
numerical studies [66]. This is because the separation point on the suction side is sensitive to the numerical scheme and turbulence model used in the simulation. At 20 $m/s$, the flow is fully separated from the leading edge over the whole region of the suction side. Thus, the flat pressure distribution is captured on the suction side in both experiment and prediction.

The thrust and torque predictions obtained from fully turbulent and transitional flow simulations are compared with the experimental data in table 6.1. At an operating wind speed of 7 $m/s$, the use of a transition model estimates better torque, while the fully turbulent simulation is under-predicted. This is because a transition model captures the laminar portion of the boundary layer on the blade where the skin friction is lower than the value in the turbulent flow region. Once the flow is separated on the suction side at higher wind speeds, the transition model has a lesser impact on the torque prediction. Overall, all torque predictions are improved using the transition model and show reasonable agreement with experimental data. The thrust is only moderately affected by transition.

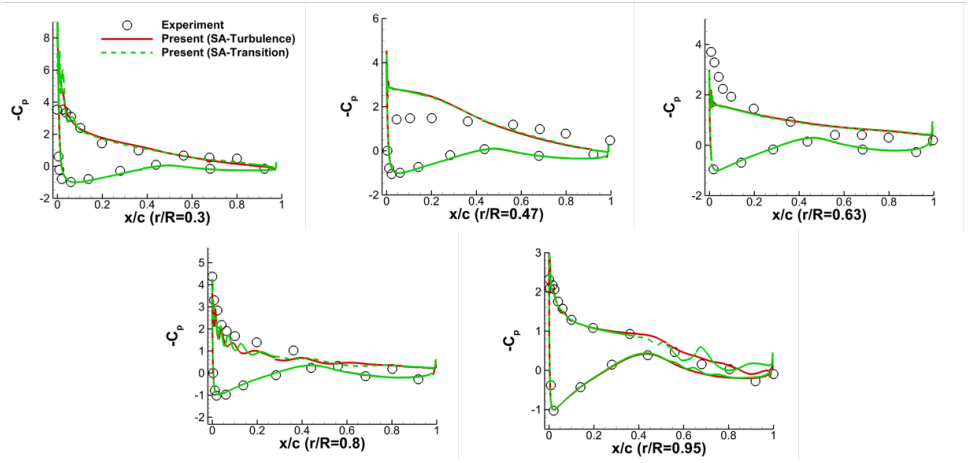
Table 6.1: Comparison of thrust and torque predictions for NREL Phase VI turbine, experimental data from [38].

| Case($m/s$) | Thrust ($N$) | | | Torque ($N-m$) | | |
|---|---|---|---|---|---|---|
| | Experiment | Transition | Turbulence | Experiment | Transition | Turbulence |
| 7.0 | 1,154 | 1,185 | 1,126 | 805 | 766 | 696 |
| 10.0 | 1,675 | 1,673 | 1,665 | 1,340 | 1,196 | 1,172 |
| 20.0 | 3,005 | 3,278 | 3,261 | 1,110 | 1,134 | 1,084 |

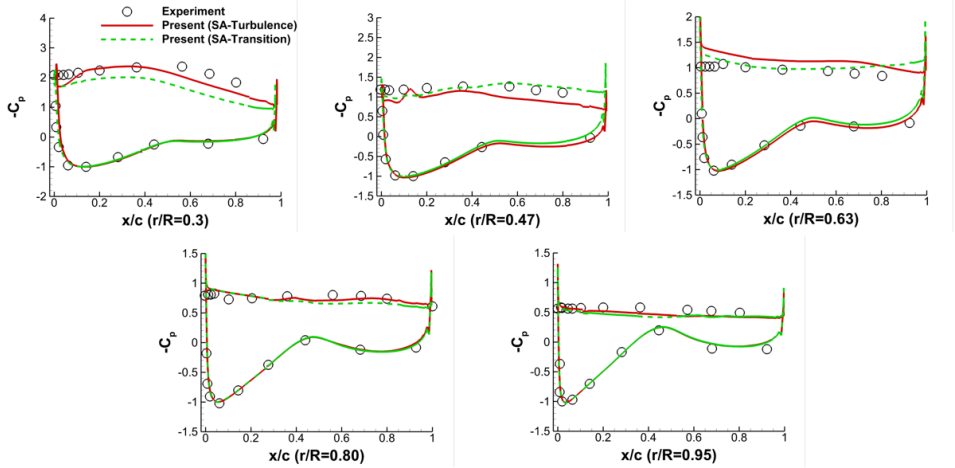
At a wind speed of 7 $m/s$, the streamlines overlaid on skin friction contours




(a) Wind speed of 7 $m/s$

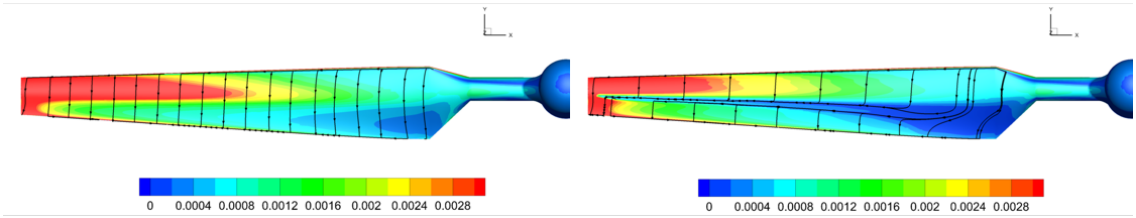



(b) Wind speed of 10 $m/s$

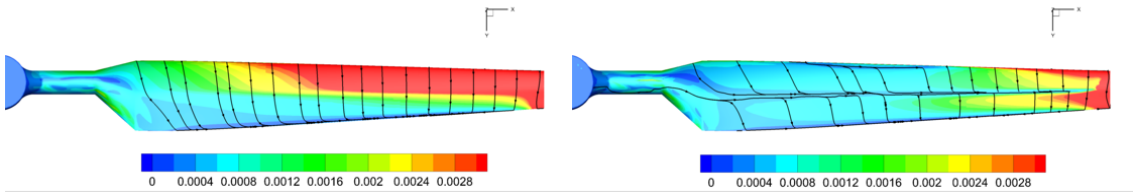



(c) Wind speed of 20 $m/s$

Figure 6.10: Sectional surface pressure for NREL Phase VI turbine blade.




(a) Windward side of blade (left:fully turbulent, right:transition)




(b) Leeward side of blade (left:fully turbulent, right:transition)

Figure 6.11: Streamline overlaid on skin friction contour for NREL Phase VI blade at wind speed of 7 $m/s$.

are shown in Fig. 6.11. The results from fully turbulent and transition simulations are compared with each other on both windward and leeward sides. The results including a transition model clearly show the presence of a laminar-separation bubble along the mid-chord of the entire span on both sides of the blade.

Once the flow is separated, the flow spreads towards the blade tip due to the centrifugal force from the rotating motion and the turbulent flow re-attaches right after the bubble. Especially on the leeward side of blade, the use of a transition model predicts much less skin friction before the separation than the result from a fully turbulent flow. This difference results in the better agreement with the experiment in the turbine torque prediction.

### 6.2.2 Full Configuration Computation

For the full configuration, including the nacelle and tower components, both upwind and downwind tests are simulated at a wind speed of 7 $m/s$. Details of the operating conditions are explained in table 6.2. For the downwind configuration, the rotor is modeled without coning and teetered hub in the current simulation, which is the same assumption as in the referenced study [69].

Table 6.2: NREL Phase VI operating conditions.

| Condition | Rigid/Teetered | Coning angle | Blade tip pitch |
|---|---|---|---|
| Upwind | Rigid | 3.4° | 3.0° |
| Downwind | Teetered | 0.0° | 3.0° |

In the downwind configuration, the blade-tower interaction occurs due to the tower shed wake interference on the blade aerodynamic loading. Because the shed vortices from the tower directly interact with the blade, the effect on blade airload is more severe than the upwind configuration. One case from Sequence 'B' in the experiment is chosen [68], where the nominal inflow velocity was 7 $m/s$. However, in this study, the actual inflow velocity of 6.7 $m/s$ is used due to wind tunnel anomalies as mentioned in the referenced study by Zahle et. al [69].

The validation for this interaction is performed by comparing the azimuthal variation of normal force coefficients with the experimental data at the five spanwise locations ($0.3R$, $0.47R$, $0.63R$, $0.8R$ and $0.95R$). Figure 6.12 shows the computed normal force coefficient variation during a rotor revolution. The current predictions are compared against the averaged experimental data over the 35 consecutive

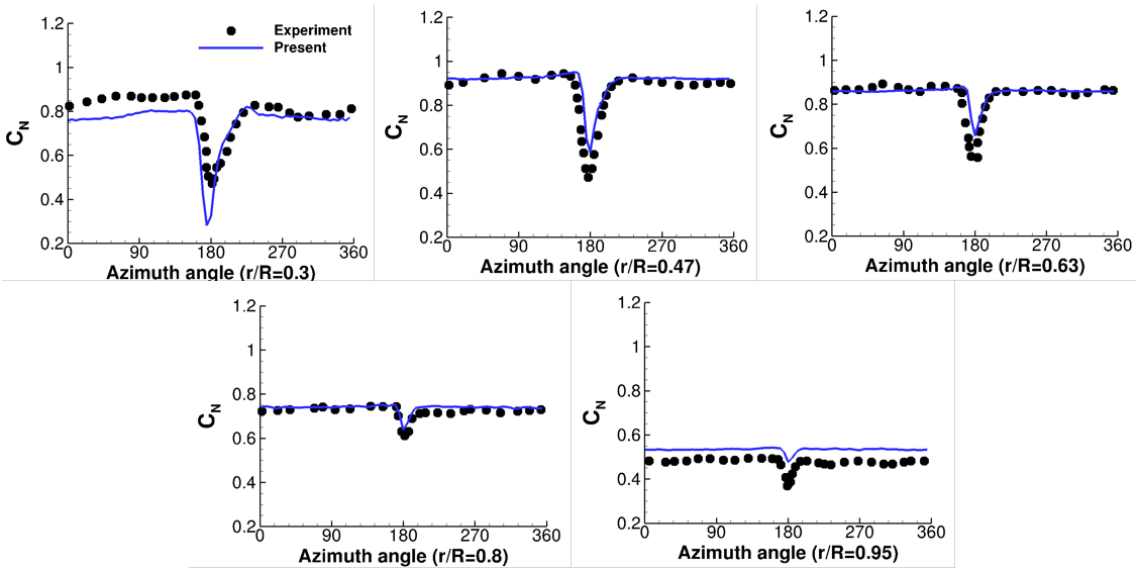



Figure 6.12: Normal force coefficient variation on a NREL Phase VI turbine blade for downwind configuration.

revolutions.

The acute interaction is observed at all of the spanwise locations and the rate of reduction is greater at the inner section than the outer section. This trend is observed in both experiment and the current prediction. This is because the decrease in effective angle of attack is more severe at the inner section which has lower tangential velocity. This resulted in almost 50% of reduction in the normal force at the $0.3R$ section.

In the upwind configuration, the free-stream wind shear due to the atmospheric boundary layer is also applied in addition to the uniform flow condition in order to study the effect on the rotor performance. The free-stream wind shear is described by the normal wind profile (NWP) model [70] using a power law as given by:

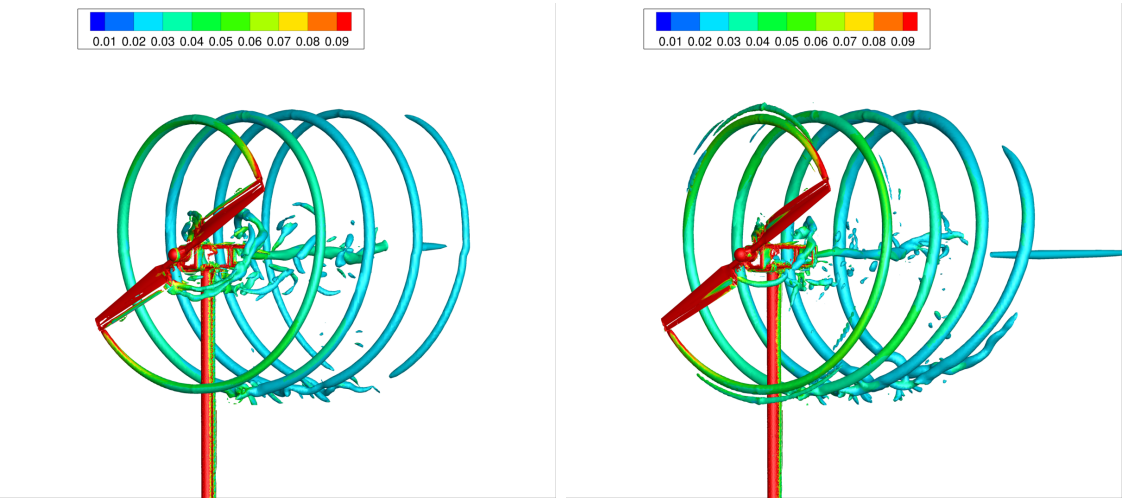
$$W(Z) = W_{hub}(\frac{Z}{Z_{hub}})^{\alpha} \tag{6.2}$$

where $W_{hub}$ is the wind speed at the hub height as 7 $m/s$ and $Z_{hub}$ is the hub height from the ground.

The value of $\alpha$ (0.2023) is determined using the correlation equation suggested by Justus [71] based on the reference hub height and the reference wind speed of 7 $m/s$. The maximum velocity on the rotor disc is about 7.53 $m/s$ at the top and the minimum velocity is about 6.25 $m/s$ at the bottom. The wind shear profile is imposed as both an initial value in the background domain and the Dirichlet far-boundary condition.

Figure 6.13 shows the wake structure from the NREL Phase VI wind turbine at both uniform inflow and at the wind shear condition. Both the blade tip and root vortices interact with nacelle and tower components. It is observed that the tip wake distances are different between the top and bottom of the rotor for the wind shear condition. This phenomenon can induce the low momentum zone to be lifted upward in the far-wake region due to more mutual interaction between the wake structures at the bottom [65].

Figure 6.14 (a) shows the single blade torque variation during a rotor revolution for both the uniform flow and wind shear cases. For the uniform flow, the region affected by the tower is not a single point, but the interaction is extended to almost half of a rotor revolution. The same trend is also captured in the experimental data. Compared to the acute interaction from the downwind configuration, this interaction

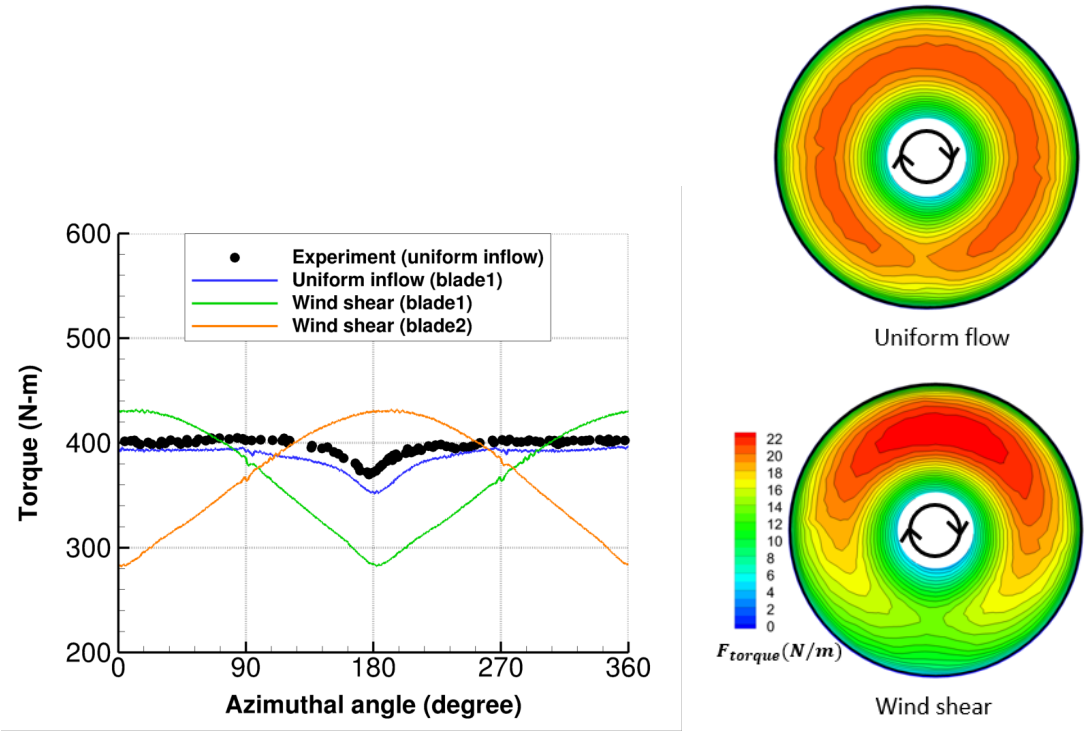



(a) Uniform inflow condition (b) Wind shear condition

Figure 6.13: Computational wake visualization of NREL Phase VI turbine using iso-surfaces (Qciteria=0.00015) colored by vorticity.

is less severe but it occurs over a wider range. For the wind shear flow, the variation of individual blade torque becomes severe which can induce highly unsteady hub moment of the turbine. However, the trends are quite opposite between the blades, thus most of the variation can be canceled out after both torques are summed up for the rotor torque. The experimental data for the wind shear flow condition is not available for the current wind turbine model.

