A Higher-Order Unstructured Finite Volume Solver for Three-Dimensional Compressible Flows

Shayan Hoshyari

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Applied Science
in
Mechanical Engineering
at
The University of British Columbia

August 2017

Abstract

High-order accurate numerical discretization methods are attractive for their potential to significantly reduce the computational costs compared to the traditional second-order methods. Among the various unstructured higher-order discretization schemes, the k-exact reconstruction finite volume method is of interest for its straightforward mathematical formulation, and its compatibility with the current lower-order industrial solvers. However, current three-dimensional finite volume solvers are limited to the solution of inviscid and laminar viscous flow problems. Since three-dimensional turbulent flows appear in many industrial applications, the current thesis takes the first step towards the development of a three-dimensional higher-order finite volume solver for the solution of both inviscid and viscous turbulent steady-state flow problems.

The k-exact finite volume formulation of the governing equations is rederived in a dimension-independent manner, where the negative Spalart-Allmaras turbulence model is employed. This one-equation model is reasonably accurate for many flow conditions, and its simplicity makes it a good starting point for the development of numerical algorithms. Then, the three-dimensional mesh preprocessing steps for a finite volume simulation are presented, including higher-order accurate numerical quadrature, and capturing the boundary curvature in highly anisotropic meshes. Also, the issues of k-exact reconstruction in handling highly anisotropic meshes are reviewed and addressed.

Since three-dimensional problems can require much more memory than their two-dimensional counterparts, solution methods that work in two dimensions might not be feasible in three dimensions anymore. As an attempt to overcome this issue, a practical and parallel scalable method for the solution of the discretized system of nonlinear equations is presented.

Finally, the solution of four three-dimensional test problems are studied: Poisson's equation in a cubic domain, inviscid flow over a sphere, turbulent flow over a flat plate, and turbulent flow over an extruded NACA 0012 airfoil. The solution is verified, and the resource consumption of the flow solver is measured. The results demonstrate the benefit and practicality of using higher-order methods for obtaining a certain level of accuracy.

Lay Summary

The finite volume method is a popular numerical scheme for the solution of aerodynamic flow problems. Although conventional finite volume schemes are mostly second-order accurate, there has been a growing interest in higher-order accurate numerical methods, since they can be considerably more efficient in terms of computational resources for achieving a certain level of accuracy.

Higher-order finite volume methods have been successfully developed for a wide range of two-dimensional flow problems. Nevertheless, numerical methods that work in two dimensions might not be feasible in three dimensions anymore, since three-dimensional problems can require much more memory than their two-dimensional counterparts. The current thesis aims at identifying, and resolving such shortcomings to construct a working three-dimensional finite volume solver with an emphasis on turbulent flows. Subsequently, three-dimensional benchmark flow problems are solved to verify and assess the performance of the developed numerical method.

Preface

All the work presented in this thesis is an intellectual product of a working relationship between Shayan Hoshyari and Dr. Carl Ollivier-Gooch. The implementation of the methods, the data analysis, and the manuscript preparations were done by Shayan Hoshyari with invaluable guidance from Carl Ollivier-Gooch throughout the process.

Contents

List of Tables

List of Figures

List of Symbols

Orders

k	Order of reconstruction
l	Order of Lagrangian polynomials
p	Order of finite element basis functions
q	Order of accuracy for a quadrature rule

Number of Entities

$N_{\mathbf{unk}}$	Number of unknowns
$N_{\mathbf{DOF}}$	Number of degrees of freedom
$N_{\mathbf{dim}}$	Number of dimensions
$N_{\mathbf{rec}}$	Number of reconstruction basis functions
$N_{\mathbf{lag}}$	Number of Lagrangian basis functions
$N_{\mathbf{qua}}$	Number of quadrature points
$N_{\mathbf{CV}}$	Number of control volumes

Numerical Discretization

$\mathbf{x}$	Cartesian coordinates
$\mathcal{L}$	Differential operator
$\mathbb{R}$	Set of rational numbers
$\mathbf{u}$	Solution vector
h	Mesh representative length scale
$\mathcal{T}_h$	Set of all control volumes
$\mathbf{u}_h$	Discrete solution vector

$\mathbf{U}_h$	Vector of degrees of freedom
$\mathbf{e}_h$	Discretization error
$\mathbf{R}$	Residual vector
Ω	Domain
$\partial\Omega$	Boundary of domain
$\mathbf{n}$	Surface unit normal
τ	A control volume
$\partial\tau$	Boundary of a control volume
Ω_τ	Volume or area of a control volume
ψ	A Finite element or Lagrangian basis function
ϕ	A Cartesian coordinates monomial
φ^*	A Principal coordinates monomial
φ^+	A Curvilinear coordinates monomial
$\mathbf{x}_\tau$	Reference location of a control volume
$\mathbf{w}_\tau$	Principal coordinates for a control volume
$\boldsymbol{\xi}_\tau$	Curvilinear coordinates for a control volume
t_τ	Tangent to wall coordinate for a control volume
d_τ	Normal to wall coordinate for a control volume
$\boldsymbol{F}$	Inviscid flux matrix
$\boldsymbol{Q}$	Viscous flux matrix
$\mathbf{S}$	Source term vector
$\mathcal{F}$	Numerical inviscid flux function
$\mathcal{Q}$	Numerical viscous flux function
B	Boundary condition constraint function
$\mathcal{B}$	Boundary conditions
MinNeigh	Minimum number of reconstruction stencil members
BdryQuad	Set of all boundary quadrature points for a control volume
Stencil	Reconstruction stencil for a control volume

Solution of System of Equations

p	ILU fill level
$\mathbf{x}$	Vector of unknowns

$\mathbf{b}$	Right hand side vector for a linear system
$\mathbf{r}$	Residual vector for a linear system
$\boldsymbol{A}$	Left hand side matrix for a linear system
$\boldsymbol{A}^*$	The matrix used for constructing the preconditioner
$\boldsymbol{P}$	Preconditioner matrix
$\tilde{\boldsymbol{L}}, \tilde{\boldsymbol{U}}$	Lower and upper matrices resulting from ILU factorization
CFL	The Courant-Friedrichs-Lewy number

Flow Related Variables

Re	Reynolds number
Pr	Prandtl number
Pr_T	Turbulent Prandtl number
Ma	Mach number
$Diff$	Diffusion term in the Spalart-Allmaras turbulence model
$Prod$	Production term in the Spalart-Allmaras turbulence model
$Dest$	Destruction term in the Spalart-Allmaras turbulence model
$Trip$	Trip term in the Spalart-Allmaras turbulence model
$\boldsymbol{\tau}$	Viscous stress tensor
$\mathbf{v}$	Velocity vector
ρ	Density
E	Total energy
P	Pressure
T	Temperature
μ	Viscosity
μ_T	Turbulent viscosity
γ	Specific heat ratio
$\tilde{\nu}$	Spalart-Allmaras working variable
t	Time
d	Distance from wall

Overscripts

$\tilde{(.)}$	Roe average state
$\hat{(.)}$	Reference element state

Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. Carl Ollivier-Gooch, for his professional mentorship, continuous support, and endless patience. It has been a pleasure for me to work under his supervision for the past two years.

I am thankful for the financial support of the Natural Sciences and Engineering Research Council of Canada and the University of British Columbia through the Canada Graduate Scholarships-Master's (CGS-M) program and the Gartshore Scholarship, respectively.

I would like to express my gratitude to my friends and fellow labmates for their valuable help and suggestions. Special thanks to Alireza for patiently helping me with my initial transition into the research environment at ANSLab, and Gary for proofreading this thesis.

I wish to thank my dear friend Mohammad. I will never forget your help and advice both inside and outside of work. Also, thank you for your technical suggestions and for proofreading a major portion of this thesis.

Last but not least, I would like to thank my parents, Iraj and Mahboubeh, and my sister, Leiana, for their unconditional love, support, and caring.

Chapter 1

Introduction

1.1 Motivation

The advent of computational fluid dynamics (CFD) has considerably improved the design and manufacturing processes in many industries such as aerospace, turbomachinery, oil and gas, and bioengineering. CFD finds solutions to engineering problems by numerically solving the governing and/or modeling partial differential equations (PDEs). CFD simulations offer a cheap alternative to experimental setups in many, though not all, situations while not suffering from the limitation of analytical methods in handling complex geometries. Nevertheless, CFD methods are not always considered a rival to the other methods of analysis. Computational methods are sometimes used for tuning parameters or selecting methods of measurement in experimental setups. For example, Yoo et al. [80] employ CFD simulations to estimate the right scale for building a nuclear fuel cask experimental model. Conversely, many PDEs that are solved in CFD are derived from experimental data, such as the turbulence model of Spalart and Allmaras [68].

A majority of CFD methods are categorized as mesh-based discretization schemes. A mesh-based discretization scheme seeks to approximate the solution to a PDE $\boldsymbol{\mathcal{L}}\mathbf{u}(\mathbf{x}) = 0$ inside a domain $\Omega \subset \mathbb{R}^{N_{\text{dim}}}$, where $\boldsymbol{\mathcal{L}}$ is a differential operator, $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^{N_{\text{unk}}}$ is the unknown exact solution, and no time dependence has been assumed for simplicity. In the first step, the domain is subdivided (meshed) into a set of non-overlapping sub-volumes $\mathcal{T}_h$, followed by defining a discrete function in the form of $\mathbf{u}_h(\mathbf{x}; \mathbf{U}_h)$. Here, the term discrete means that this function will be uniquely defined if a finite number of degrees of freedom, $\mathbf{U}_h \in \mathbb{R}^{N_{\text{DOF}}}$, are all specified. Furthermore, the subscript h, which is the representative length scale of the subdivision, emphasizes the dependency of the solution on the mesh. Finally, the goal is to find $\mathbf{U}_h$, such that $\mathbf{u}_h$ closely approximates the exact solution $\mathbf{u}$ as the mesh gets smaller. The rigorous definition of "to closely approximate" is that the discretization error, $\mathbf{e}_h = \mathbf{u}_h - \mathbf{u}$, must have an asymptotic behavior such that: $\|\mathbf{e}_h\| = O(h^p)$. The discretization scheme is then said to be pth-order accurate.

Although conventional discretization schemes for aerodynamical flows are only second-order accurate, there has been a growing interest in higher-order accurate methods. Higher-order methods can be considerably more efficient compared to second-order methods in achieving accurate solutions, since the former results in more accurate results on coarser meshes [21, 48]. Even small improvements in accuracy can be of considerable importance. For example, Vassberg et al. [74] show that for a long-range jet-aircraft that delivers a payload between distant city pairs, a 1% error in predicting the drag can reduce the carried payload by 7%. With the airlines operating on profit margins of only a few percent, such a loss can make the service completely unprofitable.

Regarding the adoption of higher-order methods, there has always been the legitimate concern of modeling errors, i.e., the discrepancies between the exact solution of the governing equations and the true physical quantities. For a problem where the modeling errors are dominant, only a limited level of reduction in the discretization errors is of interest. Even in such a case, preliminary results have shown that an hp-adaptive strategy can achieve a desired level of accuracy faster than conventional second-order methods [33, 38]. An hp-adaptive method increases the local order of accuracy p and decreases the mesh size h at only certain regions of the domain. In the context of design and optimization, thousands of simulations might be required for optimizing a subject geometry [42]. Thus, even a small runtime improvement for a single simulation can result in a considerable reduction of the overall resource consumption. Moreover, modeling errors are not always the dominant mode. In recent work by Mavriplis [45], where he studied the numerical solution of the flow around a wing-body configuration from the AIAA drag prediction workshop (DPW), the dominant errors were found to be those of discretization.

For structured meshes, highly-accurate finite difference methods have long been developed [39, 75], and are known to have superior properties in terms of computational cost and efficiency [19]. Nonetheless, generation of structured meshes for complex geometries is a challenging task, and requires extensive amounts of human input. Therefore, grids for complex geometries are often obtained by the fairly automatic and convenient alternative of unstructured mesh generation techniques.

For unstructured meshes, various approaches have been devised for achieving higher-order accuracy, most notably: continuous [5] and discontinuous [29] Galerkin finite element methods, the correction procedure via reconstruction formulations of the discontinuous Galerkin and spectral volume methods [31], and finite volume schemes [32]. (See the work of Andren et al. [8] for a detailed comparison of numerical results). The use of finite volume methods is partly motivated by their straightforward mathematical formulation compared to the complex structure of the other mentioned methods. Furthermore, most of the current industrial CFD codes are based on the finite volume method. While implementing other approaches would require the development of completely new commercial codes, higher-order finite volume methods can be integrated into the current industrial solvers. Finite volume methods are also attractive because of the smaller number of degrees of freedom that they require for the same mesh compared to finite element methods.

In two dimensions, unstructured high-order finite volume methods have been successfully applied

to a range of aerodynamic problems: the Euler equations [27, 46], laminar Navier-Stokes equations [33, 40], and turbulent Reynolds Averaged Navier-Stokes (RANS) equations [32]. Although taking the effects of turbulence into consideration is necessary for correctly capturing many aerodynamic flows, three-dimensional results are scarce and limited to the solution of Euler and laminar Navier-Stokes equations [25, 40]. In the long run, the ANSLab team at UBC is interested in developing a three-dimensional higher-order finite volume solver for the solution of both inviscid and viscous turbulent steady-state flow problems, while providing approximate error bounds on target unknowns such as lift and drag. The first step towards this goal will be taken in this thesis: solution of well-known three-dimensional benchmark flow problems.

1.2 Objectives

The ultimate goal of this thesis is the generalization of our current 2-D flow solver, ANSLib, for the solution of 3-D benchmark problems involving inviscid and viscous turbulent flows. The pursuit of this goal is divided into the following steps:

- Derive the finite volume formulation of the governing equations in a dimension-independent manner. Of course, the choice of the turbulence model in this part considerably affects the solution. In this thesis, the objective is to employ the negative Spalart-Allmaras turbulence model [6]. This one-equation model is reasonably accurate for many flow conditions, and its simplicity makes it a good starting point for the development of numerical algorithms. The negative version of this model is chosen because it permits negative values for the model working variable, which can be present when higher-order methods are employed.
- Identify and undertake the preprocessing steps involved in handling three-dimensional grids required for turbulent simulations.
- Design a practical and parallel scalable method for the solution of the discretized system of nonlinear equations. Since three-dimensional problems can require much more memory than their two-dimensional counterparts, solution methods that work in two dimensions might not be feasible in three dimensions anymore.
- Verify the performance of the solver, and the accuracy of the solutions obtained.

1.3 Thesis Outline

This thesis is organized in the following manner:

Chapter 2 provides an overview of our in-house flow solver ANSLib, in which the algorithms of this thesis have been implemented. The key concepts of the solver, i.e., k-exact reconstruction and finite volume discretization are discussed in detail. Also, governing and modeling equations of interest

are introduced. Finally, the flux functions and the boundary conditions are revisited in a dimension-independent notation.

Chapter 3 discusses the preprocessing steps of three-dimensional meshes for a finite volume simulation. First, numerical integration is addressed, where Gauss quadrature rules are employed in conjunction with reference element mappings to construct quadrature information for the control volumes and their faces. Then, the capturing of boundary curvature in highly anisotropic meshes is addressed, which is a necessity for higher-order solution methods. Finally, the issues of k-exact reconstruction in handling highly anisotropic meshes are reviewed and addressed.

In Chapter 4, the solution of the discretized system of nonlinear equations is discussed. First, the pseudo transient continuation method is revisited, which transforms the solution of the nonlinear system of equations into the solution of a series of linear systems. Then, a memory lean, yet effective, method for the solution of the corresponding linear systems is proposed, and compared to the previously available linear solution methods in ANSLib.

Chapter 5 presents the solution of four three-dimensional test problems. To test the correct implementation of the mesh preprocessing algorithms, Poisson's equation is solved in a simple geometry, where the discretization error is explicitly evaluated and its asymptotic behavior is verified. Then, the subsonic inviscid flow around a sphere is studied, where the solution accuracy is verified by measuring the deviation of the entropy from the inflow conditions throughout the domain. Finally, the problems of viscous turbulent flow over a flat plate and an extruded NACA 0012 airfoil are solved, and the solutions are verified against the reference data provided in the NASA turbulence modeling (TMR) website [60].

In the end, Chapter 6 summarizes the research of the thesis, provides conclusions, and proposes possible future work.

In addition, the command-line options that were provided to the ANSLib executable for running each test case are provided in Appendix A. A sample script for running a parallel job on the WestGrid Grex cluster [3] is also given in Appendix B, as the three-dimensional cases of this thesis were all run on this cluster.

Chapter 2

Background

This chapter presents the fundamentals upon which this research has been founded. As the algorithms involved in this thesis have been implemented as parts of ANSLib, it is essential to present the working mechanism of this solver. Thus, this chapter starts with a description of the finite volume method and k-exact reconstruction, which are the numerical solution schemes in ANSLib, and then moves on to the model equations of interest.