Figure 6.14 (b) compares the sectional torque force (tangential force) distribution on the rotor disk plane between the cases of uniform flow (on top) and wind shear (on bottom). A strong asymmetric force is obtained in the presence of the wind shear because of the change in local effective angle of attack. Also, the asymmetric force is more severe at the blade tip region. Otherwise, under the uniform inflow condition the sectional airloads distribution is symmetric except at 180° azimuth




(a) Blade torque azimuthal variation

(b) Tangential force contour on rotor disk

Figure 6.14: Blade torque variations of NREL Phase VI turbine for upwind configuration.

due to the tower blockage effect.

## 6.3 Rotor Hub Simulation

Accurate flow simulation around a rotor hub is important in helicopter design. This is because the rotor hubs can contribute significant ( 30% or more) drag at conventional speeds, and even more at the high speeds (230 $kts$) planned for some future vertical lift designs [72]. Also, the wake from the rotor hub induces aerodynamic interference at the empennage which may result in stability issues. The

rotor hub typically consists of various components such as hub arms, swashplate, and shaft. Thus, the use of advanced CFD tools is motivated to provide accurate simulation and better understanding of the complex flow around the rotor hub. Recently, a combined computational and experimental efforts have been conducted through two Rotor Hub Flow Prediction Workshops which were held at Penn State University (PSU) in 2016 and 2018 [73, 74]. The main objective of the workshops is to understand both unsteady hub drag and wake characteristics from near to far wake from a rotating hub. In the current work, the flow simulations around the PSU hub models were conducted using the developed CFD framework.

The current hub models are based on a 4-bladed large commercial helicopter and were simplified for computational validation. Phase III hub (baseline hub) model resembles a modern commercial helicopter and a low-drag hub was designed to represent a future vertical lift hub [75]. The hub radius is assumed to be 15% of the rotor radius and the hub models were sized to have the same frontal area of 59.96 $in^2$.

The full-scale Reynolds number experiment is conducted in the Garfield Thomas Water tunnel at Penn State University (see Fig. 6.15). Both hub models are mounted on the wing-shape stand (NACA0025 section) at a 5 degree forward angle in reference to forward flight. The horizontal stabilizer is also installed downstream of the hub model. The water tunnel speed is 22 $ft/s$ and the rotation rate of the hub is 152 RPM.

Figure 6.16 compares the two hub models. The baseline hub model includes upper and lower spiders, main hub arms, swashplate, and scissors. For the low-drag

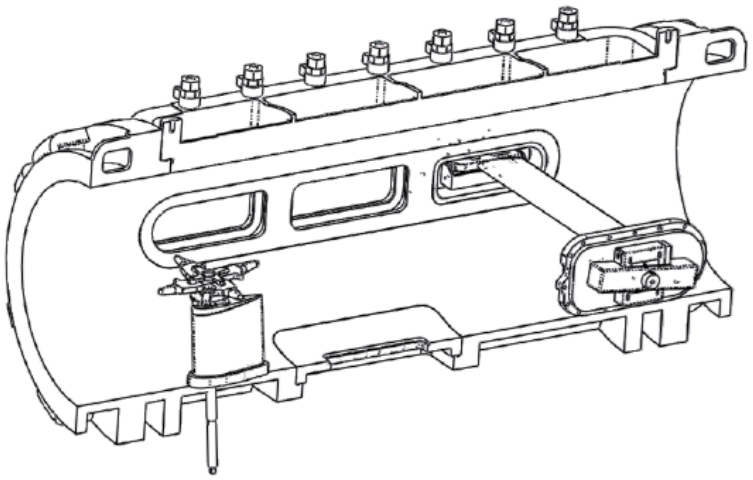
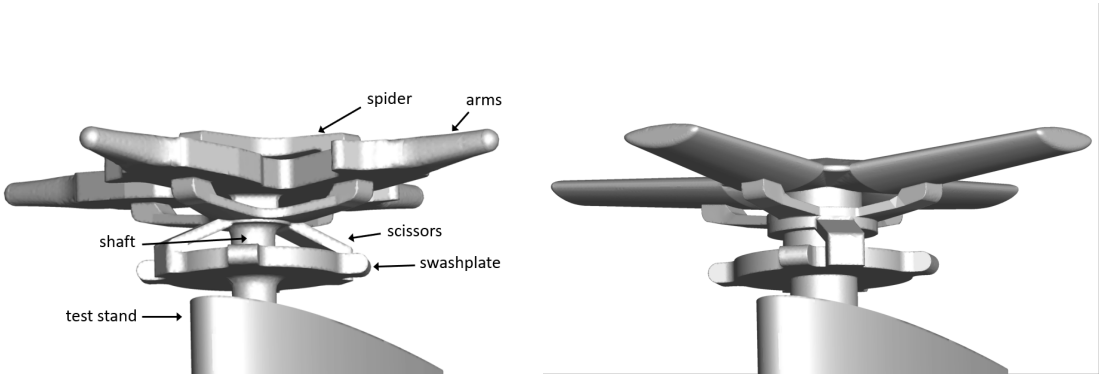


Figure 6.15: Experimental setup in the Garfield Thomas 48 inch diameter water tunnel [75].

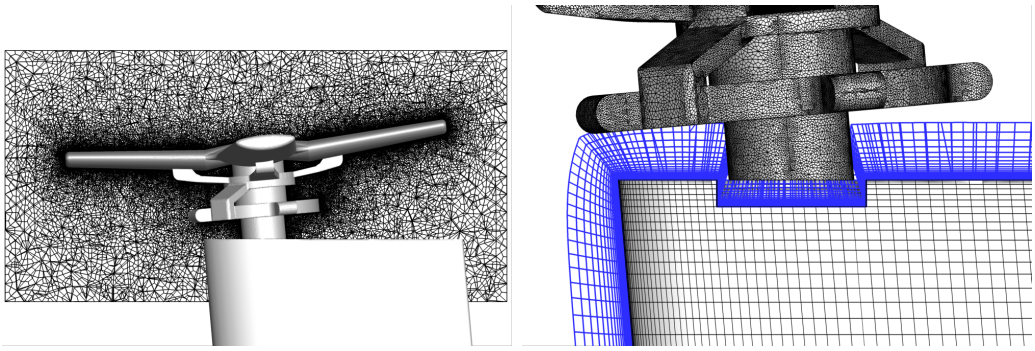
hub, the sharp corners of the baseline hub arms were replaced with a reverse flow airfoil (DBLN 526) arms and the upper spider was removed. Also, the low-drag hub has a larger shaft diameter than the baseline hub (Phase III).

For the near-body domain of both the baseline and low-drag hub models, unstructured volume elements are used which includes both prisms and tetrahedra. Then, each element is subdivided into hexahedrons to identify Hamiltonian paths on the volume domain as shown in Fig. 6.17 (a). The hub stand is also modeled inside the water tunnel and the hub model is mounted on the top of the hub stand. The initial surface mesh of the hub stand is generated by combining quadrilaterals and triangles which are eventually subdivided into all-quadrilaterals. Then, the strand grid is generated from the surface mesh as shown in Fig. 6.17 (b). Table 6.3 shows the resultant grid information of both baseline and low-drag hubs as well as the hub stand domain.




(a) Baseline hub model (b) Low-drag hub model

Figure 6.16: Computational PSU hub models.



(a) Unstructured volume element for hub model (b) Strand grid for hub stand

Figure 6.17: Computational mesh for hub model and hub stand.

Table 6.3: Near-body domain grid information for rotor hub simulation.

| | Baseline | Low-drag | Hub stand |
|---|---|---|---|
| Surface element | 84,738 | 225,000 | 22,074 |
| Normal spacing ($y^+$) | $3 \times 10^{-5}$ (2.5) | $1 \times 10^{-5}$ (1.0) | $1 \times 10^{-5}$ (1.0) |
| Volume element | 3.9 million | 12.5 million | 0.9 million |

Each of the near-body domains are connected with the off-body domains using the overset method. The overset mesh system consists of a total of 8 groups; hub, hub stand, hub nested, hub stand nested, stabilizer, stabilizer nested, tunnel, and tunnel nested. Figure 6.18 shows the side-view of the mesh system near the hub and stabilizer. The tunnel wall is a cylindrical structured mesh and viscous no-slip boundary condition was imposed on the wall. The tunnel wall ranges from -11 to 28.9 hub radius and has 1,435,140 grid elements. To better preserve the wake flows from the hub, four overset nested meshes are additionally used. Each hub nested domain and foil nested domain has a uniform grid spacing of 0.015 and 0.02 hub diameter, respectively. The total number of grid elements for the nested meshes are about 15.40 million. The off-body domains and the horizontal stabilizer domain are computed using the GPU-based structured grid flow solver (Garfield) [54] within the Python framework.

The simulations are conducted at a Reynolds number of $4.3 \times 10^6$ based on the hub diameter and freestream Mach number of 0.1 which corresponds to 0.2 advance ratio based on the rotor. The unsteady time step of 0.5 degree is used with sub-iterations. In all of the domains, the flow is assumed as fully turbulent flow and the SA-DDES model is used for a turbulence closure.

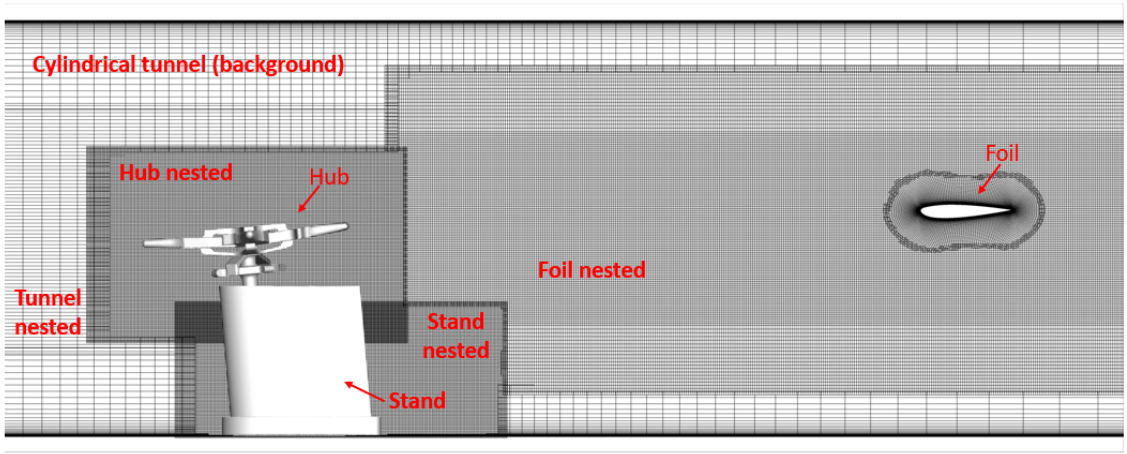
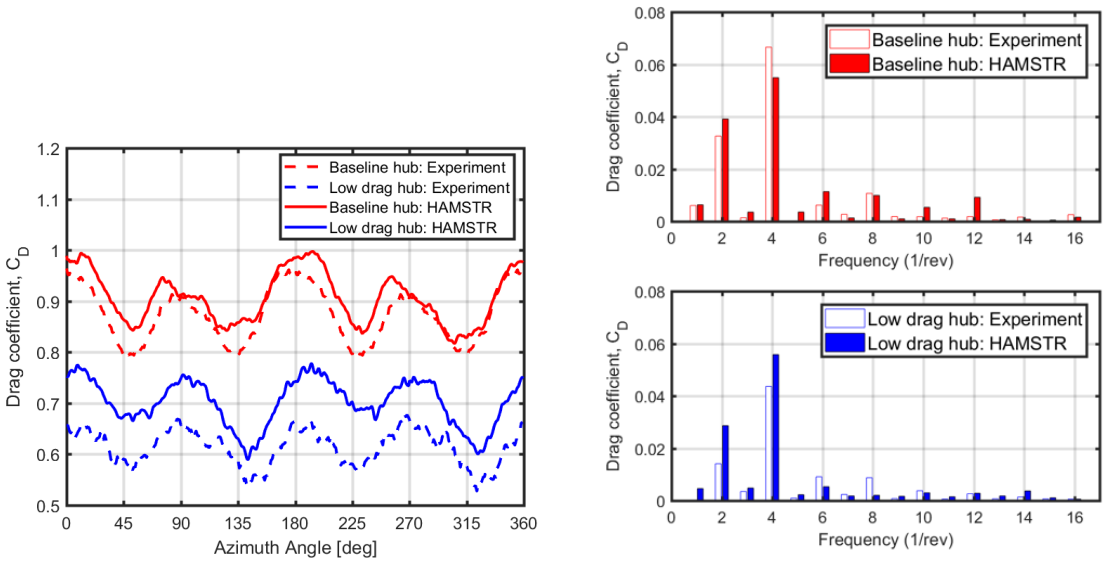



Figure 6.18: Overset mesh system for baseline hub model simulation.

In the near-body domain, WENO reconstruction is used for the baseline hub model and combined WENO and least-square reconstruction method is used for the low-drag hub model. In both cases, the DDLGS method is used for the implicit inversion. In the off-body domain, WENO reconstruction and the DADI methods are used for the inviscid flux and the implicit inversion, respectively.

Figure 6.19 (a) shows the azimuthal drag variations from the current predictions and experimental data [74]. It should be noted that the lower shaft component was excluded for the hub drag calculation in both experiment and simulations. As a validation, the current prediction from both baseline and low-drag hub models are compared against the measured drag from the same hub model. For the baseline hub model, a good agreement with the experiment is observed in terms of both mean value and unsteady trend. The current mean drag is slightly over-predicted by 4% from the experiment. For the low-drag hub model, the predicted mean drag is over-predicted by 14% although the unsteady trend is matched with the experiment




(a) Phase-averaged drag coefficient (b) Harmonics associated with hub drag

Figure 6.19: Drag coefficients for baseline and low-drag hub.

counterpart. The over-predicted drag in the low-drag hub model might be a result of poor mesh quality near the leading edge of hub-arms as stated in the previous study [76].

Frequency content analysis of the unsteady drag is shown in Fig. 6.19 (b). For the baseline hub, 2/rev and 4/rev are dominant in both experiment and prediction. Although the current result under-predicted 4/rev component and over-predicted 2/rev component, overall the current prediction shows reasonable agreement with the experiment. In the case of the low-drag hub, 2/rev and 4/rev are dominant in both experiment and prediction. However, the magnitudes are over-predicted at 2/rev and 4/rev components.

Mean drag breakdown by component is shown in Fig. 6.20. The components are divided into three groups: the first component includes lower shaft, the second

component includes swashplate, scissors, and mid-shaft, and the third component includes arms and spiders. As a validation, current results are compared with predictions using the Helios CFD framework [72, 77]. Noted that the drag of the first component for the low-drag hub model is not available in the Helios results.

For the baseline hub model, the third component has the highest drag due to their size, which corresponds to about 70% of the total drag in both the current and Helios results. The drag from the second component accounts for about 25% of total drag and less than 5% of total drag was predicted in the first component. It is observed that about 30% of the third component drag is reduced for the low-drag hub model in the current result, which is lower than a reduction from the Helios result (about 40%). The difference might be due to the poor mesh quality near the leading edge in the current simulation as mentioned before. The thicker shaft of the low-drag hub does not account for much of a drag increment.