2.1 The Finite Volume Method

The finite volume method is applied to equations which can be expressed in the conservative form:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\boldsymbol{F}(\mathbf{u}) - \boldsymbol{Q}(\mathbf{u}, \nabla \mathbf{u})) = \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}), \tag{2.1}$$

where $\boldsymbol{F} \in \mathbb{R}^{N_{\text{unk}} \times N_{\text{dim}}}$ is the inviscid flux matrix, $\boldsymbol{Q} \in \mathbb{R}^{N_{\text{unk}} \times N_{\text{dim}}}$ is the viscous flux matrix, and $\mathbf{S} \in \mathbb{R}^{N_{\text{unk}}}$ is the source term vector. In this method, the degrees of freedom are the same as the average value of the discrete solution inside every control volume. This fundamental constraint is known as the conservation of the mean, and can be written as:

$$\frac{1}{\Omega_\tau} \int_\tau \mathbf{u}_h(\mathbf{x}) d\Omega = \mathbf{U}_{h,\tau} \qquad \tau \in \mathcal{T}_h, \tag{2.2}$$

where Ω_τ and $\mathbf{U}_{h,\tau} \in \mathbb{R}^{N_{\text{unk}}}$ represent the volume and local DOF vector for control volume τ, respectively. Establishing a relation between the discrete solution $\mathbf{u}_h$, and the degrees of freedom $\mathbf{U}_h$, when satisfying the conservation of the mean constraint, is called reconstruction in the finite volume framework. To accomplish this, ANSLib uses the k-exact reconstruction scheme, which will be introduced in Section 2.2.

The finite volume method uses the divergence theorem to discretize Equation (2.1). Consider a control volume $\tau \in \mathcal{T}_h$ as depicted in Figure 2.1. Integrating Equation (2.1) inside the control volume, and using

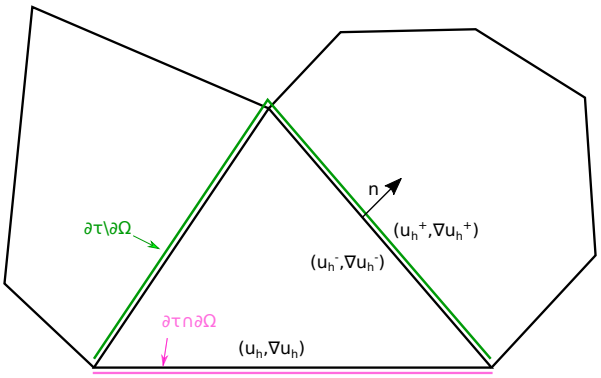


Figure 2.1: Illustration of terminology in finite volume discretization

the divergence theorem gives:

$$\frac{d\mathbf{U}_{h,\tau}}{dt} + \frac{1}{\Omega_\tau}\int_{\partial\tau\setminus\partial\Omega}\left(\boldsymbol{\mathcal{F}}_I(\mathbf{u}_h^+,\mathbf{u}_h^-) - \boldsymbol{\mathcal{Q}}_I(\mathbf{u}_h^+,\nabla\mathbf{u}_h^+,\mathbf{u}_h^-,\nabla\mathbf{u}_h^-)\right)dS + \frac{1}{\Omega_\tau}\int_{\partial\tau\cap\partial\Omega}\left(\boldsymbol{\mathcal{F}}_B(\mathbf{u}_h,\mathcal{B}) - \boldsymbol{\mathcal{Q}}_B(\mathbf{u}_h,\nabla\mathbf{u}_h,\mathcal{B})\right)dS - \frac{1}{\Omega_\tau}\int_\tau \mathbf{S}(\mathbf{u}_h,\nabla\mathbf{u}_h)d\Omega = 0, \quad (2.3)$$

where $\mathcal{B}$ represents the boundary conditions, in the case where the control volume has faces lying on the boundary. Although the discrete solution $\mathbf{u}_h$ and its gradient $\nabla\mathbf{u}_h$ are continuous inside every control volume, they can be discontinuous on the control volume boundaries $\partial\tau$. These discontinuous values are shown using the $(.)^+$ and $(.)^-$ notations. $\boldsymbol{\mathcal{F}}_I$ and $\boldsymbol{\mathcal{Q}}_I$ represent the interior numerical flux functions, while $\boldsymbol{\mathcal{F}}_B$ and $\boldsymbol{\mathcal{Q}}_B$ represent their boundary counterparts. Numerical flux functions are designed to mimic the product of the original flux matrices and the normal vector while taking into account the discontinuity of the discrete solution and the boundary conditions. The flux functions used in this thesis will be introduced in Section 2.4.

Looking back at Equation (2.3), the following system of ODEs for control volume averages can be derived:

$$\frac{d\mathbf{U}_h}{dt} + \mathbf{R}(\mathbf{U}_h) = 0, \quad (2.4)$$

which can be solved with a variety of time advance schemes when a time-accurate solution is of interest. Butcher [15] presents a detailed study of these methods. Although only the steady state solution is of interest in this thesis, the unsteady terms can be used to improve the robustness of the solver with respect to the initial solution guess. This method, known as pseudo transient continuation (PTC) [35], will be introduced in Chapter 4.

2.2 K-exact Reconstruction

The design of reconstruction schemes in the finite volume framework started with Van Leer's MUSCL scheme [72, 73]. His scheme took advantage of the uniform pattern of control volumes in structured

meshes and was not directly applicable to unstructured grids. Later on, the k-exact reconstruction [12] and the WENO/ENO [66] family of methods were designed, and did not suffer from MUSCL's shortcoming in handling unstructured meshes. The latter, however, has poor steady state convergence properties [59]. Targeted to solve aerodynamic steady state problems, ANSLib uses the k-exact reconstruction method in its finite volume formulation.

Suppose a polynomial function of order smaller or equal to k, $v(\mathbf{x})$, is integrated in every control volume to find the control volume averages $\mathbf{V}_h$. If these average values are fed to a k-exact reconstruction scheme to construct the function $v_h(\mathbf{x})$, the identity $v_h(\mathbf{x}) = v(\mathbf{x})$ must hold, which is where the name k-exact comes from. A k-exact scheme is also called $(k+1)$th-order accurate, since it results in a nominal discretization error of order $O(h^{(k+1)})$ [13]. For simplicity, let us consider this method when there is only one unknown variable, i.e., the vector $\mathbf{u}$ reduces to a scalar u. Generalization to multiple unknown variables will then be straightforward. In this case, the solution in every control volume is defined as the superposition of a set of basis functions in the form:

$$u_h(\mathbf{x}; \mathbf{U}_h, \mathcal{B})|_{x\in\tau} = u_{h,\tau}(\mathbf{x}; \mathbf{U}_h, \mathcal{B}) = \sum_{i=1}^{N_{\text{rec}}} a_\tau^i(\mathbf{U}_h, \mathcal{B})\phi_\tau^i(\mathbf{x}) \quad \tau \in \mathcal{T}_h. \tag{2.5}$$

Where $\phi_\tau^i(\mathbf{x})$ and a_τ^i represent the ith basis function and reconstruction coefficient for control volume τ, respectively. Most commonly, the basis functions will be chosen as monomials in Cartesian coordinates with origin at the reference point of each control volume:

$$\left\{ \phi_\tau^i(\mathbf{x}) \,\middle|\, i = 1 \dots N_{\text{rec}} \right\} = \left\{ \frac{1}{a!b!c!}(x_1 - x_{\tau 1})^a (x_2 - x_{\tau 2})^b (x_3 - x_{\tau 3})^c \,\middle|\, a+b+c \le k \right\}, \tag{2.6}$$

where $x_{\tau 1}$, $x_{\tau 2}$, and $x_{\tau 3}$ are the coordinates of the control volume's reference location, which is usually chosen as the centroid of volume. This particular choice of basis functions has the advantage that the discrete solution u_h will resemble the Taylor expansion of the exact solution u, which is particularly useful in theoretical analysis [54]. However, there are situations, e.g., highly anisotropic meshes, in which it would be more beneficial to use other basis functions [32]. Such a case will be introduced in Chapter 3.

The discrete solution must satisfy the conservation of the mean constraint, given in Equation (2.2). Moreover, for every control volume τ, a specific set of its neighbors are chosen as its reconstruction stencil $Stencil(\tau)$. The k-exact reconstruction requires $u_{h,\tau}$ to predict the average values of the members of $Stencil(\tau)$ closely. Furthermore, if the control volume is located on the boundary ($\partial\tau \cap \partial\Omega \neq \emptyset$), enforcing $u_{h,\tau}$ or $\nabla u_{h,\tau} \cdot \mathbf{n}$ to have certain values at boundary quadrature points, can improve convergence in the presence of certain boundary conditions [54]. Thus, the reconstruction coefficients can

be found by solving the following constrained minimization problem:

$$\begin{aligned} \underset{a_\tau^1 \ldots a_\tau^{N_{\text{rec}}}}{\text{minimize}} \quad & \sum_{\sigma \in Stencil(\tau)} \left(\frac{1}{\Omega_\sigma} \int_\sigma u_{h,\tau}(\mathbf{x}) d\Omega - U_{h,\sigma} \right)^2 + \sum_{\mathbf{q} \in BdryQuad(\tau)} \left(B(u_{h,\tau}(\mathbf{q}), \nabla u_{h,\tau}(\mathbf{q}), \mathcal{B}) \right)^2 \\ \text{subject to} \quad & \frac{1}{\Omega_\tau} \int_\tau u_h(\mathbf{x}) d\Omega = U_{h,\tau}, \end{aligned} \tag{2.7}$$

where $BdryQuad(\tau)$ is the set of boundary quadrature points for control volume τ, and B is the boundary condition constraint function. The number of control volumes in $Stencil(\tau)$ must be greater than the number of reconstruction coefficients N_{rec}, so that the minimization problem does not become undetermined. To construct $Stencil(\tau)$, all the neighbors at a given topological distance from τ are added to the stencil until the number of stencil members gets bigger than a given value $MinNeigh(k)$. For a well behaved problem on a mesh with high regularity, choosing $MinNeigh(k) = N_{\text{rec}}$ can result in an acceptable solution. However, a value of $MinNeigh(k) = 1.5N_{\text{rec}}$ is chosen to cope with mesh irregularities and oscillatory solution behavior. Figure 2.2 shows a control volume and its reconstruction stencil for different k values.