Figures 6.21 and 6.22 show the wake visualization of the rotating baseline and low-drag hub models at a specific time instance, respectively. The top of each figure shows the Q-criterion iso-surfaces colored by vorticity magnitude and the bottom figure shows the vorticity magnitude contours on the horizontal plane of the water tunnel at $Z/R = -0.1$. In both results, the wake structures from the hub model are well preserved in the off-body domain using the overset method. It should be noted that the use of the SA-DDES turbulence method in the current simulation prevents the excessive dissipation of wake structures in the off-body domain compared to a RANS simulation. Thus, it is observed that the large wake structures were broken down into small eddies at downstream locations

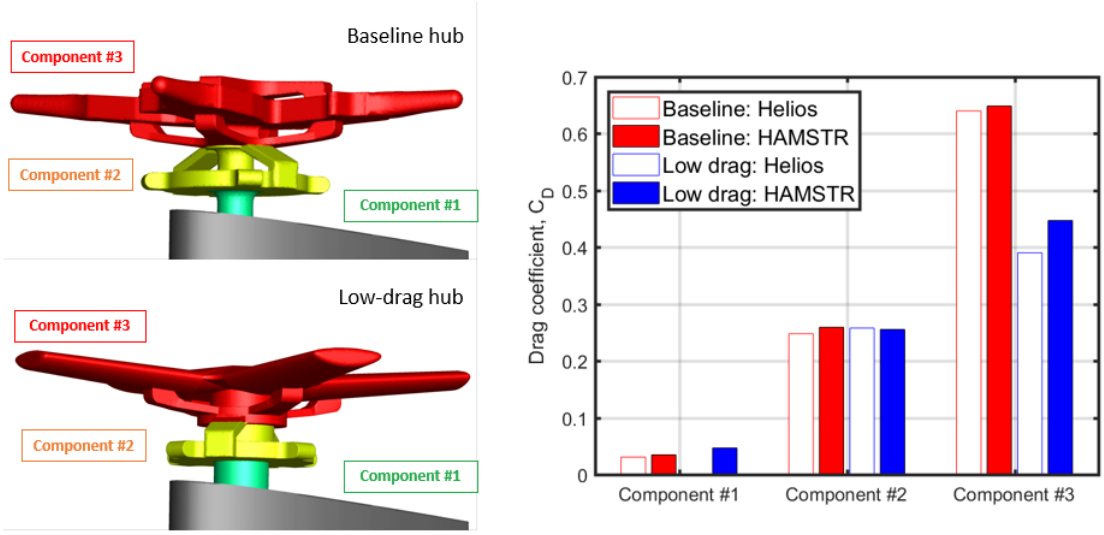
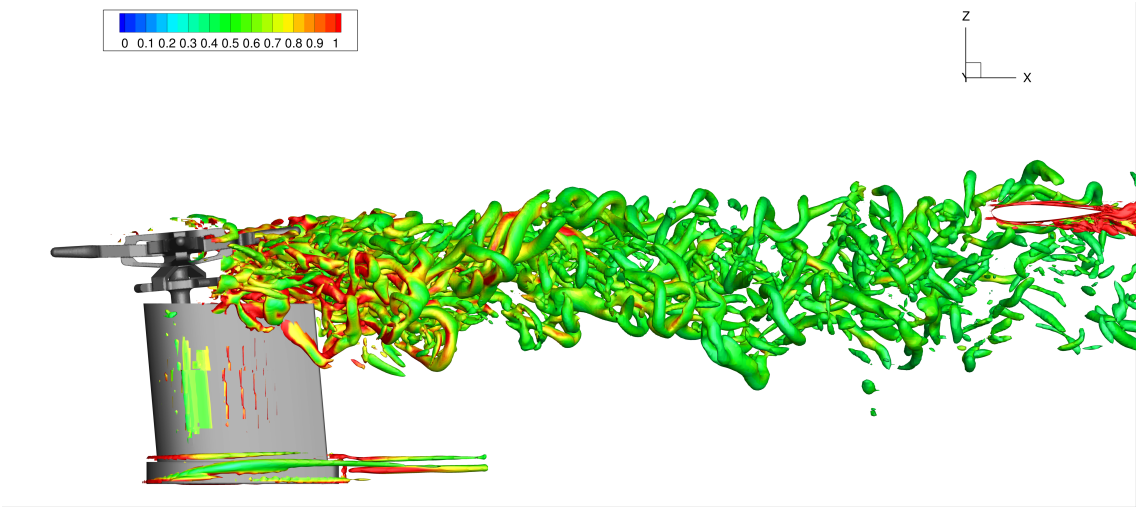


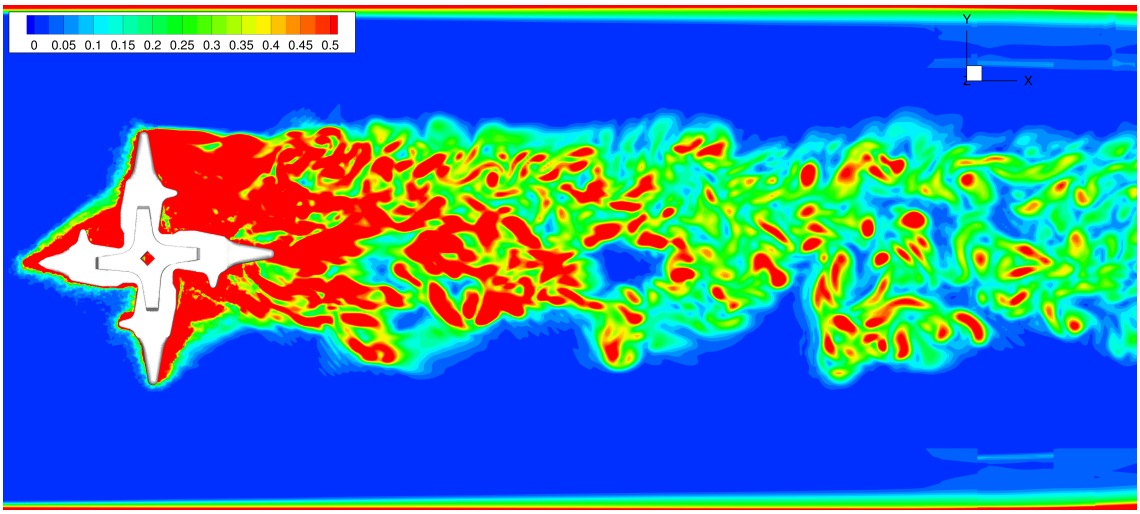

Figure 6.20: Mean hub drag breakdown by component for baseline and low-drag hubs.

Compared to the result from the baseline hub model, less wake structures are predicted from the low-drag hub model especially on the retreating side. This is because the whole retreating side of the hub corresponds to a reverse flow region and the use of a reverse flow airfoil in the low-drag hub generates less wake structures than the baseline hub arm with sharp corners.

For the baseline hub model, the current predictions of the mean streamwise velocities are compared with the experimental LDV wake measurements at $X/R$ = 2.09 (near-wake), 4.14 (mid-wake), and 7.15 (far-wake). As a feature of this experimental data, the wake was measured at the far downstream from the hub. In this comparison, the stabilizer is not included in both experiment and simulation. For the current prediction, the mean velocities are computed by time-averaging over 3 hub revolutions.

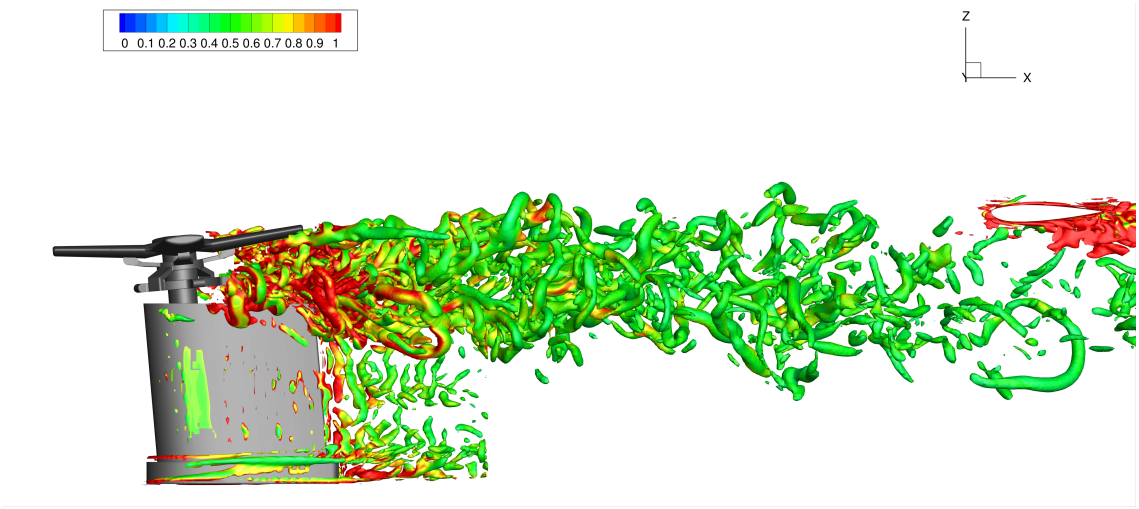



(a) Iso-surfaces (Qciteria=0.04) colored by vorticity (side view)

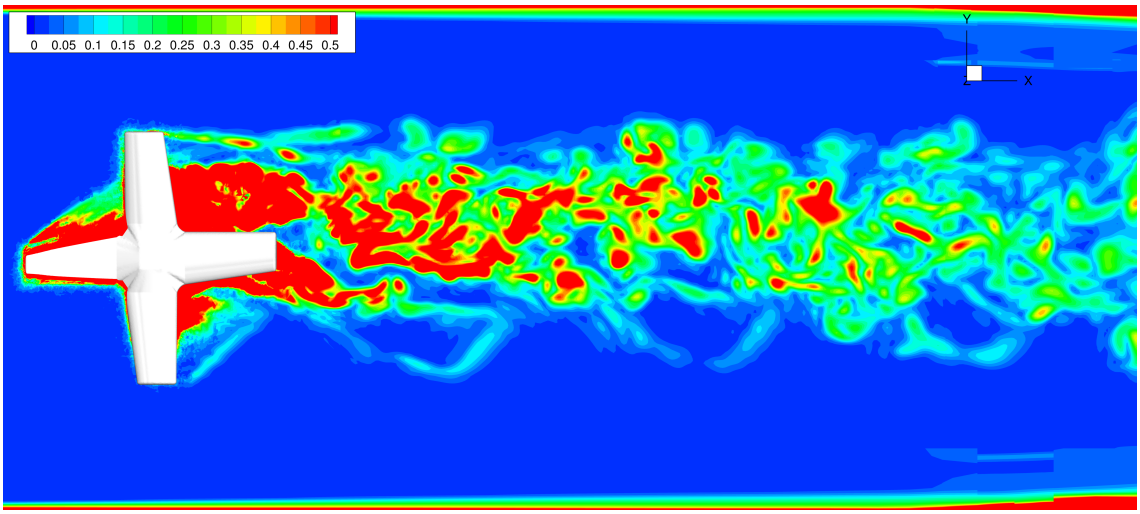



(b) vorticity magnitude at $Z/R = -0.1$ plane

Figure 6.21: Computational wake visualization of baseline hub.

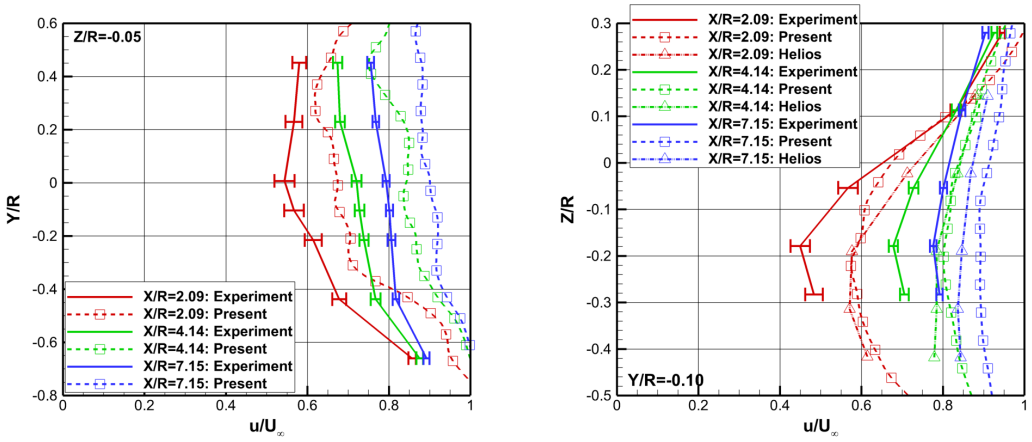



(a) Iso-surfaces (Qciterion=0.04) colored by vorticity (side view)




(b) vorticity magnitude at z/R=-0.1 plane

Figure 6.22: Computational wake visualization of low-drag hub.




(a) Varying spanwise locations at $Z/R = -0.05$ (b) Varying vertical locations at $Y/R = -0.10$

Figure 6.23: Comparison of streamwise velocity wake profiles of the baseline hub without the stabilizer.

Figure 6.23 (a) shows the streamwise velocity profile along the spanwise direction at a height of $Z/R$ = -0.05. Figure 6.23 (b) shows the streamwise velocity profile along the vertical direction at $Y/R$ = -0.10. The current simulation results capture a general trend of wake deficit. For example, a more wake deficit is observed at the advancing side of the hub (positive y-axis). However, overll the current results under-predict the wake deficit compared to experimental data. As a validation, available Helios results [72] are included in the comparison as shown in Fig. 6.23 (b). It is observed that the current predictions are very similar with the Helios results especially at near-wake and mid-wake regions. Overall, both simulation results under-predict the wake deficit compared to experimental data.

## 6.4 Slowed Mach-Scaled Rotor at High Advance Ratio

Reducing rotor RPM in forward flight is a key feature in many new high-speed rotorcraft. Reducing the rotor tip speed by slowing down the main rotor can enable a compound helicopter to expand the cruise speed envelope. Slowed rotors have been employed in the Sikorsky X2 and Eurocopter $X^3$ helicopters. However, many aerodynamic phenomena in the high advance ratio flight conditions ($\mu > 0.5$) are still unclear, such as dynamic stall due to the reverse flow on the retreating side.

Recently, a series of wind tunnel tests were performed for a 4-bladed articulated slowed rotor at the Glenn L. Martin wind tunnel at the University of Maryland [78]. In the experiment, pressure data were obtained at 30% rotor radius using blade embedded sensors to calculate the integrated sectional airloads (see Fig. 6.24).

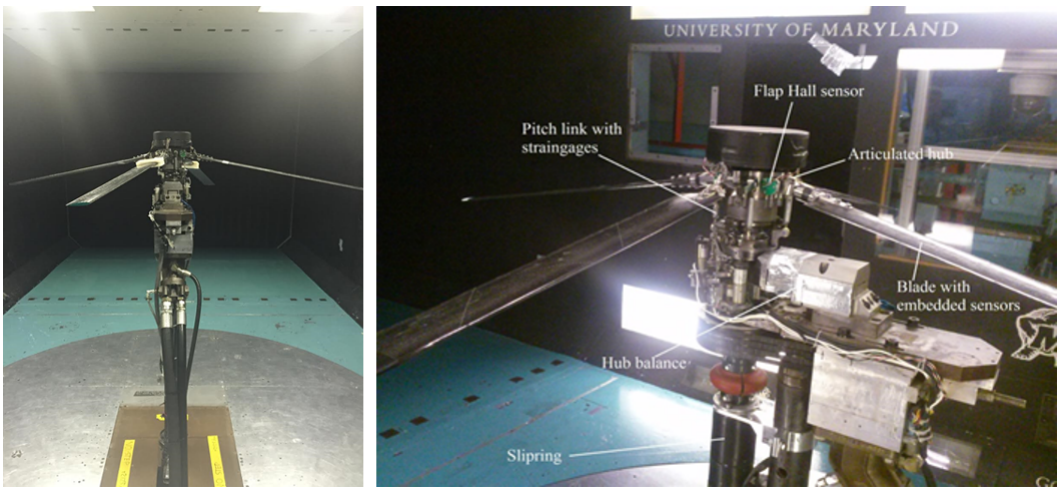



Figure 6.24: Experimental setup for slowed rotor at Glenn L. Martin wind tunnel (left: rear view, right: side view) [78].

In this work, the slowed Mach-scale rotor is simulated using the current CFD

framework to compare against the experimental data and to better understand the aerodynamic phenomena. To reach a trim state, the current CFD method is coupled with a Computational Structural Dynamics (CSD) model to improve the prediction of aeroelastic effects of the rotor blade.

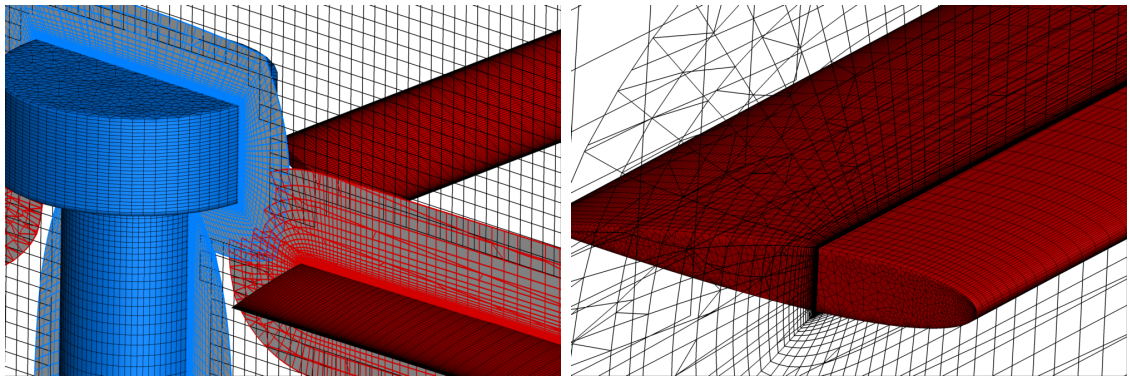
The current CFD-CSD coupled method is achieved using a loose coupling method, which follows the delta-airload method [79]. The airloads and blade deformations are transferred between CFD and CSD at 100 spanwise points. For the CSD code, PrasadUM is used which was developed at the University of Maryland [80]. Using the CSD code, trim states are computed. For the trim approach, the collective angle is prescribed with the experimental value, and only the two cyclis are adjusted for zero hub moment. On the other hand, zero 1/rev flap angles ($\beta_{1c}$, $\beta_{1s}$) was the trim target in the experiment.