By introducing the integral of each basis function of every control volume inside itself and its stencil members:

$$I_{\tau\sigma}^i = \int_\sigma \phi_\tau^i(\mathbf{x}) d\Omega \qquad \sigma \in Stencil(\tau) \cup \{ \tau \}, \tag{2.8}$$

the constrained minimization problem in Equations (2.7), can be written in a compact matrix form:

$$\left[\begin{array}{ccc} I_{\tau\tau}^1 & \cdots & I_{\tau\tau}^{N_{\text{rec}}} \\ \hline I_{\tau\sigma_1}^1 & \cdots & I_{\tau\sigma_1}^{N_{\text{rec}}} \\ \vdots & \ddots & \vdots \\ I_{\tau\sigma_{\text{NS}(\tau)}}^1 & \cdots & I_{\tau\sigma_{\text{NS}(\tau)}}^{N_{\text{rec}}} \end{array}\right] \left[\begin{array}{c} a_\tau^1 \\ \vdots \\ a_\tau^{N_{\text{rec}}} \end{array}\right] = \left[\begin{array}{c} U_{h,\tau} \\ \hline U_{h,\sigma_1} \\ \vdots \\ U_{h,\sigma_{\text{NS}(\tau)}} \end{array}\right], \tag{2.9}$$

where the boundary constraints have been neglected for the sake of simplicity. The symbols σ_1, σ_2, …, $\sigma_{\text{NS}(\tau)}$ represent the members of $Stencil(\tau)$. The first row of the matrix is a constraint and must be exactly satisfied, while the other rows correspond to equations that have to be minimized in a least squares fashion. As the left hand side matrix in Equation (2.9) is only dependent on geometric terms, and does not include the average solution values $\mathbf{U}_h$, the solution can be written as:

$$\left[\begin{array}{c} a_\tau^1 \\ \vdots \\ a_\tau^{N_{\text{rec}}} \end{array}\right] = \boldsymbol{A}_\tau^\dagger \left[\begin{array}{c} U_{h,\tau} \\ U_{h,\sigma_1} \\ \vdots \\ U_{h,\sigma_{\text{NS}(\tau)}} \end{array}\right], \tag{2.10}$$

where the matrix $\boldsymbol{A}_\tau^\dagger$ is the pseudo-inverse of the left hand side matrix. A change of variables inspired by the Gaussian Elimination method is used to convert Equation (2.9) into an unconstrained optimiza-

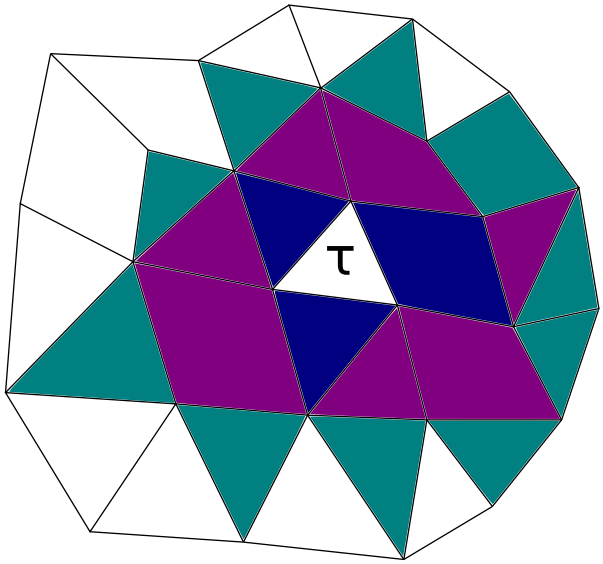


Figure 2.2: Reconstruction stencils for a control volume τ: A set of values $MinNeigh(1) = 3$, $MinNeigh(2) = 9$, and $MinNeigh(3) = 18$ have been used, which have resulted in reconstruction stencils {blue}, {blue, magenta}, and {blue, magenta, cyan} for the 1-exact, 2-exact, and 3-exact reconstruction schemes, respectively.

tion problem [54]. Then the unconstrained problem is solved by singular value decomposition (SVD) [24] to yield the pseudo-inverse matrix. This process has to be evaluated for every control volume, only as a preprocessing step, which prevents it from becoming a bottleneck in our computations.

In the case of equations with discontinuous solutions, shock capturing numerical schemes such as slope limiters [76] or artificial diffusion [69] are required in conjunction with the k-exact reconstruction method to ensure convergence to the correct solution. For more recent implementation of these methods in conjunction with higher-order schemes see [11, 46, 53]. In this thesis, however, the emphasis is on model problems without discontinuities. Thus, no shock capturing methods have been used.

2.3 Studied Equations

This section introduces the equations of interest. Namely, Poisson's, Euler, laminar and Reynolds averaged Navier-Stokes, and the Spalart-Allmaras turbulence closure equations.

2.3.1 Poisson's Equation

The Poisson's equation is a simple yet powerful tool to verify the correct implementation of many algorithms in ANSLib. This equation has a scalar unknown u and a solution-independent source term f in the form of:

$$-\nabla \cdot (\nabla u) = f(\mathbf{x}). \tag{2.11}$$

To simplify the notation, we consider the gradient operator on a scalar function as a row vector. For example, $\nabla u = [\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}]$. Thus the Poisson equation can be recovered from the steady state form of the conservative Equation (2.1) by replacing $\boldsymbol{F} = \nabla u$, $\boldsymbol{Q} = \mathbf{0}$, and $S = f(\mathbf{x})$.

2.3.2 Navier-Stokes Equations

The Navier-Stokes and the continuity equations are widely used for the simulation of fluid flow. In the compressible form of these equations, the vector of unknowns is $\mathbf{u} = [\rho, \rho\mathbf{v}^T, E]^T$, where ρ is the fluid density, $\mathbf{v} = [v_1, v_2, v_3]^T$ is the velocity vector, and E is the total energy. When combined with the ideal gas internal energy and state equations, the nondimensionalized Navier-Stokes equations can be identified by a zero source term vector and flux matrices:

$$\boldsymbol{F} = \begin{bmatrix} \rho\mathbf{v}^T \\ \rho\mathbf{v}\mathbf{v}^T + P\boldsymbol{I} \\ (E+P)\mathbf{v}^T \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 0 \\ \frac{Ma}{Re}\boldsymbol{\tau} \\ (E+P)\boldsymbol{\tau}\mathbf{v} + \frac{1}{\gamma-1}\left(\frac{\mu}{Pr}\right)\nabla T \end{bmatrix}, \tag{2.12}$$

where Ma, Re, Pr, and γ represent the Mach number, the Reynolds number, the Prandtl number, and the specific heat ratio, respectively. T is the temperature, $(.)^T$ denotes matrix transpose, P is the pressure, $\boldsymbol{\tau}$ is the viscous stress matrix, μ is the dimensionless fluid viscosity, and $\boldsymbol{I}$ is the identity matrix. Since the emphasis is on air as the working fluid, the values $\gamma = 1.4$, $Pr = 0.72$ are used, and μ is found from Sutherland's law. The pressure is related to the dependent variables via the formula:

$$P = (\gamma - 1)\left(E - \frac{1}{2}\rho(\mathbf{v}\cdot\mathbf{v})\right). \tag{2.13}$$

Similarly, temperature is related to pressure and density in the form:

$$T = \frac{\gamma P}{\rho}. \tag{2.14}$$

For Newtonian compressible fluids, the viscous stress tensor is related to the velocity as:

$$\boldsymbol{\tau} = 2\mu\left(\frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right) - \frac{1}{3}\,\mathrm{trace}(\nabla\mathbf{v})\boldsymbol{I}\right), \tag{2.15}$$

where $\nabla\mathbf{v}$ represents the gradient of the velocity vector, such that $(\nabla\mathbf{v})_{ij} = \frac{\partial v_i}{\partial x_j}$.

If the viscous terms are neglected, i.e., either $\boldsymbol{Q}$, or equivalently μ is set to zero, the Euler equations would be recovered. The Euler equations are used in this work to asses the capability of the solver in handling reasonably big three-dimensional problems.

2.3.3 Extension to Turbulent Flows

In this thesis, the Reynolds averaged Navier Stokes (RANS) equations are used for modeling turbulent flows, and are coupled with the negative Spalart-Allmaras (negative S-A) turbulence model [6]. This model includes a number of dimensionless empirical constants, listed in Table 2.1, and a few empirical functions, shown by the symbol f and an appropriate subscript, which will be introduced shortly. The nondimensionalized flux matrices for the RANS + negative S-A equations are defined as:

$$\boldsymbol{F} = \begin{bmatrix} \rho \mathbf{v}^T \\ \rho \mathbf{v}\mathbf{v}^T + P\boldsymbol{I} \\ (E+P)\mathbf{v}^T \\ \tilde{\nu}\rho\mathbf{v}^T \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 0 \\ \frac{Ma}{Re}\boldsymbol{\tau} \\ (E+P)\boldsymbol{\tau}\mathbf{v} + \frac{1}{\gamma-1}\left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T}\right)\nabla T \\ -\frac{Ma}{Re\,\sigma}(\mu+\mu_T)\nabla\tilde{\nu} \end{bmatrix}, \tag{2.16}$$

where $\tilde{\nu}$ is the negative S-A working variable, μ_T is the turbulent viscosity, and Pr_T is the turbulent Prandtl number, which has a value of 0.9 for air. The source term of the nondimensionalized RANS + negative S-A is defined as:

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ Diff + \rho(Prod - Dest + Trip) \end{bmatrix}, \tag{2.17}$$

where $Diff$, $Prod$, $Dest$, and $Trip$ represent diffusion, production, destruction, and trip terms, respectively. The viscous stress tensor can be found via the modified equation:

$$\boldsymbol{\tau} = 2(\mu+\mu_T)\left(\frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right) - \frac{1}{3}\,\mathrm{trace}(\nabla\mathbf{v})\boldsymbol{I}\right). \tag{2.18}$$

The turbulent viscosity is expressed as:

$$\mu_T = \begin{cases} \mu' f_{v1}\rho\tilde{\nu} & \tilde{\nu} \geq 0 \\ 0 & \tilde{\nu} < 0 \end{cases}, \tag{2.19}$$

where μ' is the reference value that is used to nondimensionalize the turbulence working variable. In this work, we have used $\mu' = 1000$ to make $\tilde{\nu}$ comparable in size to the other nondimensionalized variables, which enhances the performance of the numerical solver [17]. The production term in Equation (2.17) is defined as:

$$Prod = \begin{cases} c_{b1}(1-f_{t2})\tilde{S}\tilde{\nu} & \tilde{\nu} \geq 0 \\ c_{b1}(1-c_{t3})S & \tilde{\nu} < 0 \end{cases}. \tag{2.20}$$

The destruction term is found from the equation:

$$Dest = \begin{cases} \frac{\mu' Ma}{Re}\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 & \tilde{\nu} \geq 0 \\ -\frac{\mu' Ma}{Re}c_{w1}\left(\frac{\tilde{\nu}}{d}\right)^2 & \tilde{\nu} < 0 \end{cases}, \tag{2.21}$$

Table 2.1: Dimensionless empirical parameters used in the negative S-A model

Name	Value	Name	Value	Name	Value
c_{b1}	0.1355	c_{b2}	0.622	c_{w1}	$\frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}$
c_{w2}	0.3	c_{w3}	2.0	c_{t3}	1.2
c_{t4}	0.5	c_{n1}	16	c_{v1}	7.1
c_{v2}	0.7	c_{v3}	0.9	κ	0.41
σ	0.66				

where d is the minimum distance to wall boundaries. The diffusion term is given as:

$$Diff = \frac{Ma}{Re\,\sigma}\left(\mu' c_{b2}\rho\nabla\tilde{\nu}\cdot\nabla\tilde{\nu} - \frac{\mu}{\rho}(1+\chi f_n)\nabla\rho\cdot\nabla\tilde{\nu}\right), \tag{2.22}$$

where χ is:

$$\chi = \frac{\mu'\rho\tilde{\nu}}{\mu}. \tag{2.23}$$

The trip term in Equation (2.17) models flows that include the laminar to turbulent transition phenomenon. Since the emphasis in this thesis is on fully turbulent flows, the trip term is neglected and set to zero. The vorticity S, and its modified forms, $\tilde{S}$, $\bar{S}$, are found from the equations:

$$\begin{aligned} S &= \sqrt{\frac{1}{2}\sum_i\sum_j\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right)^2} \\ \bar{S} &= \frac{\mu' Ma}{Re}\frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \\ \tilde{S} &= \begin{cases} S + \bar{S} & \bar{S} \geq -c_{v2}S \\ S + \frac{c_{v2}^2 S + c_{v3}\bar{S}}{(c_{v3}-2c_{v2})S - \bar{S}} & \bar{S} < -c_{v2}S \end{cases}. \end{aligned} \tag{2.24}$$

The empirical function f_n ensures the positivity of μ_T, and is defined as:

$$f_n = \begin{cases} 1 & \tilde{\nu} \geq 0 \\ \frac{c_{n1}+\chi^3}{c_{n1}-\chi^3} & \tilde{\nu} < 0 \end{cases}. \tag{2.25}$$

The functions f_{v1}, f_{v2}, f_{v3} are given as:

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \quad f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}} \quad f_{t2} = c_{t3}\exp(-c_{t4}\chi^2), \tag{2.26}$$

and the function f_w is given as:

$$r = \frac{\mu' Ma}{Re}\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \quad g = r + c_{w2}(r^6 - r) \quad f_w = g\left(\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right). \tag{2.27}$$

Having defined the equations of interest, let us now introduce the numerical flux functions.

2.4 Numerical Flux Functions

As discussed earlier, the finite volume method relies on flux functions that not only take into account the discontinuity of the solution along internal faces, but also correctly capture the influence of the boundary conditions. In this section, we will introduce the numerical flux functions used in ANSLib for the solution of the more general RANS + negative S-A equations. Flux functions for the simpler Euler and Poisson equations can simply be derived by omitting the relevant terms from the more general case.

2.4.1 Inviscid Flux

The inviscid fluxes represent propagation of information via finite speed waves, and are convective in nature. Most numerical flux functions seek a solution to the approximate one-dimensional equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \boldsymbol{F}(\mathbf{u})\mathbf{n}}{\partial s} = 0, \tag{2.28}$$

and then use this solution to find the numerical flux. In Equation (2.28), $\boldsymbol{F}$ is the inviscid flux matrix, $\mathbf{n}$ is the face normal vector, and s is the unit of length. The artificial viscosity method [79] adds extra nonlinear dissipative terms to Equation (2.28) to give it an elliptic nature. On the other hand, the Godunov method [23] is based on the exact solution of Equation (2.28), with piecewise constant initial conditions corresponding to the left and right states (the Riemann Problem). Due to the expensive cost of exactly solving the Riemann problem, many researchers have developed approximate solutions such as the Rusanov [61], HLL family [28, 70], and Roe [58] methods. In this thesis, a computationally efficient formulation [57] of the approximate Riemann solver of Roe has been used due to its effectiveness and simplicity. This formulation has the form of

$$\boldsymbol{\mathcal{F}}_I(\mathbf{u}_h^+, \mathbf{u}_h^-) = \frac{1}{2}\Big(\boldsymbol{F}(\mathbf{u}_h^+)\mathbf{n} + \boldsymbol{F}(\mathbf{u}_h^-)\mathbf{n} - \mathbf{D}(\mathbf{u}_h^+, \mathbf{u}_h^-)\Big), \tag{2.29}$$

where $\mathbf{D}$ represents the diffusion vector given by the formula:

$$\mathbf{D}(\mathbf{u}_h^+, \mathbf{u}_h^-) = \begin{bmatrix} |\tilde{\lambda}_2|\Delta\rho + \delta_1 \\ |\tilde{\lambda}_2|\Delta(\rho\mathbf{v}) + \delta_1\tilde{\mathbf{v}} + \delta_2\mathbf{n} \\ |\tilde{\lambda}_2|\Delta E + \delta_1\tilde{H} + \delta_2\tilde{\mathbf{v}}\cdot\mathbf{n} \\ |\tilde{\lambda}_2|\Delta(\rho\tilde{\nu}) + \delta_1\tilde{\tilde{\nu}} \end{bmatrix}. \tag{2.30}$$