Figure 6.25 shows the computational mesh system for the simulation. The rotor radius is 10.63 chords and the root cutout was 16.4% of the rotor radius. The blade is untwisted and untapered with a symmetric NACA0012 airfoil. Each rectangular blade is modeled with a sharp blade tip which is consistent with the experimental blade shape.

As shown in Fig. 6.25 (b), the blade mesh consists of 200 points in the airfoil-wrap direction and 100 points in the span-wise direction. Only the blade tip region was discretized using an unstructured grid. The surface mesh is extruded for the volume mesh using 47 strand layers, which resulted in a total of 4 million hexahedral elements for the rotor. The initial wall-normal spacing is $5 \times 10^{-5}$ chord which corresponds to a $y^+$ of 0.7. As shown in Fig. 6.25 (a), a simplified hub/shaft model

is included in the simulation to consider the effect of wake from the hub/shaft on the blade airloads. 0.51 million hexahedral elements are used for the hub/shaft near-body domain. Using the overset method, the near-body domains are connected with the off-body domain where the uniform grid spacing of 10% chord is distributed around the rotor. All of the domains are computed using the HAMSTR flow solver.

The simulations are conducted at a Reynolds number of $4.72\times10^5$ which is based on the advancing side tip Mach number of 0.327. The freestream Mach number is 0.145 which corresponds to the advance ratio of 0.8. Although the experiments are conducted at various advance ratios ranging from 0.3 to 0.8, the current simulation focuses on the advance ratio of 0.8 at 3° and 11° collective angles. In all domains, flow is assumed as fully turbulent flow and the SA model is used for a turbulence closure. The azimuthal time step of 1.0 degree is used with 15 sub-iterations. In all domains, WENO reconstruction is used for the inviscid flux and the DDLGS method is used for the implicit inversion.



(a) Rotor blade with hub/shaft model (b) Sharp blade tip shape

Figure 6.25: Overset grid system for the slowed rotor simulation.

The instantaneous pitch angle ($\theta$) at each rotor blade azimuth angle ($\psi$) is

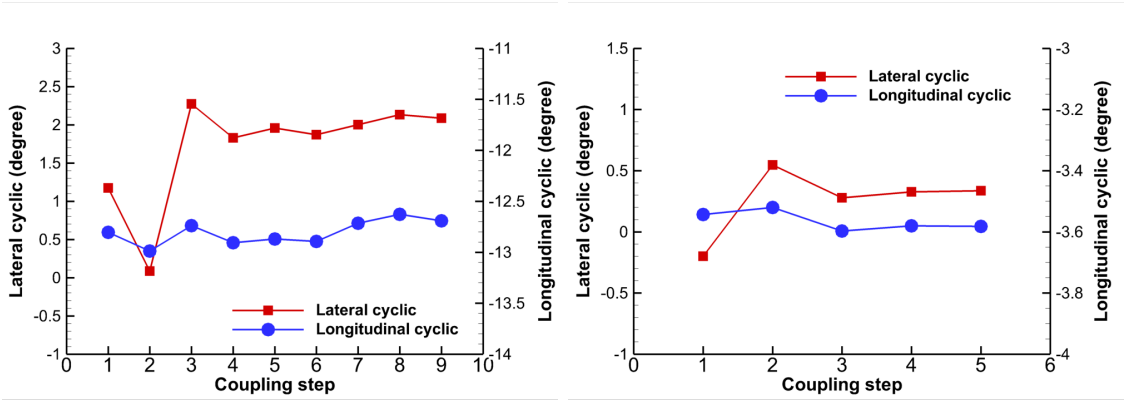
determined using:

$$\theta = \theta_0 + \theta_{1c}\cos\psi + \theta_{1s}\sin\psi \tag{6.3}$$

where $\theta_0$, $\theta_{1c}$, and $\theta_{1s}$ represent the collective, lateral cyclic, and longitudinal cyclic control angles required to trim the rotor, respectively.

During the CFD-CSD coupled method, the collective angle is fixed and only the control cyclic angles are allowed to be corrected based on the CFD airloads.

Figure 6.26 shows the convergence history of control cyclic angles during coupling iterations. The control cyclics are converged within 10 coupling steps in both $\theta_0 = 11°$ and $\theta_0 = 3°$ cases. It is observed that $\theta_{1c}$ experienced more corrections for the trim state, otherwise minor variation was observed in $\theta_{1s}$.




(a) At $\theta_0 = 11°$ (b) At $\theta_0 = 3°$

Figure 6.26: Convergence of control cyclic angle during CFD-CSD coupling steps.

Table 6.4 compares the converged control inputs with the experimental data. Compared to the result from CSD, the current CFD-CSD result shows closer lateral trim angle ($\theta_{1c}$) to the value from experiment in both the cases. However, the

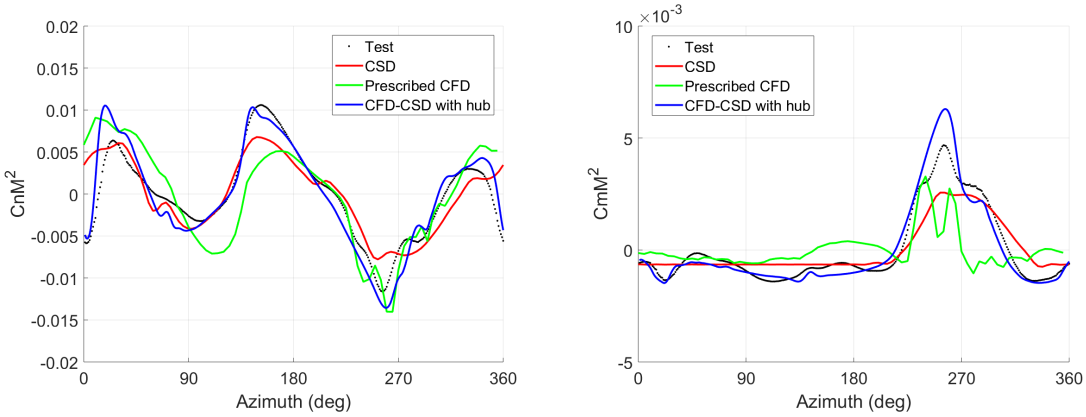
deviation with the experiment still remained. The cause of this discrepancy is not clear. However, the lack of test stand model in the simulation can be one of possible reason (see Fig. 6.24). The longitudinal cyclic in CSD shows a reasonable agreement with the experiment and the values from CFD-CSD method did not change much from the CSD prediction.

Table 6.4: Comparison of control angle for trim state of rotor at $\mu = 0.8$.

| Collective | Cyclics | Experiment | CSD(PrasadUM) | CFD/CSD |
|---|---|---|---|---|
| $\theta_0 = 3°$ | $\theta_{1c}$ | 2.79° | -0.2° | 0.36° |
| | $\theta_{1s}$ | -5.27° | -3.54° | -3.58° |
| $\theta_0 = 11°$ | $\theta_{1c}$ | 5.51° | 1.17° | 2.09° |
| | $\theta_{1s}$ | -13.94° | -12.80° | -12.69° |

Figure 6.27 shows the comparison of sectional airloads between the experiment and predictions at the 30% radial station at $\theta_0 = 11°$. It should be noted that only the periodic components of the results are shown, and the mean values are removed for all airload results. This is because there are a few limitations for a direct comparison: 1. deviation of lateral trim angle, 2. limited pressure sensor locations at $\leq 0.8$ chord. Especially, the lack of pressur sensor near the trailing edge ($> 0.8$ chord) can affect sectional pitching moment.

The CSD aerodynamic model is based on the blade element theory and free wake model. The prescribed CFD result is obtained by assuming rigid blade and using experimental control setting and flap angles. Although the result from CSD shows a good agreement with experiment as a general trend, the details of the reverse flow or vortex interactions are not captured.



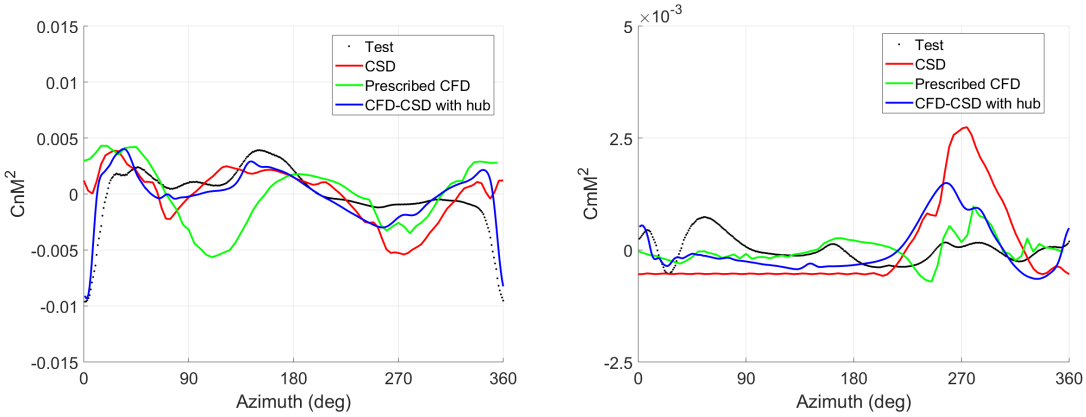

(a) Normal force

(b) Pitching moment

Figure 6.27: 30% rotor radius sectional airload at $\theta_0 = 11°$.

The current CFD-CSD method improves agreement with the experiment over all azimuth angles. First of all, the sectional airloads on the retreating side ($\psi \simeq 270°$) are better predicted where the reverse flow is dominant. The dynamic stall phenomenon is well predicted by showing normal force loss and nose up moment peak. Secondly, the pitching moment variation is better captured than for the CSD result in the advancing side. Thirdly, the hub wake is accurately predicted. The interference with the hub wake results in normal force loss and a change in pitching moment at 0° azimuth angle.

Figure 6.28 shows the comparison of results at the lower collective angle of 3°. The improvement in CFD-CSD result are also observed compared to the results from CSD or prescribed CFD. First of all, the effect of hub wake is more severe than the higher collective angle case. Both loss of normal force and nose up moment at 0° are in a good agreement between CFD-CSD result and experiment. Secondly, the phase of normal force variation is better matched with the experiment. However, CFD-




(a) Normal force

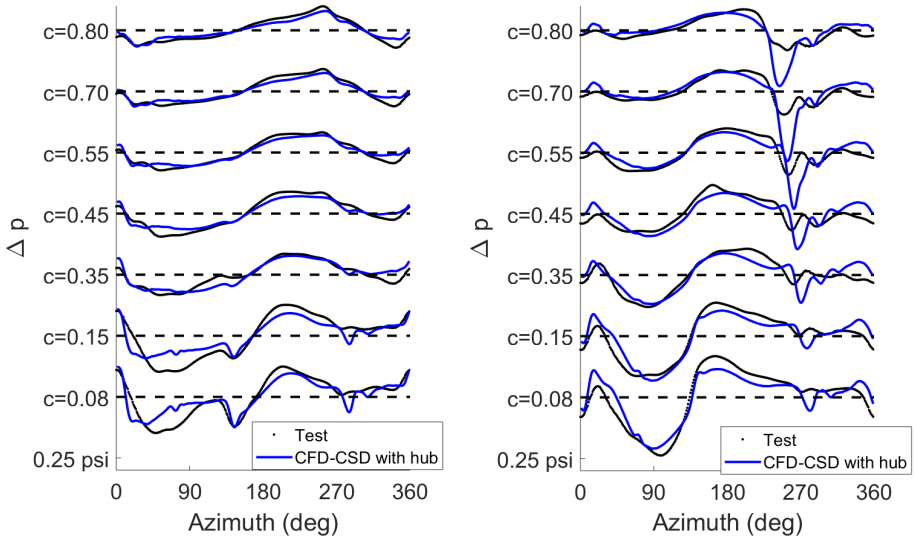
(b) Pitching moment

Figure 6.28: 30% rotor radius sectional airload at $\theta_0 = 3°$.

CSD result still over-predicts the dynamic stall phenomenon on the retreating side. Overall, the agreement with experiment is less satisfactory at the lower collective angle especially at pitching moment.

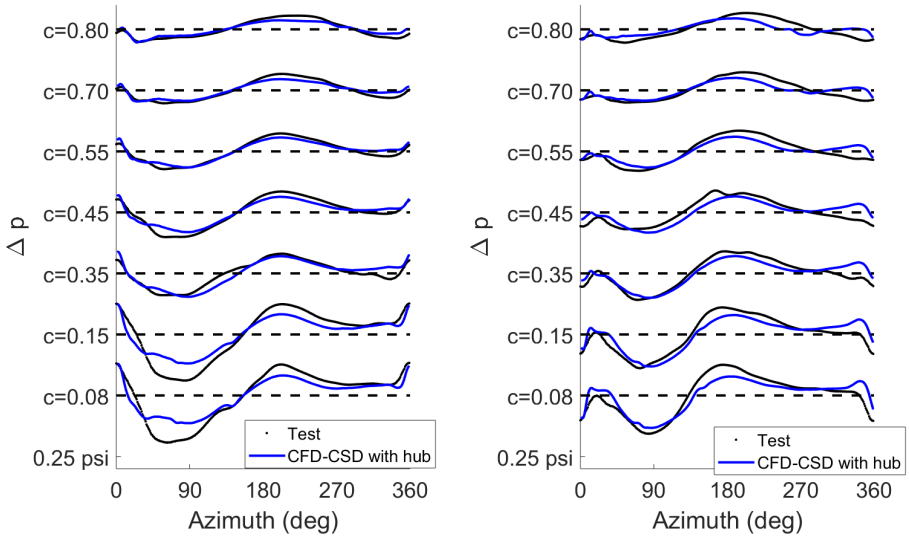
The comparison of surface pressure at 14 points along the chordwise direction at the 30% radial station is performed at both collective angles. At each upper and lower surface of the blade, the seven sensors are located from 0.08 to 0.8 chord. Figure 6.29 compares the unsteady pressure time history between the simulation results and experimental data over the rotor azimuth angles at $\theta_0 = 11°$. It should be noted that only the periodic components of the results are shown, and the mean values are removed for all pressure results. The general trend of pressure variation is captured in the simulation result on both the upper and lower surfaces including the hub wake interaction at the leading edge at 0° and the dynamic stall phenomenon at the lower surface trailing edge at 270°. However, the current result overpredicts the pressure loss due to the dynamic stall at the lower surface.

A similar comparison is conducted for $\theta_0 = 3°$ case as shown in Fig. 6.30. The prediction shows reasonable agreement with the measured data on both the upper and lower surfaces. Some discrepancies observed at the upper surface leading edge on the advancing side and the lower surface trailing edge on the retreating side. These discrepancies might be due to the mismatch of the control cyclic angles between the experiment and the simulation.




(a) Upper surface (b) Lower surface

Figure 6.29: Pressure variations at 30% rotor radius section at $\theta_0 = 11°$.




(a) Upper surface

(b) Lower surface

Figure 6.30: Pressure variations at 30% rotor radius section at $\theta_0 = 3°$.

## 6.5 Summary

In this chapter, the current method within the Python framework is applied to the rotary wing system problems. The summerized observations are provided as below for each problems.

- Pressure Sensitive Paint (PSP) hovering rotor simulations are conducted to validate a capability for aerodynamic performance prediction including a laminar-turbulent boundary layer transition. In this simulation, structured O-O type grid is used for the blade domain. The predicted figure of merits (FM) are reasonably matched with the experimental data at both laminar-turbulent transition and fully-turbulent flow conditions. The increase in the figure of merit is observed in both experiment and simulation when a boundary layer is allowed to have natural transition. This is because the skin friction is much lower in the laminar boundary layer than the turbulent boundary layer.

- NREL Phase VI wind turbine simulations are conducted to validate a capability for predicting interactional aerodynamics between sub-components. Therefore, a full configuration is used in the simulation including blades, nacelle, and tower. The unstructured surface grid and strand volume grid are used for the full configuration of the wind turbine model. Different blade-tower interactions are captured between the upwind and downwind configurations. For the downwind configuration, the acute severe interactions in the blade normal force are observed at all spanwise locations. Otherwise, the interac-

tion is less severe but it occurs over a wider range in the azimuthal direction for the upwind configuration. The characteristics of each interaction are well matched with the experimental data. The effect of freestream wind shear due to the atmospheric boundary layer on rotor performance is studied for the upwind configuration. Highly unsteady blade torque in the azimuthal direction is observed due to the wind shear.