Here, $\Delta(.) = (.)_h^+ - (.)_h^-$, and the variables $\lambda_1, \ldots, \lambda_6$ represent the six eigenvalues of the Jacobian matrix $\frac{\partial \boldsymbol{F}(\mathbf{u})\mathbf{n}}{\partial \mathbf{u}}$, expressed as:

$$\lambda_1 = \mathbf{v}\cdot\mathbf{n} - c \quad \lambda_2 = \ldots = \lambda_5 = \mathbf{v}\cdot\mathbf{n} \quad \lambda_6 = \mathbf{v}\cdot\mathbf{n} + c. \tag{2.31}$$

The variable c is the speed of sound, defined as:

$$c = \sqrt{\frac{\gamma P}{\rho}}. \tag{2.32}$$

In Equation (2.30) the symbol $\tilde{(.)}$ represents the Roe average state, evaluated from the equation:

$$\tilde{(.)} = \begin{cases} \sqrt{(.)_h^-(.)_h^+} & (.) = \rho \\ \frac{\sqrt{\rho_h^-}(.)_h^- + \sqrt{\rho_h^+}(.)_h^+}{\sqrt{\rho_h^-}+\sqrt{\rho_h^+}} & (.) = \mathbf{v}, \tilde{\nu}, H \end{cases}, \tag{2.33}$$

where $H = \frac{P+E}{\rho}$ is the enthalpy. Note that $\tilde{\mathbf{v}}$ and $\tilde{\tilde{\nu}}$ are the velocity and the S-A working variable at the Roe average state, respectively. The variables δ_1 and δ_2 in Equation (2.30) are given as:

$$\begin{aligned} \delta_1 &= \frac{\Delta P}{\tilde{c}^2}\left(-|\tilde{\lambda}_2| + \frac{|\tilde{\lambda}_1| + |\tilde{\lambda}_6|}{2}\right) + \frac{\tilde{\rho}}{2\tilde{c}^2}\Delta(\mathbf{v}\cdot\mathbf{n})\left(|\tilde{\lambda}_6| - |\tilde{\lambda}_1|\right) \\ \delta_2 &= \frac{\Delta P}{2\tilde{c}^2}\left(|\tilde{\lambda}_6| - |\tilde{\lambda}_1|\right) + \tilde{\rho}\Delta(\mathbf{v}\cdot\mathbf{n})\left(-|\tilde{\lambda}_2| + \frac{|\tilde{\lambda}_1| + |\tilde{\lambda}_6|}{2}\right). \end{aligned} \tag{2.34}$$

ANSLib evaluates the the inviscid boundary fluxes using the method of characteristics [30]. For wall and symmetry boundary conditions, the mass flux must be zero. Thus, the inviscid flux is evaluated as:

$$\boldsymbol{\mathcal{F}}_B(\mathbf{u}_h, \mathcal{B}) = [0, P_h\mathbf{n}^T, 0]^T \qquad \mathcal{B} \text{ is symmetry or wall.} \tag{2.35}$$

On farfield boundaries, however, the boundary flux formulation changes based on the sign of the normal velocity, $\mathbf{v}\cdot\mathbf{n}$. In this work, we consider subsonic inflow and outflow boundary conditions which are identified by the inequalities $0 \leq \mathbf{v}\cdot\mathbf{n} < c$, and $-c < \mathbf{v}\cdot\mathbf{n} \leq 0$, respectively. In either case, the boundary flux is evaluated in terms of an intermediate state, $\mathbf{u}_h^*$, defined as a function of both the boundary conditions and the discrete solution:

$$\boldsymbol{\mathcal{F}}_B(\mathbf{u}_h, \mathcal{B}) = \boldsymbol{F}(\mathbf{u}_h^*)\mathbf{n} \qquad \mathbf{u}_h^* = (\mathbf{u}_h, \mathcal{B}) \qquad \mathcal{B} \text{ is inflow or outflow.} \tag{2.36}$$

For subsonic inflow, five values of farfield turbulence working variable $\tilde{\nu}_{\text{far}}$, total temperature T_t, total pressure P_t, side slip angle ψ, and angle of attack α must be specified. Subsequently, the intermediate

state $\mathbf{u}_h^*$ can be found as:

$$\begin{gathered} P_h^* = P_h \quad T_h^* = T_t \left(\frac{P_h^*}{P_t}\right)^{\frac{\gamma-1}{\gamma}} \quad \rho_h^* = \gamma \frac{P_h^*}{T_h^*} \quad \tilde{\nu}_h^* = \tilde{\nu}_{\text{far}} \\ \|\mathbf{v}_h^*\|_2 = \sqrt{\frac{2}{\gamma-1}\left(\frac{T_t}{T_h^*} - 1\right)} \\ v_{h1}^* = \|\mathbf{v}_h^*\|_2 \cos\alpha \cos\psi \quad v_{h2}^* = \|\mathbf{v}_h^*\|_2 \sin\alpha \cos\psi \quad v_{h3}^* = \|\mathbf{v}_h^*\|_2 \sin\psi. \end{gathered} \tag{2.37}$$

For subsonic outflow, only the back-pressure value P_b must be specified. The intermediate state will then be defined as:

$$\rho_h^* = \rho_h \quad \mathbf{v}_h^* = \mathbf{v}_h \quad P_h^* = P_b \quad \tilde{\nu}_h^* = \tilde{\nu}_h. \tag{2.38}$$

In this thesis, the same values are chosen for $\tilde{\nu}_{\text{far}}$, T_t, P_t, and P_b as the work of Jalali and Ollivier-Gooch [32]:

$$\tilde{\nu}_{\text{far}} = \frac{3}{\mu'} \quad T_t = 1 + \frac{\gamma-1}{2} Ma^2 \quad P_t = \frac{1}{\gamma} T_t^{\frac{\gamma}{\gamma-1}} \quad P_b = \frac{1}{\gamma}. \tag{2.39}$$

2.4.2 Viscous Flux

The viscous flux functions are evaluated in a different manner compared to their inviscid counterparts. When evaluating internal flux values, an intermediate state $\mathbf{u}_h^*$ is used in the form:

$$\boldsymbol{\mathcal{Q}}_I(\mathbf{u}_h^+, \nabla\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla\mathbf{u}_h^-) = \boldsymbol{Q}(\mathbf{u}_h^*, \nabla\mathbf{u}_h^*)\mathbf{n}. \tag{2.40}$$

Finding the intermediate solution as the numerical average of the left and right states,

$$\mathbf{u}_h^* = \frac{1}{2}\left(\mathbf{u}_h^+ + \mathbf{u}_h^-\right), \tag{2.41}$$

results in a sufficiently accurate solution [51]. When evaluating the intermediate solution gradient, however, simple averaging may lead to spurious solutions and instabilities. Nishikawa [50] suggested the following modified formula, in line with the interior penalty formulation [9] used in the DG framework:

$$\nabla\mathbf{u}_h^* = \frac{1}{2}\left(\nabla\mathbf{u}_h^+ + \nabla\mathbf{u}_h^-\right) + \eta\left(\frac{\mathbf{u}_h^+ - \mathbf{u}_h^-}{\|\mathbf{x}_{\tau+} - \mathbf{x}_{\tau-}\|_2}\right)\mathbf{n}, \tag{2.42}$$

where $\mathbf{x}_{\tau-}$ and $\mathbf{x}_{\tau+}$ are the reference locations of the adjacent control volumes, respectively, and η is a heuristic damping factor, known as the jump term. Jalali et al. [34] have numerically tested different values of η in high-order finite volume simulations, and have recommended a value of $\eta = 1$, which is also used in this thesis.

The boundary viscous fluxes are evaluated in the same manner as reference [32], by simply replacing the interior state into the viscous flux matrix, $\boldsymbol{\mathcal{Q}}_B(\mathbf{u}_h, \nabla\mathbf{u}_h, \mathcal{B}) = \boldsymbol{Q}(\mathbf{u}_h, \nabla\mathbf{u}_h)\mathbf{n}$. The boundary condi-

tions are then enforced through a combination of soft and hard constraints. For adiabatic walls, the velocity, heat flux, and turbulence working variable have to be zero, which are expressed as:

$$\begin{aligned} &\text{Hard constraints:} \quad \mathbf{v}_h = 0 \quad \tilde{\nu}_h = 0 \\ &\text{Soft constraint:} \quad \nabla T_h \cdot \mathbf{n} = 0 \end{aligned} \qquad \mathcal{B} \text{ is adiabatic wall,} \tag{2.43}$$

where the hard constraints are applied to the k-exact reconstruction process through the boundary condition constraint function B. The soft constraint, however, is applied by simply replacing the value of $\nabla T_h \cdot \mathbf{n}$ with zero when evaluating the boundary flux. On symmetry boundaries, the heat flux, normal derivative of the turbulence working variable, and the tangential viscous force have to be zero, which are enforced via the following soft constraints:

$$\text{Soft constraints:} \quad \nabla \tilde{\nu}_h \cdot \mathbf{n} = 0 \quad \nabla T_h \cdot \mathbf{n} = 0 \quad \Big((\mathbf{n}^T \boldsymbol{\tau}_h \mathbf{n}) \boldsymbol{I} - \boldsymbol{\tau}_h \Big) \mathbf{n} = 0 \quad \mathcal{B} \text{ is symmetry.} \tag{2.44}$$

Finally, no constraints are applied when evaluating the viscous flux function on the farfield boundaries.

Chapter 3

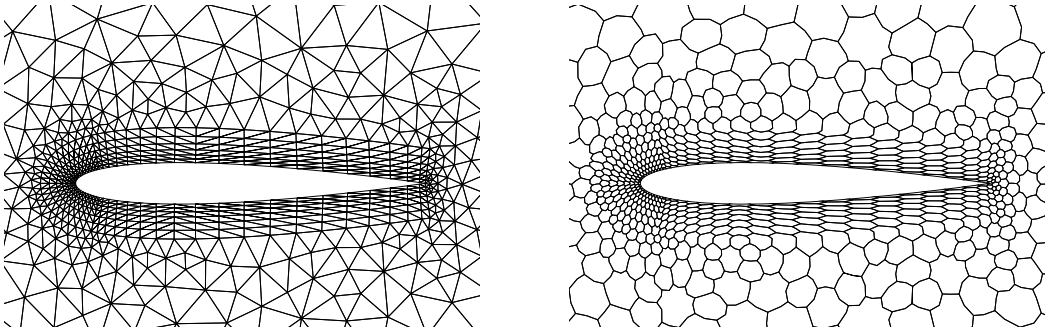
Three-Dimensional Mesh Processing

The finite volume method requires the subdivision of the domain into a set of control volumes, which are obtained from a mesh. There are two different approaches for constructing the control volumes: cell-centered, and vertex-centered (cell-vertex). In the cell-centered approach, every cell of the mesh is considered as a control volume, as shown in Figure 3.1a. Conversely, the vertex-centered approach associates a control volume to each vertex of the mesh, and is dependent on the definition of the control volume faces. Figure 3.1b shows an example of vertex-centered control volumes, which are created by connecting the barycenter of each triangle to the midpoints of its edges. This method for constructing the vertex-centered control volumes is known as the median-dual approach.

The only data that the solver requires about the geometry of the control volumes is their adjacency and quadrature information. Thus, relevant data structures and algorithms have to be implemented in a numerical solver package to offer access to such information, with reasonable time and memory cost. Although the cell-centered and cell-vertex methods are both straightforward to implement in two dimensions, three-dimensional implementation of the latter is more difficult to code, and requires more quadrature points for integration at a given order of accuracy. As a result, the cell-centered method has been chosen for the purpose of this thesis, and is implemented in ANSLib. This chapter discusses the preprocessing steps required for a three-dimensional cell-centered finite volume solver.

3.1 Element Mapping and Quadrature

In terms of solution accuracy, hexahedral and quadrilateral meshes are known to have superior properties in three and two dimensions, respectively, compared to tetrahedral and triangular meshes. Although structured mesh generation techniques can create such grids, they can be expensive in terms of time resources, since applying these algorithms to complex geometries requires a considerable amount of human input. Conversely, unstructured mesh generation techniques are rather automatic, but are not able to handle complex geometries without resorting to robust algorithms that only create simplex cells, such as the Delaunay mesh generation methods [18]. Therefore, simulation grids are usually com-



(a) **(b)**

Figure 3.1: Schematic of control volumes: (a) cell-centered, (b) median-dual cell-vertex.

posed of a mixture of different cell types. The purpose of this section is to introduce the connectivity and quadrature formulas for the control volumes and their faces in a computational mesh. Hereinafter, both the control volumes and their faces will be referred to as elements.

The process starts with mapping each element E to its reference version $\hat{E}$, which resides in a nondimensional space. The mapping is a polynomial of degree l, that must guarantee geometric continuity across neighboring elements. Thus, it is usually constructed from Lagrangian basis functions (polynomials). Using the $\hat{(.)}$ notation for the reference space, the mapping can be written as:

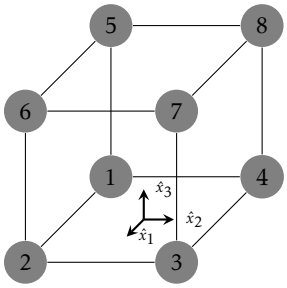
$$\mathbf{x}(\hat{\mathbf{x}}) = \sum_{i=1}^{N_{\text{lag}}} \mathbf{y}_i \hat{\psi}_i(\hat{\mathbf{x}}) \quad \hat{\mathbf{x}} \in \hat{E}, \tag{3.1}$$

where $\hat{\psi}_i$ and $\mathbf{y}_i$ are the Lagrange basis functions and nodal locations, respectively, and N_{lag} is the number of Lagrange points. Hexahedra, prisms and tetrahedra are the most commonly used three-dimensional elements in both finite element and finite volume simulations. Pyramids, on the other hand, are not very popular, and are used for making transitions between different element types in a mixed mesh. Quadrilateral and triangular elements are also used to represent the control volume faces. Figure 3.2 shows the first order ($l = 1$) versions of these elements in the reference space. A well defined mesh must provide the location of every node. Furthermore, it must store the local to global node numbering for every element, along with its type and interpolation order.

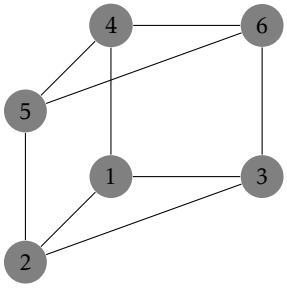
For a reference element, a qth-order accurate quadrature rule, consists of N_{qua} point locations $\hat{\mathbf{z}}_i$, and weights $\hat{w}_i$, such that for every polynomial $\hat{P}(\hat{\mathbf{x}})$ of degree less than or equal to $q-1$:

$$\int_{\hat{E}} \hat{P}(\hat{\mathbf{x}}) d\hat{E} = \sum_{i=1}^{N_{\text{qua}}} \hat{P}(\hat{\mathbf{z}}_i) \hat{w}_i. \tag{3.2}$$

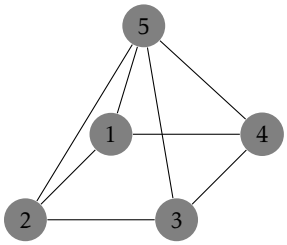
These quadrature rules are tabulated for different orders and types of elements in the literature, such as the work of Solin et al. [67]. By a change of variables, quadrature rules can be constructed for elements



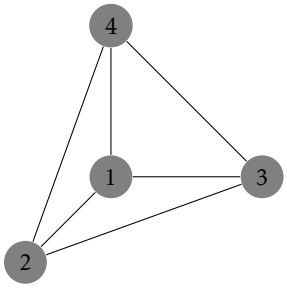
$$\hat{\mathbf{y}}_1 = (-1,-1,-1) \quad \hat{\psi}_1 = (1-\hat{x}_1)(1-\hat{x}_2)(1-\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_2 = (+1,-1,-1) \quad \hat{\psi}_2 = (1+\hat{x}_1)(1-\hat{x}_2)(1-\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_3 = (+1,+1,-1) \quad \hat{\psi}_3 = (1+\hat{x}_1)(1+\hat{x}_2)(1-\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_4 = (-1,+1,-1) \quad \hat{\psi}_4 = (1-\hat{x}_1)(1+\hat{x}_2)(1-\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_5 = (-1,-1,+1) \quad \hat{\psi}_5 = (1-\hat{x}_1)(1-\hat{x}_2)(1+\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_6 = (+1,-1,+1) \quad \hat{\psi}_6 = (1+\hat{x}_1)(1-\hat{x}_2)(1+\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_7 = (+1,+1,+1) \quad \hat{\psi}_7 = (1+\hat{x}_1)(1+\hat{x}_2)(1+\hat{x}_3)/8$$
$$\hat{\mathbf{y}}_8 = (-1,+1,+1) \quad \hat{\psi}_8 = (1-\hat{x}_1)(1+\hat{x}_2)(1+\hat{x}_3)/8$$



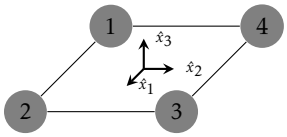
$$\hat{\mathbf{y}}_1 = (0,0,-1) \quad \hat{\psi}_1 = (1-\hat{x}_3)(1-\hat{x}_2-\hat{x}_1)/2$$
$$\hat{\mathbf{y}}_2 = (1,0,-1) \quad \hat{\psi}_2 = (1-\hat{x}_3)\hat{x}_1/2$$
$$\hat{\mathbf{y}}_3 = (0,1,-1) \quad \hat{\psi}_3 = (1-\hat{x}_3)\hat{x}_2/2$$
$$\hat{\mathbf{y}}_4 = (0,0,+1) \quad \hat{\psi}_4 = (1+\hat{x}_3)(1-\hat{x}_2-\hat{x}_1)/2$$
$$\hat{\mathbf{y}}_5 = (1,0,+1) \quad \hat{\psi}_5 = (1+\hat{x}_3)\hat{x}_1/2$$
$$\hat{\mathbf{y}}_6 = (0,1,+1) \quad \hat{\psi}_6 = (1+\hat{x}_3)\hat{x}_2/2$$