- Rotor hub simulations are conducted to validate a capability for flow simulations around complex geometries. In this simulation, unstructured volume meshes are used for both baseline and low-drag hub models. Strand volume grid is used for the hub stand model. For the validation, unsteady hub drag and associated drag harmonics are compared with the experimental data for both baseline and low-drag hub models. For the baseline hub, current prediction shows a good agreement with experiment and other simulation prediction. However, mean drag is over-predicted for the low-drag hub because of the poor mesh quality near the leading edge of hub arms. In both simulation and experiment, less drag is predicted for the low-drag hub model than the baseline hub model. Wake deficits at near-wake, mid-wake, and far-wake are compared with both expriemental data and other simulation's prediction. Although general trend is matched with the experiment, wake deficits are under-predicted in the current prediction. The under-prediction is also observed in the other simulation's result.

- Slowed Mach-scaled rotor at a high advance ratio simulations are conducted

to validate a capability for elastic rotor blade simulations. Current method is coupled with a Computational Structured Dynamics (CSD) solver to account blade structure deformation and reach trim state. By using a CFD-CSD coupled method, the sectional airloads prediction at 30% rotor radius is improved compared to a pure low-fidelity aerodynamic model (e.g. blade element theory) or CFD simulation with prescribed motions. However, the comparisons are conducted only for the periodic components of the results due to the simplifications in the current simulation.

# Chapter 7: Conclusions

## 7.1 Summary

The overall objective of this research is to develop and validate a novel CFD approach using Hamiltonian path and strand grid for an unstructured grid based compressible Reynolds Averaged Navier–Stokes (RANS) solver. The method entails the identification of Hamiltonian paths, which are created on pure quadrilateral elements to represent two distinct surface coordinate directions, and strands to represent the wall normal direction. Starting from a purely unstructured surface mesh, these line structures are identified uniquely and robustly for a three-dimensional domain.

The path identification is also possible for an initially mixed element unstructured volume elements. In this case, each element (tetrahedron and prism) is divided into hexahedra and the same algorithm used for the surface mesh is valid for the path identification. As a result, each hexahedron is visited by three distinct Hamiltonian paths. Ensuring the even number of edges/faces per element makes the nesting for the factorization method to be always satisfied. Therefore, the novel approach allows for line implicit methods and stencil-based discretization along the line structures, which is similar to the strategies of a structured grid based flow solver to be applied

to an unstructured mesh.

Initially, the concept of Hamiltonian path was developed in the reference [14]. However, the application of the method was limited to two-dimensional laminar flow and executed on a serial processor. To be applied to realistic problems, this methodology had to be further improved and verified in both parts of mesh generation and flow solver.

## Improvement on mesh generation

1. The generation of Hamiltonian paths is extended to general two-dimensional and three-dimensional unstructured grids. The prerequisite for the robust line identification is the generation of pure quadrilateral or hexahedral elements from mixed elements which can be achieved using quadrilateral or hexahedral subdivision. Starting from any midpoint edge/face, the resulting Hamiltonian paths are grown by connecting midpoints of opposite edge/face until all the edge/face are part of the loop. This process allows the Hamiltonian paths to traverse both structured and unstructured grids robustly and uniquely. One interesting feature is the self-crossing path, which occurs when a quadrilateral element is surrounded by triangles before subdivision. In this case, all edges/faces for each element are still part of the loops and the current formulation of the flow solver can handle the situation without special treatment.

2. Two-dimensional Hamiltonian paths are extended to three-dimension by applying strand grids. The strand grids emanate from the cells on the surface of the

object and are extruded in the wall-normal direction, thus they represent the third "out-of-plane" spatial direction. The nature of a strand grid allows for the generation of line structures in the wall-normal direction and preservation of Hamiltonian paths on the surface-direction in each strand layers. Also, each strand easily remains in a single domain without being broken during the domain decomposition process.

3. The mesh generation code is parallelized to be executed on distributed computer memory system using a message passing interface (MPI). METIS [20] is used for graph partitioning and the number of cells is evenly balanced in each partitioned domain. The partitioning process was applied to either surface domain or volume domain for the case of strand grids or an unstructured volume mesh, respectively.

4. The mesh system is extended to utilize overset meshes. This overset technique allows for multiple mesh systems, which consists of a near-body Hamiltonian/Strand grid and off-body Cartesian nested meshes, for example. The connectivity between the different overset meshes is computed using a topology independent overset grid assembler (TIOGA) [39]. The overset meshes are particularly useful with the strand grids for the near-body domain. This is because the near-body domain can transition to the off-body Cartesian domain before the strands cross each other in concave regions.

## Improvement on flow solver technique

1. The capability of the flow solver is provided to simulate viscous flows. The viscous fluxes are computed using either a finite difference scheme or least-squares approach which both provide second-order accuracy. In order to predict turbulent flow features, the one equation Spalart-Allmaras (SA) and the two equations Menter Shear Stress Transport (SST) turbulence models have been implemented and validated through the comparison against several well-established flow solvers for representative cases. As a hybrid RANS/LES method, the delayed detached eddy simulation (DDES) method was integrated to the SA model for the case of massively separated flows. The accuracy of the RANS simulation is further improved by allowing for laminar-turbulent boundary layer transition. The existing transition model has been extended for crossflow instability on three-dimensional bodies and evaluated though a variety of test cases within the current framework. The transition model is coupled with the SA turbulence model.

2. Various reconstruction schemes are applicable for the different types of grid structure. Both stencil- and gradient-based reconstruction methods have been tested in the current grid system to evaluate their accuracy. The identification of line-structures on the unstructured grid facilitates the stencil-based reconstructions along those lines. In the current work, three stencil-based reconstruction schemes are available: third-order MUSCL, fifth-order WENO and CRWENO. Although the higher-order schemes provide less dissipation error than the lower-order scheme, the formal order of accuracy of each scheme are not obtained on unstructured grids.

The gradient-based reconstruction scheme using least-squares is also enabled in the current method. As a conventional method for unstructured grids, this formulation provided second-order accuracy through the Method of Manufactured Solution (MMS) on the unstructured surface mesh. Alternatively, a combined least-squares for the wall-tangential directions and MUSCL/WENO scheme for the strand direction is proposed for the Hamiltonian/Strands grid system.

3. The efficiency of the flow solver is validated through comparison studies with the well-established structured grid based solver, OVERTURNS [46]. For two- and three-dimensional comparison studies, both structured and unstructured grids were used in the current flow solver. The performance of Diagonally Dominant Line Gauss-Seidel (DDLGS) method in the current solver is compared with various line-implicit methods (LU-SGS, DADI, and DDGLS) in OVERTURNS. Comparable solution residual convergence was obtained in the current solver as with OVERTURNS in terms of CPU time at various flow conditions: transonic inviscid and fully turbulent flows.

4. The performance of the current line-implit method (DDLGS) is validated by comparing the residual convergence rate with the point-implicit method (point Gauss-Seidel) which is a traditional implicit method for typical unstructured grid based solvers. The comparisons were conducted using various types of grid: two-dimensional unstructured, three-dimensional strand, and three-dimensional unstructured volume meshes. Overall, the line-implicit method outperformed the point-implicit method for all of the test meshes.

5. The Generalized Minimum Residual (GMRES) method is implemented as

another option for the implicit time integration method. GMRES, a Krylov subspace class of methods, has strong convergence properties and shows better solution convergence rate compared to line-implicit methods. In the current work, GMRES method requires the preconditioned step which is performed using Diagonally Dominant Line Gauss-Seidel (DDLGS). GMRES does not exhibit as much improvement for turbulent flow simulations as for inviscid or laminar flow simulations. This is because the current GMRES is used only for the mean flow equations and the turbulence model is solved using the line-implicit method, separately.

6. The flow solver has been extended for time-accurate unsteady flow simulations with grid motion, which is required for the simulation of rotary-wing systems. A second order time-accurate method with dual-time-stepping strategy was explored. Through various test cases, reasonable unsteady residual convergence rates were observed during the dual-time-stepping using the CFL number scaled pseudo time step size. Grid motion terms are augmented into both the LHS and RHS of the flow solver assuming either first- or second-order discretization in time.

7. The flow solver is parallelized using a message passing interface (MPI). The concept of ghost cells is adopted along the boundary between sub-domains, which enables to apply the same reconstruction scheme and implicit operator between the interior and boundary domains. In the current work, a maximum of three ghost layers are used for fifth-order reconstruction scheme on the RHS and a single ghost layer is used for the implicit operator on the LHS. A strong scalability test was conducted to validate the parallelized flow solver performance using up to 250 CPUs. It was observed that the solution residual convergence rate is not affected by

the number of processors in both the mean flow and turbulence model. Also, about 90 % parallel efficiency is obtained at 250 CPUs in terms of the speed up.

8. The flow solver is originally developed as an alternative for near-body unstructured grid solvers within a multi-mesh/multi-solver paradigm. Therefore, the flow solver is wrapped in Python to allow for ease of integration with the other codes. Through the Python-based interface, the current CPU-based flow solver is coupled with a GPU-based structured grid solver as a heterogeneous, overset solution framework. Through the lightweight Python-based framework, the communication between flow solvers can be performed efficiently by sharing data pointers without data transfer using file I/O. The coupled CFD framework has been applied to various interactional aerodynamic flow problems of the rotary wing system with the complex geometries and the multiple sub-components.

## Observations from rotary wing simulations

The extended novel method has been integrated in the Python framework for multi-mesh/multi-solver paradigm and applied to helicopter/wind turbine flow simulations.

1. Pressure Sensitive Paint (PSP) hovering rotor simulations are conducted to study a effect of boundary layer transition on a hovering rotor performance. The flow around the rotor blade is computed using the current method and the O-O type structured grid is used for the domain. The blade domain is connected with the off-body Cartesian domain where the in-house OVERTURNS code is used. As

results, the increase in the figure of merit is observed by allowing the boundary layer to have natural transition at three different collective angles (6°,8°, and 10°). The same trends are observed in the experimental data and the other simulation results.

2. NREL Phase VI wind turbine full configuration including blades, tower, and nacelle is simulated to predict interactional aerodynamics between the sub-components. All of the near-body domains with strand grids are computed using the current method and the domains are connected with the off-body wind tunnel domain which is computed using GPU-based structured flow solver (Garfield). As results, two different types of blade-tower interaction are captured between the upwind and downwind configurations. The characteristics of each interaction are well matched with the experiment. Additionally, the effect of freestream wind shear due to atmospheric boundary layer on rotor performance is studied for the upwind configuration.

3. Penn State University (PSU) rotor hub simulations are conducted to validate the current method for problems with complex geometries. The unstructured volume mesh is used for the hub model and the strand grid is used for the test stand model. The unsteady hub drag predictions are compared against the measured drags for both baseline and low-drag hub models. For the baseline hub model, a good agreement is observed in terms of both mean value and unsteady trend. For the low-drag hub model, the predicted mean drag is over-predicted by 14% although the unsteady trend is reasonably matched with the experiment counterpart. The wake deficits are validated by comparing the current result with other simulation result. The wake profiles are well matched between the simulation results.

4. Slowed Mach-scaled rotor at the advance ratio of 0.8 is simulated to validate a capability for trimmed elastic rotor blade simulation. Current method is coupled with CSD solver as a CFD-CSD loose coupled method. The sectional airloads at 30% rotor radius are compared between the predictions and experimental data. The improved predictions are observed using the CFD-CSD coupled method compared to the predictions using either CSD alone or CFD alone. However, the comparisons are conducted only for the periodic components of the results.

## 7.2 Contributions

The contributions from the current study are listed below.

- The in-house mesh generation code is provided which runs in parallel using multiple processors. The mesh generation code generates novel unstructured grid data for both two- and three-dimensional simulations. Either initial surface mesh data or volume mesh data is required as an input file for strand grids or unstructured volume grids generation, respectively.

- The in-house (two- and three-dimensional) flow solvers are provided which can use pure line-based methods in both flux evaluation and implicit time integration even on unstructured grids. Efficiency of the current method has been demonstrated with comparable execution time as a structured grid flow solver on a structured grid and less than a factor of two times slower on an unstructured grid with similar grid resolution. The developed flow solver has been integrated into in-house Python CFD framework for the demonstration

of multi-mesh/multi-solver paradigm.

- The current solution algorithm explores pure line-based methods for three-dimensional simulations for the first time, which can start from either unstructured surface mesh or unstructured volume mesh. Therefore, not only it can be applied to various flow simulations, but it also can be provided as a basic framework for the future solution algorithm improvements.

## 7.3 Recommendations for Future Work

1. The current subdivision process generates the new points on the edges/faces of the initial element. For example, one grid point is newly created on each edge using quad-level 0 subdivision and the number of new points increases as the quad-level increases. The newly created points especially on the object surface should be flushed carefully to be located on the actual object. The analytic equation for the surface definition can be used to flush the points. However, the unstructured meshing typically starts with a CAD geometry and the analytic equation for the geometry is often not available. Therefore, a robust flushing technique for the arbitrary geometry needs to be developed inside the mesh generation code.

2. The current stencil-based reconstruction schemes on the finite volume formulation limit the solution order of accuracy. On irregular unstructured grids, the accuracy of WENO scheme decreased to first-order because the varying curvature and grid spacing along the loop are not considered in the formulation. In order to obtain formal high-order accuracy (higher than second-order) on an irregular un-

structured grid, other existing methods can be explored: quadratic reconstruction, finite-element type discretization, or spectral volume method. As an alternative a conservative finite difference approach with grid metrics should also be examined.

3. The current GMRES implementation is limited in the mean flow equations. Therefore, the strong convergence property of GMRES is not fully obtained with the addition of the turbulence model. Once the transport equation of the turbulence model is integrated into the current GMRES implementation, better convergence rate is expected for the turbulent flow simulations.

4. The flow solver can be extended to the adjoint formulation for sensitivity capability, which is desirable for design optimization. The adjoint method is well known and advantageous for cases with a large number of design parameters. Because the current method can use an unstructured grid and handle a deformed mesh system, the surface of the model can be deformed easily during the optimization process.

5. The current CPU-based flow solver can execute on multiple graphic processing units (GPUs) for further speed up in wall clock time (by higher than 5 times, typically). For this, the entire source code needs to be re-written into a GPU programming language, such as CUDA (which is NVIDIA's C-based language). Since line structures are identified the resulting code will be in may ways similar to that for an implicit structured solver on the GPUs (such as Garfield [54]).

# Appendix A: Extension of Laminar-Turbulent Transition Model

The present transition model formulation ($\gamma - \overline{Re_{\theta t}}$−SA) is described in this appendix, a detailed description of the model can be found in the previous work [36–38]. The transport equation for the intermittency, $\gamma$, is given by:

$$\frac{D(\rho\gamma)}{Dt} = P_\gamma - D_\gamma + \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\frac{\partial \gamma}{\partial x_j}\right] \tag{A.1}$$

$$P_\gamma = \rho F_{onset} G_{onset} \max\left[\frac{\Omega}{F_{length}}, \frac{1.0}{F_{length,min}}\right], \quad \text{if } \gamma > 1.0,\ P_\gamma = (1-\gamma) \tag{A.2}$$

$$D_\gamma = \rho\Omega\gamma(1.0 - G_{onset}) \tag{A.3}$$

where $G_{onset}$ is computed as summation of $F_{onset}$ along a grid line in a wall normal direction. The purpose of $G_{onset}$ is to turn on/off the intermittency production term uniformly along the height of a transitional boundary layer at the transition onset location. The non-local variable $G_{onset}$ is computed easily by using the strand grid concept, even from an unstructured surface grid, within the current solver strategy.

$$F_{onset} = \max(F_{onset2} - F_{onset3}, 0) \tag{A.4}$$

$$F_{onset1} = \frac{Re_v}{2.193 Re_{\theta c}} \tag{A.5}$$

$$F_{onset2} = \min(\max(F_{onset1}, F_{onset1}^2), 4.0) \tag{A.6}$$

$$F_{onset3} = \max(2 - (0.25 R_T)^3, 0) \tag{A.7}$$

$$Re_v = \frac{\rho d^2 S}{\mu}, \quad R_T = \frac{\mu_t}{\mu}, \quad Re_{\theta c} = \alpha \overline{Re_{\theta t}}, \quad \alpha = 0.62 \tag{A.8}$$

$$F_{length} = 10.0, \quad F_{length,\min} = 2.5 \tag{A.9}$$

The transport equation for the transition momentum thickness Reynolds number, $\overline{Re_{\theta t}}$, is given by:

$$\frac{D\left(\rho \overline{Re_{\theta t}}\right)}{Dt} = P_{\theta t} + \frac{\partial}{\partial x_j}\left[2.0(\mu + \mu_t)\frac{\partial \overline{Re_{\theta t}}}{\partial x_j}\right] \tag{A.10}$$

$$P_{\theta t} = 0.03 \frac{\rho}{t}(Re_{\theta t} - \overline{Re_{\theta_t}})(1.0 - F_{\theta t}) \tag{A.11}$$

$$t = \frac{500\mu}{\rho U^2} \tag{A.12}$$

$t$ is a timescale, which is present for dimensional reason [35] and $U$ is local velocity which is defined as in Eq. A.22.

$$F_{\theta t} = \min\left(e^{-(\frac{d}{\delta})^4}, 1.0\right) \tag{A.13}$$

$$\theta_{BL} = \frac{\overline{Re_{\theta t}}\mu}{\rho U}; \quad \delta_{BL} = 7.5\theta_{BL}; \quad \delta = \frac{50\Omega d}{U}\delta_{BL} \tag{A.14}$$

$Re_{\theta t}$ is computed by solving experimental correlations as shown in Eq. A.15 and A.16 iteratively, such as by the Newton-Raphson method.

$$Re_{\theta t} = \begin{cases} [1173.51 - 589.428Tu + \frac{0.2196}{Tu^2}]F(\lambda_\theta), & Tu \leq 1.3 \\ 331.50[Tu - 0.5658]^{-0.671}F(\lambda_\theta), & Tu > 1.3 \end{cases} \tag{A.15}$$

$$F(\lambda_\theta) = \begin{cases} 1 - [-12.986\lambda_\theta - 123.66\lambda_\theta^2 - 405.689\lambda_\theta^3]e^{-\left[\frac{Tu}{1.5}\right]^{1.5}}, & \lambda_\theta \leq 0 \\ 1 + 0.275[1 - e^{[-35\lambda_\theta]}]e^{-\left[\frac{Tu}{0.5}\right]}, & \lambda_\theta > 0 \end{cases} \tag{A.16}$$

The pressure gradient parameter, $\lambda_\theta$, and the momentum thickness, $\theta$, are given by:

$$\lambda_\theta = \frac{\rho\theta^2}{\mu}\frac{dU}{ds}, \; \theta = \frac{Re_{\theta t}\mu}{\rho U} \tag{A.17}$$

$$\frac{dU}{ds} = \frac{u}{U}\frac{dU}{dx} + \frac{v}{U}\frac{dU}{dy} + \frac{w}{U}\frac{dU}{dz} \tag{A.18}$$

$$\frac{dU}{dx} = \frac{1}{2U}\left[2u\frac{du}{dx} + 2v\frac{dv}{dx} + 2w\frac{dw}{dx}\right] \tag{A.19}$$

$$\frac{dU}{dy} = \frac{1}{2U}\left[2u\frac{du}{dy} + 2v\frac{dv}{dy} + 2w\frac{dw}{dy}\right] \tag{A.20}$$

$$\frac{dU}{dz} = \frac{1}{2U}\left[2u\frac{du}{dz} + 2v\frac{dv}{dz} + 2w\frac{dw}{dz}\right] \tag{A.21}$$

$$U = \sqrt{u^2 + v^2 + w^2} \tag{A.22}$$

## A.1 Crossflow-Induced Transition

The current $\gamma - \overline{Re_{\theta t}}$−SA transition model is extended for a three-dimensional boundary layer by integrating the model of crossflow-induced transition, which was proposed by Muller and Herbst [81]. This crossflow transition model is based on six calibration constants and was originally developed to extend the $\gamma - \overline{Re_{\theta t}}$ formulation. The additional production term as shown in below equation is only required in the transport equation for $\overline{Re_{\theta t}}$ as a sink term. The additional production term acts inside the boundary layer by lowering $\overline{Re_{\theta t}}$, which results in a decrease of $Re_{\theta c}$. Then this destabilizes the boundary layer further to trigger crossflow transition.

$$P_{CF} = -min(max[0, (\frac{\rho}{1000 \cdot t} \cdot (\frac{Re_H}{6})^{c1} \cdot (Re_\Omega)^{c2} \cdot (\frac{12\theta}{d})^{c3} - c4) \cdot c5], c6) \cdot c7 \tag{A.23}$$

The set of constants used in the current study are the same values with reference [81] except the constant $c4$ and $c7$, which were changed through NLF(2)-0415 swept wing simulations (see table A.1).

Table A.1: Combination of constants for crossflow model.

| c1 | c2 | c3 | c4 | c5 | c6 | c7 |
|---|---|---|---|---|---|---|
| 0.548 | 0.1912 | -0.298 | 5.0 | 60.0 | 1666.5 | 1.5 |

In the crossflow model, the first term is the Reynolds number based on local helicity.

$$Re_H = \frac{\theta}{\nu}\sqrt{\theta H} \tag{A.24}$$

where $H$ (local helicity) is defined as $H = |u_i \cdot \omega_i|$ and $\theta$ is used from Eq. A.17. For

two-dimensional flow, the local helicity naturally equates to zero and this results in the additional production term, $P_{CF}$, to be zero as well.

Second term is the modified vorticity Reynolds number and defined as below.

$$Re_{\Omega} = \frac{\rho d^2}{\mu}\Omega \tag{A.25}$$

The vorticity Reynolds number is similar to the formulation of Langtry and Menter [35], though it is rather based on the magnitude of vorticity for the accuracy at the stagnation point [81].

The current extended transition model was applied to the infinite swept NLF(2)-0415 wing to validate crossflow transition. The wing has a geometric sweep angle of 45° and the angle of attack was fixed as -4° in the experiment [82]. For the validation, a total of six different Reynolds number were tested which vary from 1.92 million to 3.73 million. The FSTI was set as 0.05 %. To simulate an infinite swept wing configuration, a structured C-type airfoil mesh was extended in the spanwise direction by 0.3 chord length with 0.005 chord length grid spacing. At each end of the boundary surface, the periodic boundary condition was imposed. The number of points on the airfoil was 272 and the wall normal spacing was $1 \times 10^{-5}$ which corresponds to a $y^+$ value from 0.8 to 1.5 depending on the test Reynolds number.

Figure A.1 (a) shows the chordwise skin friction coefficient distribution on the upper surface (pressure side) of the NLF(2)-0415 wing. The sharp increases in skin friction (transition onset location) are clearly shown at all the Reynolds numbers and the locations move towards the leading edge as the Reynolds number increases. Also, note that the friction is close to zero right before transition at the lowest

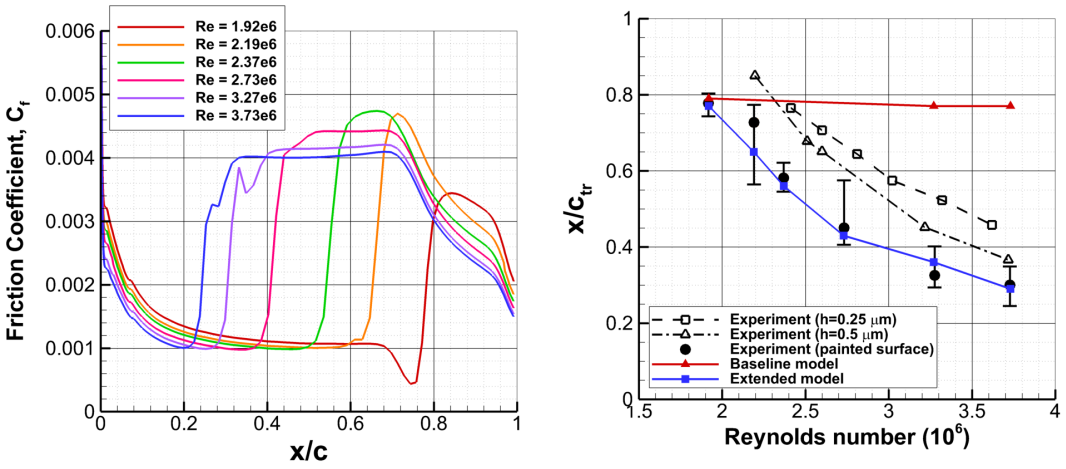
Reynolds number. This indicates the separation-induced transition, otherwise the transitions occur due to pure crossflow instability at the other Reynolds numbers.

Figure A.1 (b) shows the comparison of transition onset location against experimental data [82] on the upper surface of the NLF(2)-0415 wing. It should be noted that the stationary crossflow transition is influenced by the surface roughness, and three different surface characteristics were tested in the experiment: painted surface which has peak-to-peak roughness of 9 $\mu m$ and two polished surfaces which have rms roughness of 0.25 and 0.5 $\mu m$ respectively.

The current crossflow model was calibrated based on the experimental data of the painted surface and the model is not able to consider the effect of different roughness heights on the crossflow transition. In the comparison, the transition onset location is defined as the point of intermittency crossing 0.5 on the surface, and the current prediction has a good agreement with experiment (painted surface) over the range of Reynolds numbers as shown in Fig. A.1 (b).

## A.2 Surface Roughness-Induced Transition

The current transition model has been further extended by incorporating the effect of surface roughness on the boundary layer transition process [85]. In this work, an additional transport equation for the "roughness amplification ($A_r$)" parameter is used which was originally developed to be coupled with the $\gamma - \overline{Re_{\theta t}}$ model as proposed by Dassler et al [83].




(a) Skin friction profile on the upper surface (b) Transition onset location on the upper surface

Figure A.1: NLF(2)-0415 infinite swept wing simulation results.

$$\frac{\partial\left(\rho A_r\right)}{\partial t}+\frac{\partial\left(\rho U_j A_r\right)}{\partial x_j}=\frac{\partial}{\partial x_j}\left[\sigma_{ar}\left(\mu+\mu_t\right)\frac{\partial A_r}{\partial x_j}\right] \tag{A.26}$$

where $\sigma_{ar}=10$.

The roughness amplification ($A_r$) variable is treated as an additional non-physical quantity that will be produced at rough surface boundaries. The quantity is transported through the flow field by the convection and diffusive terms of Eq. A.26. This behavior enables the flow history effects to be taken into account. Through interaction of $A_r$ with the $\overline{Re_{\theta_t}}$ transport equation, the transition onset is triggered by reducing $\overline{Re_{\theta_t}}$. It should be noted that the $A_r$ equation does not include a production term, alternatively the distribution of $A_r$ is determined with a boundary condition at rough walls as given by:

$$A_r = 8.0 \times k^+ \tag{A.27}$$

where the user inputs a non-dimensional equivalent sand grain roughness height.

$$k^+ = \sqrt{\frac{\tau_w}{\rho_w}} \frac{k_s}{\nu} \tag{A.28}$$

Thus, the current model is able to simulate different roughness heights on the object and to capture the effect on transition due to varying roughness height over the surface. Once $A_r$ is solved throughout the entire computational domain, the $F_{A_r}$ expression is given by:

$$F_{A_r} = \begin{cases} c_{Ar2}(A_r)^3, & \text{if } A_r < C_{A_r} \\ c_{Ar3}\left(A_r - C_{A_r}\right) + c_{Ar2}\left(C_{A_r}\right)^3, & \text{if } A_r \geq C_{A_r} \end{cases} \tag{A.29}$$

The function is switched at $C_{A_r} = \sqrt{c_{Ar3}/3c_{Ar2}}$ to allow for a smooth transition between the cubic and linear functions. The same model parameters are used as Langel et al. [84] which is given by:

$$c_{Ar2} = 0.0005 \qquad c_{Ar3} = 2.0 \tag{A.30}$$

The final production term of the transport equation for transition momentum thickness Reynolds number, $\overline{Re_{\theta t}}$, is given by:

$$P_{\theta t} = 0.03 \frac{\rho}{t} \left[ (Re_{\theta t} - \overline{Re_{\theta_t}})(1.0 - F_{\theta t}) - F_{A_r} \right] + P_{CF} \tag{A.31}$$

where the $F_{A_r}$ and $P_{CF}$ are the corrections for roughness-induced and crossflow-induced transition, respectively. It should be noted that $\overline{Re_{\theta t}}$ was limited with a

minimum value of 20.0 to avoid nonphysical overshoots of shear stress for the cases with high Reynolds number and high roughness height [83].

The extended transition model for the surface roughness has been validated through the same zero pressure gradient flat plate test case as for the original validation of the model by Dassler et al [83]. Various equivalent sand grain roughness heights have been applied along the no-slip boundary, which ranges from $Re_k = 0$ (smooth surface) to $Re_k = 381$. The $Re_k$ (the equivalent sand grain roughness height Reynolds number) was achieved as given by:

$$Re_k = \frac{Uk_s}{\nu} \tag{A.32}$$

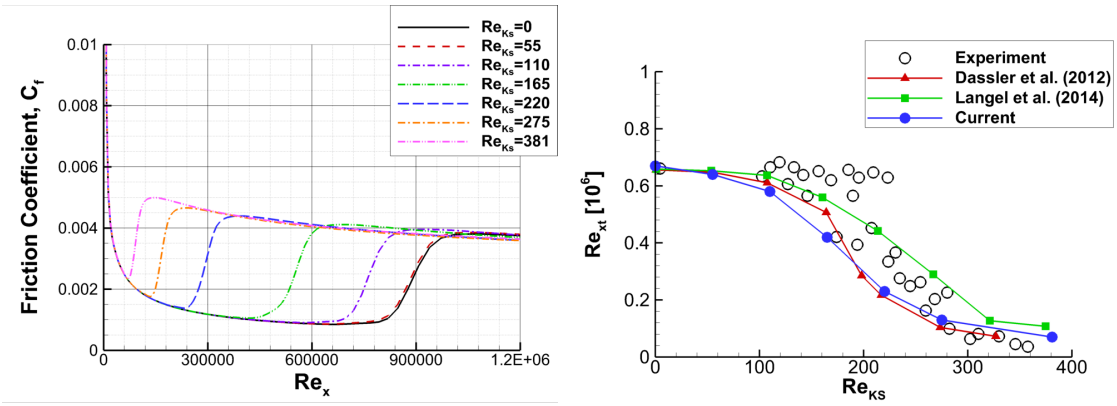
According to Dassler et al. [83], FSTI was set in a way that the predicted transition onset on the smooth wall is the same as in the experiment because FSTI was not given in the experiment. In the current study, FSTI was set in the same way and 1.0 % was chosen for the simulations, which is somewhat higher than the value (0.91 %) from Dassler et al.

Figure A.2 (a) shows the skin friction ($C_f$) distribution along the wall with varying $Re_k$. The initial wall normal spacing for the case was $2 \times 10^{-6}$ which corresponds to a $y^+ = 0.1$. As expected, the transition onset location moves upstream as roughness height increases.

The predicted transition onset locations were compared with experiment and other numerical simulation results [83, 84] as shown in Fig A.2 (b). The transition onset location, $Re_{xt}$ was defined in this case as the point of minimum skin friction

as in Dassler et al. [83]. The current model can predict the overall trends of onset location when the results are compared with the scattered experimental data. It is also observed that the effect of surface roughness on skin friction becomes important starting from $Re_{xt}$ of 110, which is similar with the experimental data.

The sensitivity of aerodynamic coefficients to varying surface roughness was compuated for both a rotorcraft fuselage (ROBIN-Mod7) and an airfoil (SC1095) using the extended transition model in the previous study [85].




(a) Skin friction distribution

(b) Comparison of transition onset locations

Figure A.2: Transitional flow over zero pressure rough surface flat plate simulation results.

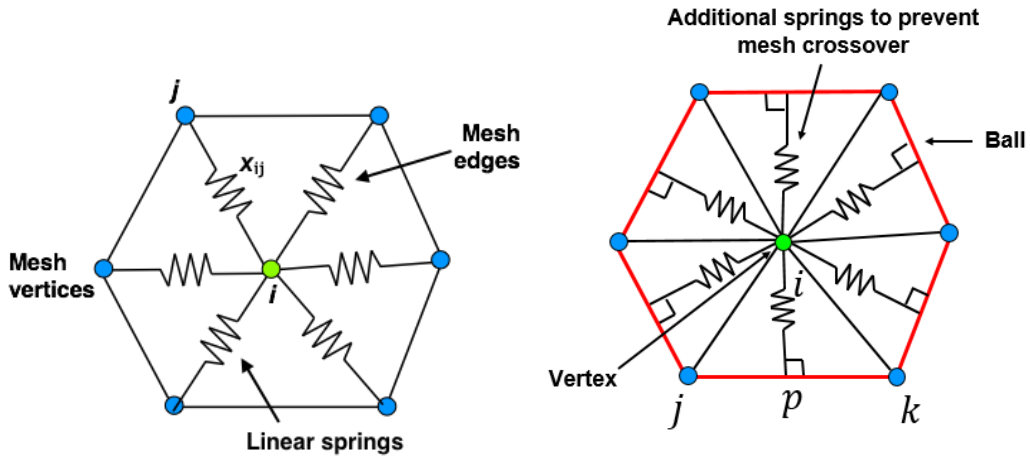
# Appendix B: Mesh Deformation Technique

The mesh deformation technique for current unstructured grids is described in this appendix. This technique is necessary for moving boundary problems, such as fluid-structure interaction or shape optimization to avoid repeated mesh generation every time after the boundary points are re-positioned. The most widely used mesh deformation technique for traditional unstructured grids is the spring analogy method [86]. In this method, each edge is replaced by a spring, whose stiffness is inversely proportional to the edge length. For a given displacement of the boundary points, the displacement of the interior mesh points are determined by solving static force equilibrium equations of the spring network system. This spring analogy is applied in the current mesh system [87].

## B.1 Spring Analogy

For the current quadrilateral mesh system in two-dimensions and hexahedral mesh system in three-dimensions, each element was assumed as triangles or tetrahedra by sub-dividing a quadrilateral into two triangles or a hexahedron into five tetrahedra. With the assumption, the traditional spring analogy for a triangle or a tetrahedron can be directly applied in the current mesh system.

Figure B.1 shows schematics of the linear spring method and ball-vertex spring method. In both methods, each edge of the mesh is considered as a linear spring whose stiffness is inversely proportional to edge length




(a) Linear spring (b) Ball-vertex spring

Figure B.1: Mesh deformation technique using: (a) Linear spring analogy, and (b) Ball-vertex analogy.

$$k_{ij} = \frac{1}{\sqrt{x_{ij} \cdot x_{ij}}} \tag{B.1}$$

where $x_{ij}$ indicates the vector from vertex $i$ to vertex $j$. Therefore, longer edges become "softer" while shorter edges are "stiffer." The force applied to the vertex $i$ and $j$ because of their displacements of $u_i$ and $u_j$ can be written as

$$f_{i,ij} = k_{ij}(u_j - u_i) \cdot i_{ij} = -f_{j,ij} \tag{B.2}$$

where $i_{ij}$ is the unit vector of $x_{ij}$.

For static equilibrium, the total force acting on the vertex $i$ should be zero.

$$\sum_{j=1}^{n_e} f_{i,ij} = 0 \tag{B.3}$$

where $n_e$ is the number of edges that are connected to vertex $i$.

The static equilibrium equation can be rewritten in the matrix formulation.

$$\begin{pmatrix} -k_{ij}\mathbf{I} & k_{ij}\mathbf{I} \\ k_{ij}\mathbf{I} & -k_{ij}\mathbf{I} \end{pmatrix} \begin{pmatrix} u_i \\ u_j \end{pmatrix} = \begin{pmatrix} f_{i,ij} \\ f_{j,ij} \end{pmatrix} \tag{B.4}$$

where $\mathbf{I}$ is a $n_{\text{dim}} \times n_{\text{dim}}$ identity matrix and $n_{\text{dim}}$ is the number of dimensions.

By applying to all mesh vertices, the global force-displacement equation is given by:

$$\mathbf{KU} = \mathbf{b} \tag{B.5}$$

The vector $\mathbf{b}$ includes boundary mesh displacements, and by solving this system of equations, the unknown interior mesh displacements can be obtained. The matrix $\mathbf{K}$ is a block matrix, and each size of block is a $n_{\text{dim}} \times n_{\text{dim}}$.

However, the linear spring analogy method often fails due to possible face-vertex intersection when the boundary movements are not small compared to the local mesh size. To overcome this limitation, a ball-vertex spring method [86] is utilized by employing additional linear springs, which connect the vertex $i$ to the closest point $p$ in the plane of opposite face, as shown in Fig. B.1 (b).

$$x_p = x_i + x_{ij} \cdot n \tag{B.6}$$

where $n$ is the unit normal vector of edge $jk$. The existence of spring $ip$ creates an additional spring force that is applied to both vertices $i$ and $p$.

$$f_{ip} = k_{ip}(u_p - u_i) \cdot i_{ip} = -f_{p,ip} \tag{B.7}$$

where $k_{ip}$ is the stiffness of spring $ip$, $u_p$ and $u_i$ are the displacements of vertices $p$ and $i$, respectively. As vertex $p$ is located on the edge $jk$, $u_p$ can be written using area coordinates $\xi$ and $\eta$ of the face as follow:

$$u_p = \xi u_j + \eta u_k \tag{B.8}$$

where $\xi + \eta = 1$.

The forces at the vertices $j$ and $k$, which are induced by the spring $ip$, can be written as follows:

$$\begin{aligned} f_{j,ip} &= \xi f_{p,ip} \\ f_{k,ip} &= \eta f_{p,ip} \end{aligned} \tag{B.9}$$

Finally, Eq. B.7 and Eq. B.9 can be written as the following matrix form as

$$\begin{Bmatrix} f_{i,ip} \\ f_{j,ip} \\ f_{k,ip} \end{Bmatrix} = \begin{bmatrix} -k_{ip} & \xi k_{ip} & \eta k_{ip} \\ \xi k_{ip} & -\xi^2 k_{ip} & -\xi\eta k_{ip} \\ \eta k_{ip} & -\xi\eta k_{ip} & -\eta^2 k_{ip} \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \\ u_k \end{Bmatrix} \tag{B.10}$$

This additional spring, called ball-vertex spring, effectively constrains each vertex within the polyhedral ball that encloses it, and therefore the face-vertex crossover can be avoided as shown in Fig. B.1 (b).

The global force-displacement equations are solved using one of iterative methods, such as point Gauss-Seidel (for detail see [87]).

## B.2 Algebraic Method

Meshes that are used for high Reynolds number flow simulations typically require thin and highly stretched cells in a boundary layer. In this region, the spring analogy method is too expensive to be used, and it can be failed due to the face-vertex crossover. Therefore, the algebraic method is used for the boundary layer region [88].

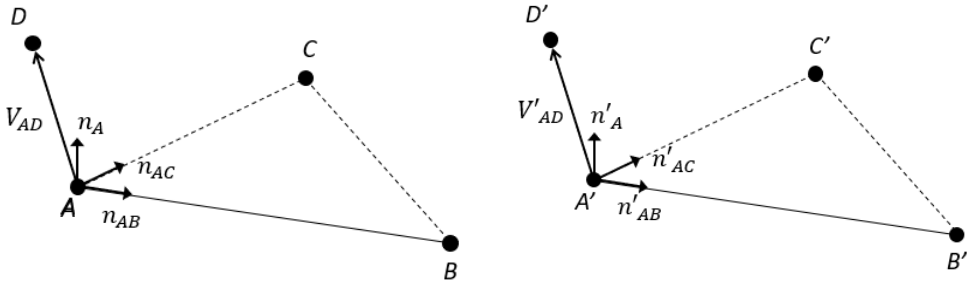
This algebraic method displaces the nodes along the strand grid that originate on the surface through the surface and normal vectors. In detail, the vectors $\mathbf{V}_{AD}$ from nodes on the surface $A$ to the nodes on upper layer $D$ along the strand are expressed in terms of the basis vectors (one normal vector $\mathbf{n}_A$ and two surface vectors $\mathbf{n}_{AB}$ and $\mathbf{n}_{AC}$) through the use of dot products (see Fig. B.2).

$$\mathbf{V}_{AD} = C_1\mathbf{n}_A + C_2\mathbf{n}_{AB} + C_3\mathbf{n}_{AC} \tag{B.11}$$

where $C_1 = \mathbf{V}_{AD} \cdot \mathbf{n}_A$, $C_2 = \mathbf{V}_{AD} \cdot \mathbf{n}_{AB}$, and $C_3 = \mathbf{V}_{AD} \cdot \mathbf{n}_{AC}$,

When the nodes $A, B, C$ are moved to their new positions $A', B', C'$, the new coordinate $D'$ is produced by applying the components of $\mathbf{V}_{AD}$ expressed in $\mathbf{n}_A$, $\mathbf{n}_{AB}$, and $\mathbf{n}_{AC}$ directions to the new basis vector. This procedure was applied along the strand grid to propagate the deformation of surface while strand grid remains normal to the surface.

$$D' = A' + C_1\mathbf{n}'_A + C_2\mathbf{n}'_{AB} + C_3\mathbf{n}'_{AC} \tag{B.12}$$




(a) Basis decomposition of a normal edge vector in original initial mesh

(b) New normal edge vector in deformed mesh

Figure B.2: Algebraic method for deformation of prismatic elements.

## B.3 Validation

The current grid deformation method is tested with a simulated fish locomotion, where the surface points experience large displacement (many orders greater than the cell size) [89]. Figure B.3 shows the example of deformed interior nodes from the initial hybrid mesh around the NACA0012 airfoil. The interior nodes are re-located using both the algebraic method for the structured grid and the spring analogy for the unstructured domain. It is observed that the mesh deformation technique is robust for both the unstructured and structured meshes, and there is no instance of any mesh crossover.

In addition to grid deformation technique, Geometric Conservation Law (GCL) is also validated by comparing the results of pitching airfoil simulation between from

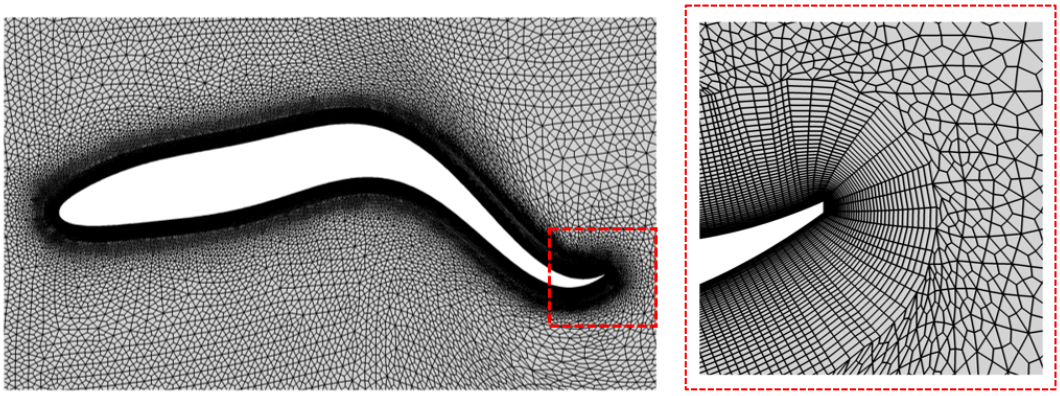


Figure B.3: Spring analogy and algebraic mesh deformation methods for the swimming fish-like body.

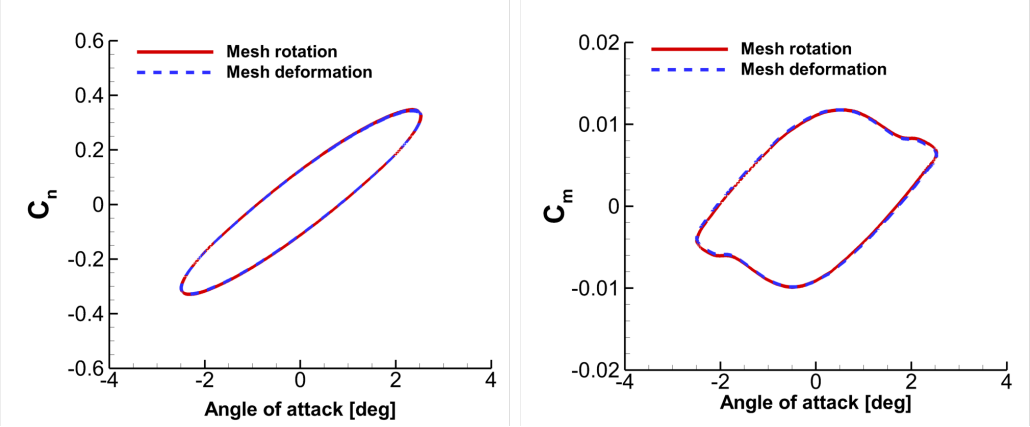
rigid grid rotation and grid deformation method. For the rigid grid rotation, the entire grid is rotated about the quarter-chord of the airfoil, whereas for the case of grid deformation, only the node points on the airfoil surface are rotated keeping the node points on the far-boundary stationary.

The variation of angle of attack is defined by $\alpha(t) = \alpha_0 + \alpha_m \sin(2M_\infty kt)$, where $\alpha_0$ (0.016°) is initial angle of attack, $\alpha_m$ (2.51°) is the pitching amplitude, $k$ (0.0814) is reduced frequency, and $t$ is non-dimensional time step. The freestream Mach number ($M_\infty$) is 0.755 and the Reynolds number is $5.5 \times 10^6$ based on the chord length. The grid for this calculation has a initial wall normal spacing of $1 \times 10^{-5}$ chord and 41 viscous mesh layers.

Figure B.4 compares the unsteady force hysteresis for the pitching airfoil between the results using either grid rotation and grid deformation. It is observed that the results obtained from the different methods are well matched with each other.

The current method is also applied to the forward flight rotor simulation with

prescribed deflection in the previous study [87]. In this previous study, only a few strand layers near the blade surface are treated as viscous mesh where the algebraic mesh deformation is applied and the other part of domains are treated using the spring analogy with the ball-vertex method.




(a) Lift coefficient (b) Moment coefficient

Figure B.4: Unsteady non-dimensional force hysteresis for a pitching NACA0012 airfoil.

# Bibliography

[1] Murayama, M., Yamamoto, K., and Kobayashi, K., "Validation of Computations Around High-Lift Configurations by Structured- and Unstructured-Mesh," *Journal of Aircraft*, Vol. 43, No. 2, 2006, pp. 395–406.

[2] Venkatakrishnan, V., "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, Vol. 118, No. 1, 1995, pp. 120–130.

[3] Koren, B., "A Robust Upwind Discretization Method for Advection, Diffusion and Source Terms," *Notes on Numerical Fluid Mechanics*, Vol. 45, 1993, pp. 117–138.

[4] Jiang, G. S., and Shu, C. W., "Efficient Implementation of Weighted ENO Schemes," *Journal of Computational Physics*, Vol. 126, No. 1, 1996, pp. 202–228.

[5] Sitaraman, J., Katz, A., Jayaraman, B., Wissink, A., and Sankaran, V., "Evaluation of a Multi-Solver Paradigm for CFD using Overset Unstructured and Structured Adaptive Cartesian Grids," 46th AIAA Aerospace Sciences Meeting, AIAA Paper 2008-660, Reno, NV, January 2008.

[6] Wissink, A., Potsdam, M., Sankaran, V., Sitaraman, J., and Mavriplis, D., "A Dual-Mesh Unstructured Adaptive Cartesian Computational Fluid Dynamics Approach for Hover Prediction," *Journal of the American Helicopter Society*, Vol. 61, No. 1, 2016, pp. 1–19.

[7] Kamkar, S., Wissink, A., Jameson, A., and Sankaran, V., "Feature-Driven Cartesian Adaptive Mesh Refinement in the Helios Code," 48th AIAA Aerospace Sciences Meeting, AIAA Paper 2010-171, Orlando, FL, January 2010.

[8] Hassan, O., Morgan, K., and Peraire, J., "An Implicit Finite-Element Method for High-Speed Flows," *International Journal for Numerical Methods in Engineering*, Vol. 32, 1991, pp. 183–205.

[9] Martin, D., and Lohner, R., "An Implicit Linelet-Based Solver for Incompressible Flows," 30th AIAA Aerospace Sciences Meeting, AIAA Paper 1993-0668, Reno, NV, January 1992.

[10] Mavriplis, D., "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes," *Journal of Computational Physics*, Vol. 145, No. 1, 1998, pp. 141–165.

[11] Cete, A., Yukselen, M., and Kaynak, U., "A Unifying Grid Approach for Solving Potential Flows Applicable to Structured and Unstructured Grid Configurations," *Computers and Fluids*, Vol. 37, 2008, pp. 35–50.

[12] Meakin, R. L., Wissink, A. M., Chan, W. M., Pandya, S. A., and Sitaraman, J., "On Strand Grids for Complex Flows," 18th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2007-3834, Miami, FL, June 2007.

[13] Lakshminarayan, V., Sitaraman, J., Roget, B., and Wissink, A., "Development and Validation of A Multi-Strand Solver for Complex Aerodynamic Flows," *Computers and Fluids*, Vol. 147, 2017, pp. 41–62.

[14] Sitaraman, J., and Roget, B., "Solution Algorithm for Unstructured Grids Using Quadrilateral Subdivision and Hamiltonian Paths," 52nd AIAA Aerospace Sciences Meeting, AIAA Paper 2014-0079, National Harbor, MD, January 2014.

[15] Muller, J., Roe, P., and Deconinck, H. "A Frontal Approach for Internal Node Generation for Delaunay Triangulations," *International Journal of Numerical Methods in Fluids*, Vol. 17, 1993, pp. 241–256.

[16] Govindarajan, B., Jung, Y., Baeder, J., and Sitaraman, J., "Efficient Three-Dimensional Solution for Unstructured Grids Using Hamiltonian Paths and Strand Grids," American Helicopter Society 71th Annual Forum, Virginia Beach, VA, May 2015.

[17] Bommes, D., Bruno, L., Pietroni, N., Puppo, E, Silva, C., Tarini, M., and Zorin, D., "Quad-Mesh Generation and Processing: A Survey," *Computer Graphics Forum*, Vol. 32, No. 6, 2013, pp. 51–76.

[18] Zihao, Z., "HAMSTRAN, an Indirect Method to Create All-quadrilateral Grids for the HAMSTR Flow Solver," *University of Maryland Master thesis*, 2017.

[19] Jung, Y., Govindarajan, B., and Baeder, J., "Turbulent and Unsteady Flows on Unstructured Line-Based Hamiltonian Paths and Strand Grids," *AIAA Journal*, Vol. 55, No. 6, 2017, pp. 1986–2001.

[20] Karypis, G., and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, 1999, pp. 359–392.

[21] Aftosmis, M., Gaitonde, D., and Tavares, T., "Behavior of Linear Reconstruction Techniques on Unstructured Meshes," *AIAA Journal*, Vol. 33, No. 11, 1995, pp. 2038–2049.

[22] Haselbacher, A. and Blazek J., "Accurate and Efficient Discretization of Navier–Stokes Equations on Mixed Grids," *AIAA Journal*, Vol. 38, 2000, pp. 2094–2102.

[23] Lee, E., Ahn, H., and Luo, H., "Cell-centered high-order hyperbolic finite volume method for diffusion equation on unstructured grids," *Journal of Computational Physics*, Vol. 355, 2018, pp. 464–491.

[24] Rieper, F., "A Low-Mach Number Fix for Roe's Approximate Riemann Solver," *Journal of Computational Physics*, Vol. 230, No. 13, 2011, pp. 5263–5287.

[25] Harten, A., Engquist, B., Osher, S., and Chakravarthy, S., "Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III," *Journal of Computational Physics*, Vol. 131, No. 1, 1997, pp. 3–47.

[26] Buelow, P. E. O., Venkateswaran, S., and Merkle, C. L., "Stability and Convergence Analysis of Implicit Upwind Schemes," *Computers and Fluids*, Vol. 30, No. 78, 2001, pp. 961–988.

[27] Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statiscal Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[28] Behr, M. and Tezduyar, T.E., "Finite Element Solution Strategies for Large-Scale Flow Simulations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 112, 1994, pp. 3–24.

[29] Anderson, W. k., Rausch, R. D., and Bonhaus, D. L., "Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids," *Journal of Computational Physics*, Vol. 128, No. 219, 1996, pp. 391–408.

[30] Blanco, M and Zingg, D. W., "Fast Newton-Krylov Method for Unstructured Grids," *AIAA Journal*, Vol. 36, No. 4, 1998, pp. 607–612.

[31] Spalart, P. and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," 30th AIAA Aerospace Sciences Meeting, AIAA Paper 1992-493, Reno, NV, January 1992.

[32] Menter, F. R., "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598–1605.

[33] Spalart, P., Deck, S., Shur, M., and Squires, K., "A New Version of Detached-eddy Simulation, Resistant to Ambiguous Grid Densities," *Theoretical and Computational Fluid Dynamics*, Vol. 20, 2006, pp. 181–195.

[34] Scotti, A., Meneveau, C., and Fatica, M., "Dynamic Smagorinsky model on anisotropic grids," *Physics of Fluids*, Vol. 9, 1997, pp. 1856–1858.

[35] Langtry, R. and Menter, F., "Correlation-Based Transition Modeling for Unstructured Parallelized Computational Fluid Dynamics Codes," *AIAA Journal*, Vol. 47, 2009, No. 12, pp. 2894–2906.

[36] Jung, Y. and Baeder, J., "$\gamma - \overline{Re_{\theta t}}-$SA with Crossflow Transition Model using Hamiltonian-Strand Approach," 56th AIAA Aerospace Sciences Meeting, AIAA Paper 2018-1040, Kissimmee, FL, January 2018.

[37] Jung, Y. and Baeder, J., "$\gamma-\overline{Re_{\theta t}}-$Spalart–Allmaras with Crossflow Transition Model using Hamiltonian-Strand Approach," *Journal of Aircraft*, Published Online, 2019.

[38] Medida, S., "Correlation-based Transition Modeling for External Aerodynamic Flows," *University of Maryland PhD Dissertation*, 2014.

[39] Brazell, M., Sitaraman, J., and Mavriplis, D., "An Overset Mesh Approach for 3D Mixed Element High-order Dicretizations," *Journal of Computational Physics*, Vol. 322, 2016, pp. 33–51.

[40] Jude, D., Lee, B., Jung, Y., Petermann, J., Govindarajan, B., and Baeder, J., "Application of a Heterogeneous CFD Framework Towards Simulating Complete Rotorcraft Configurations," American Helicopter Society 74th Annual Forum, Phoenix, AZ, May 2018.

[41] Jung, Y., Jude, D., Govindarajan, B., and Baeder, J., "Wind Turbine Simulations Using CPU/GPU Heterogeneous Computing," North America Wind Energy Academy Symposium, 2017.

[42] Salari, K. and Knupp, P., "Code Verification by the Method of Manufactured Solutions," *Sandia Report*, Sandia National Laboratories, 2000.

[43] Ghosh, D. and Baeder, J., "Compact Reconstruction Schemes with Weighted ENO Limiting for Hyperbolic Conservation Laws," *SIAM Journal on Scientific Computing*, Vol. 34, No. 3, 2012, pp. A1678–A1706.

[44] "Turbulence Modeling Resource," *NASA TR 2017,* http://turbmodels.larc.nasa.gov.

[45] Schaeffler, N. W., Allan, B. G., Lienard, C., and Le Pape, A., "Progress Towards Fuselage Drag Reduction via Active Flow Control: A Combined CFD and Experimental Effort," 36th European Rotorcraft Forum Paper 064, Paris, France, September 2010.

[46] Srinivasan, G. and Baeder, J., "TURNS: A Free-Wake Euler/Navier–Stokes Numerical Method for Helicopter Rotors," *AIAA Journal*, Vol. 31, 1993, pp. 959–962.

[47] Johnson, T. and Patel, V., "Flow Past a Sphere up to a Reynolds Number of 300," *Journal of Fluid Mechanics*, Vol. 378, 1999, pp. 19–70.

[48] Ladson, C., "Effect of Independent Variation of Mach and Reynolds Numbers on the Low-Speed Aerodynamic Characteristics of the NACA 0012 Airfoil Section," *NASA TM 4074*, 1988.

[49] McAlister, K., and Takahashi, R., "NACA 0015 Wing Pressure and Trailing Vortex Measurements," *Technical Report A-91056*, NASA Langley Research Center, 1991.

[50] Kim, I., Elghobashi, S., and Sirignano, W., "Three-Dimensional Flow over Two Spheres Placed Side by Side," *Journal of Fluid Mechanics*, Vol. 246, 1993, pp. 465–488.

[51] Rogers, S. E., "Comparison of Implicit Schemes for the Incompressible Navier–Stokes Equations," *AIAA Journal*, Vol. 33, No. 11, 1995, pp. 2066–2072.

[52] Bas, O., Cete, A., Mengi, S., Tuncer, I., and Kaynak, U., "A Novel Alternating Cell Directions Implicit Method for the Solution of Incompressible Navier Stokes Equations on Unstructured Grids," *Journal of Applied Fluid Mechanics*, Vol. 10, No. 6, 2017, pp. 1561–1570.

[53] Caradonna, F. X., and Tung, C., "Experimental and Analytical Studies of a Model Helicopter in Hover," *Technical Report TM 81232*, NASA Langley Research Center, September, 1981.

[54] Jude, D. and Baeder, J., "Extending a Three-Dimensional GPU RANS Solver for Unsteady Grid Motion and Free-Wake Coupling," 54th AIAA Aerospace Sciences Meeting, AIAA Paper 2016-1811, San Diego, CA, January 2016.

[55] Jung, Y. Govindarajan, B. and Baeder, J., "On the Accuracy and Convergence of a Hamiltonian-Strand Approach for Aerodynamic Flows," 55th AIAA Aerospace Sciences Meeting, AIAA Paper 2017-1194, Grapevine, TX, January 2017.

[56] Chin, V., Peters, D., Spaid, F., and McGhee, R., "Flowfield Measurements About a Multi-Element Airfoil at High Reynolds Numbers," AIAA 24th Fluid Dynamics Conference, AIAA Paper 1993-3137, Orlando, FL, July 1993.

[57] Kim, S., Alonso, J., and Jameson, A., "Multi-Element High-Lift Configuration Design Optimization Using Viscous Continuous Adjoint Method," *Journal of Aircraft*, Vol. 41, No. 5, 2004, pp. 1082–1097.

[58] Cai, J., Tsai, H., and Liu, F., "An Overset Grid Solver for Viscous Computations with Multigrid and Parallel Computing," 16th AIAA Computational Fluid Dynamic Conference, AIAA Paper 2003-4232, Orlando, FL, June 2003.

[59] Costenoble, A., Jung, Y., Govindarajan, B., and Baeder, J., "Automated Mesh Generation and Solution Analysis of Arbitrary Airfoil Geometries," AHS Technical Conference on Aeromechanics Design for Transformative Vertical Flight, San Francisco, CA, January 2018.

[60] Rogers, S., Menter, F., Durbin, P., and Mansour, N., "A Comparison of Turbulence Models In Computing Multi-Element Airfoil Flows," 32nd AIAA Aerospace Sciences Meeting, AIAA Paper 1994-291, Reno, NV, January 1994.

[61] Overmeyer, A. and Martin, P., "Measured Boundary Layer Transition and Rotor Hover Performance at Model Scale," 55th AIAA Aerospace Sciences Meeting, AIAA Paper 2017-1872, Grapevine, TX, January 2017.

[62] Lee, B., Jung, Y., Jude, D., and Baeder, J., "Turbulent Transition Prediction of PSP Hovering Rotor using $\gamma - \overline{Re_{\theta t}}$−SA with Crossflow Transition Model," 57th AIAA Aerospace Sciences Meeting, AIAA Paper 2019-0286, San Diego, CA, January 2019.

[63] Jain, R., "CFD Performance and Turbulence Transition Predictions on an Installed Model-scale Rotor in Hover," 55th AIAA Aeroepace Sciences Meeting, AIAA Paper 2017-1871, Grapevine, TX, January 2017.

[64] Parwani, A. and Coder, J., "CFD predictions of Rotor-Fuselage Interactins Using Laminar-Turbulent Transition Modeling," American Helicopter Society 74th Annual Forum, Phoenix, AZ, May 2018.

[65] Sezer-Uzol, N. and Uzol, O., "Effect of Steady and Transient Wind Shear on the Wake Structure and Performance of a Horizontal Axis Wind Turbine Rotor," *Wind Energy*, Vol. 16, 2013, pp. 1–17.

[66] Sorensen, N., Michelsen, J., and Schreck, S., "Navier–Stokes Predictions of the NREL Phase VI Rotor in the NASA Ames 80ft x 120ft Wind Tunnel," *Wind Energy*, Vol. 5, No. 2-3, 2002, pp. 151–169.

[67] Sorensen, N., "CFD Modeling of Laminar-turbulent Transition for Airfoils and Rotors Using the $\gamma - \widetilde{Re_\theta}$ Model," *Wind Energy*, Vol. 12, No. 8, 2002, pp. 715–733.

[68] Hand, M., Simms, D., Fingersh, L., Jager, D., Cotrell, J., Schreck, S., and Larwood, S., "Unsteady Aerodynamics Experiment Phase VI: Wind Tunnel Test Configurations and Available Data Campaigns," *NREL/TP-500-29955*, 2001.

[69] Zahle, F., Sorensen, N., and Johansen, J., "Wind Turbine Rotor-Tower Interaction Using an Incompressible Overset Grid Method," *Wind Energy*, Vol. 12, No. 6, 2009, pp. 594–619.

[70] Wind turbines, part 1:Design Requirements, *International Standard IEC 61400-1*, 2005.

[71] McGowan and Rogers, *Wind Energy Explained: Theory, Design and Application* 2nd edition. Wiley, 2010.

[72] Potsdam, M., Cross, P., and Hill, M., "Assessment of CREATE-AV Helios for Complex Rotating Hub Wakes," American Helicopter Society 73rd Annual Forum, Fort Worth, TX, May 2017.

[73] Schmitz, S., Reich, D., Smith, M., and Centolanze, L., "First Rotor Hub Flow Prediction Workshop Experimental Data Campaigns and Computational Analyses," American Helicopter Society 73rd Annual Forum, Fort Worth, TX, May 2017.

[74] Schmitz, S., Tierney, C., Metkowski, L., Reich, D., Jaffa, N., Centolanze, L., and Thomas, M., "2nd Rotor Hub Flow Prediction Workshop Experimental Data Campaigns and Computational Analyses," Vertical Flight Society 75th Annual Forum, Philadelphia, PA, May 2019.

[75] Metkowski, L., Reich, D., Sinding, K., Jaffa, N., and Schmitz, S., "Full-Scale Reynolds Number Experiment on Interactional Aerodynamics Between Two Model Rotor Hubs and a Horizontal Stabilizer," American Helicopter Society 74th Annual Forum, Phoenix, AZ, May 2018.

[76] Lee, B., Jung, Y., Jude, D., and Baeder, J., "Prediction of Rotor Hub Flow Using Mercury Framework," Vertical Flight Society 75th Annual Forum, Philadelphia, PA, May 2019.

[77] Potsdam, M. and Sitaraman, J., "Assessment of HPCMP CREATE-AV Helios for Interactional Aerodynamics of Hub Wakes Impinging on a Horizontal Stabilizer," Vertical Flight Society 75th Annual Forum, Philadelphia, PA, May 2019.

[78] Wang, X., Jung, Y., Baeder, J., and Chopra, I., "CFD Pressure/Airload Correlation with Experimental Data on a Slowed Mach-Scaled Rotor at High Advance Ratios," American Helicopter Society 74th Annual Forum, Phoenix, AZ, May 2018.

[79] Datta, A., Sitaraman, J., Chopra, I., and Baeder, J., "CFD/CSD Prediction of Rotor Vibratory Loads in High-Speed Flight," *Journal of Aircraft*, Vol. 43, No. 6, 2006, pp. 1698–1709.

[80] Sridharam, A., Rubenstein, G., Moy, D., and Chopra, I., "A Python-Based Framework for Computationally Efficient Trim and Real-Time Simulation Using Comprehensive Analysis," *Journal of Aircraft*, Vol. 63, No. 1, 2018, pp. 1–15.

[81] Muller, C. and Herbst, F., "Modelling of Crossflow-induced Transition based on Local Variables," 11th World Congress Computational Mechanics and 5th European Conference on Computational Mechanics and 6th European Conference on Computational Fluid Dynamics, Barcelona, Spain, July, 2014.

[82] Radeztsky, R., Reibert, M., and Saric, W., "Effect of Micron-Sized Roughness on Transition in Swept-Wing Flows," 31st AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 1993-0076, Reno, NV, January 1993.

[83] Dassler, P., Kozulovic, D., and Fiala, A., "An Approach for Modelling the Roughness-Induced Boundary Layer Transition Using Transport Equations,"

European Conference on Computational Methods in Applied Sciences and Engineering, Vienna, Austria, September 2012.

[84] Langel, C., Chow, R., Dam, C., Maniaci, D., Ehrmann, R., and White, E., "A Computational Approach to Simulating the Effects of Realistic Surface Roughness on Boundary Layer Transition," 52nd AIAA Aerospace Sciences Meeting, AIAA Paper 2014-0234, National Harbor, MD, January 2014.

[85] Jung, Y. and Baeder, J., "Uncertainty Quantification for Laminar-Turbulent Transition on Airfoil and Fuselage," Vertical Flight Society 75th Annual Forum, Philadelphia, PA, May 2019.

[86] Acikgoz, N. and Bottasso, C., "A unified approach to the deformation of simplicial and non-simplicial meshes in two and three dimensions with guaranteed validity," *Computers and Structures*, Vol. 85, 2007, pp. 944–954.

[87] Jung, Y., Jude, D., Govindarajan, B., and Baeder, J., "Line-Based Unstructured/Structured Heterogenous CPU/GPU Framework for Complex Aerodynamic Flows," American Helicopter Society 73rd Annual Forum, Fort Worth, TX, May 2017.

[88] Kholodar, D., Morton, S., and Cummings, R., "Deformation of Unstructured Viscous Grids," 43rd AIAA Aerospace Sciences Meeting, AIAA Paper 2005-926, Reno, NV, January 2005.

[89] Bergmann, M. and Iollo, A., "Modeling and simulation of fish-like swimming," *Journal of Computational Physics*, Vol. 230, 2011, pp. 329–348.