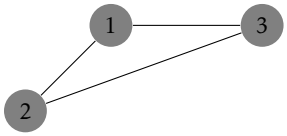
$$\hat{\mathbf{y}}_1 = (-1,-1,0) \quad \hat{\psi}_1 = (\hat{x}_3+\hat{x}_1-1)(\hat{x}_3+\hat{x}_2-1)/4((1-\hat{x}_3)+\epsilon)$$
$$\hat{\mathbf{y}}_2 = (+1,-1,0) \quad \hat{\psi}_2 = (\hat{x}_3-\hat{x}_1-1)(\hat{x}_3+\hat{x}_2-1)/4((1-\hat{x}_3)+\epsilon)$$
$$\hat{\mathbf{y}}_3 = (+1,-1,0) \quad \hat{\psi}_3 = (\hat{x}_3-\hat{x}_1-1)(\hat{x}_3-\hat{x}_2-1)/4((1-\hat{x}_3)+\epsilon)$$
$$\hat{\mathbf{y}}_4 = (+1,-1,0) \quad \hat{\psi}_4 = (\hat{x}_3+\hat{x}_1-1)(\hat{x}_3-\hat{x}_2-1)/4((1-\hat{x}_3)+\epsilon)$$
$$\hat{\mathbf{y}}_5 = (0,0,1) \quad \hat{\psi}_5 = \hat{x}_3 \quad \epsilon = 10^{-20}$$



$$\hat{\mathbf{y}}_1 = (0,0,0) \quad \hat{\psi}_1 = 1-\hat{x}_1-\hat{x}_2-\hat{x}_3$$
$$\hat{\mathbf{y}}_2 = (1,0,0) \quad \hat{\psi}_2 = \hat{x}_1$$
$$\hat{\mathbf{y}}_3 = (0,1,0) \quad \hat{\psi}_3 = \hat{x}_2$$
$$\hat{\mathbf{y}}_4 = (0,0,1) \quad \hat{\psi}_4 = \hat{x}_3$$



$$\hat{\mathbf{y}}_1 = (-1,-1,0) \quad \hat{\psi}_1 = (1-\hat{x}_1)(1-\hat{x}_2)/4$$
$$\hat{\mathbf{y}}_2 = (+1,-1,0) \quad \hat{\psi}_2 = (1+\hat{x}_1)(1-\hat{x}_2)/4$$
$$\hat{\mathbf{y}}_3 = (+1,+1,0) \quad \hat{\psi}_3 = (1+\hat{x}_1)(1+\hat{x}_2)/4$$
$$\hat{\mathbf{y}}_4 = (-1,+1,0) \quad \hat{\psi}_4 = (1-\hat{x}_1)(1+\hat{x}_2)/4$$



$$\hat{\mathbf{y}}_1 = (0,0,0) \quad \hat{\psi}_1 = 1-\hat{x}_1-\hat{x}_2$$
$$\hat{\mathbf{y}}_2 = (1,0,0) \quad \hat{\psi}_2 = \hat{x}_1$$
$$\hat{\mathbf{y}}_3 = (0,1,0) \quad \hat{\psi}_3 = \hat{x}_2$$

Figure 3.2: First order reference Lagrange elements and polynomials. From top to bottom: hexahedron, prism, pyramid, tetrahedron, quadrilateral, and triangle.

in the physical space. For a three-dimensional element, we can write:

$$\int_E f(\mathbf{x}) dE = \int_{\hat{E}} f(\mathbf{x}(\hat{\mathbf{x}})) \det(\boldsymbol{J}(\hat{\mathbf{x}})) d\hat{E} \approx \sum_{i=1}^{N_{\text{qua}}} f(\mathbf{x}(\hat{\mathbf{z}}_i)) \det(\boldsymbol{J}(\hat{\mathbf{z}}_i)) \hat{w}_i. \tag{3.3}$$

Here, $\boldsymbol{J} = \frac{\partial \mathbf{x}}{\partial \hat{\mathbf{x}}}$ represents the Jacobian of the transformation given in Equation (3.1). A similar statement can be made for two-dimensional elements, giving way to the following procedure for finding quadrature rules in the physical space:

$$\begin{aligned} &\text{2D Element:} \quad w_i = \|\partial_{\hat{x}_1} \mathbf{x} \times \partial_{\hat{x}_2} \mathbf{x}\|_2 \hat{w}_i \quad \mathbf{z}_i = \mathbf{x}(\hat{\mathbf{z}}_i) \\ &\text{3D Element:} \quad w_i = \det(\boldsymbol{J}(\hat{\mathbf{z}}_i)) \hat{w}_i \qquad\quad \mathbf{z}_i = \mathbf{x}(\hat{\mathbf{z}}_i) \end{aligned}. \tag{3.4}$$

It is noteworthy to mention that the higher the interpolation order of an element, l, the higher the values of q are required to evaluate an integral up to a certain order of accuracy. This is due to the presence of the term $\det(\boldsymbol{J})$ in Equation (3.3) that prevents a quadrature rule to maintain its nominal order of accuracy in the physical space. Furthermore, for integrals on the faces that involve the normal vector, the following formula can be used:

$$\mathbf{n} = \frac{\partial_{\hat{x}_1} \mathbf{x} \times \partial_{\hat{x}_2} \mathbf{x}}{\|\partial_{\hat{x}_1} \mathbf{x} \times \partial_{\hat{x}_2} \mathbf{x}\|_2}. \tag{3.5}$$

In this thesis we have used the Lagrange polynomials and reference element quadrature rules that are provided by the libMesh finite element library [36].

3.2 Creating Curved Anisotropic Meshes

High-order numerical discretization schemes must correctly account for the curvature of domain boundaries to maintain their nominal order of accuracy [14]. As conventional mesh generators create only meshes with planar faces, modification schemes are required before such meshes can be used for a high-order numerical simulation. This modification can be as simple as just replacing the boundary faces with higher-order representations, as in isotropic meshes. Successful resolution of quantities of interest in turbulent flows, however, requires highly anisotropic meshes that have big spacing in the wall tangent direction compared to small spacing in the wall normal direction. Replacing only the boundary faces in this case would create self intersecting invalid domain subdivisions. Thus, the curvature of the boundary faces must somehow be propagated throughout the domain to prevent mesh tangling.

For structured meshes, a simple remedy is to agglomerate a fine mesh and use the extra points to define the higher order representation of the cells and their faces. Although this method is effective, its limitation in only working with structured meshes confines its use to verification and benchmark problems. A more general approach is to use a solid mechanics analogy, and model the faceted mesh

as an elastic solid. Then, the boundary of the solid can be deformed to match its curved representation, leading to an internal deformation which is consequently used to define the curvature of the inner faces and cells. Hartmann and Leicht [29] gave a detailed summary of this family of methods. They also introduced modifications that greatly reduce the computational cost, while increasing the quality of the final mesh produced. Another approach was presented by Toulorge et al. [71], which is based on minimizing a novel nonlinear energy function. Although this approach is more complicated than using a linear elasticity analogy, the authors argue that it has better properties in terms of robustness and efficiency. Jalali and Ollivier-Gooch [32] have developed a simplified two-dimensional linear elasticity solver, which is currently used in ANSLib to generate high-order curved meshes for two-dimensional turbulent flow simulations. In this section, their approach will be extended to work with three-dimensional unstructured meshes.

A linear elastic solid is governed by the Navier equations, which are derived from the conservation of linear momentum, and expressed as [37]:

$$-\nabla \cdot \boldsymbol{\sigma} = 0, \tag{3.6}$$

where $\boldsymbol{\sigma}$ is the stress tensor, and is defined as:

$$\boldsymbol{\sigma} = \frac{\mu}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) - \frac{\lambda}{3}\,\mathrm{trace}(\nabla \mathbf{u})\boldsymbol{I}, \tag{3.7}$$

where $\mathbf{u} = [u_1, u_2, u_3]^T$ is the displacement vector, and the variables μ and λ are the Lame's coefficients. The elastic solid properties are usually reported in terms of the Young's modulus, E, and Poisson's ratio, ν, which are related to the Lame's coefficients as:

$$\mu = \frac{E}{2(1+\nu)} \quad \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}. \tag{3.8}$$

On the boundaries, either the displacement, the normal force, or a linear relation between them has to be specified, which results in Dirichlet, Neumann, or Robin boundary conditions, respectively. These conditions can all be expressed through a single equation:

$$\boldsymbol{Q}(\mathbf{x})\mathbf{u} + \boldsymbol{\sigma}\mathbf{n} = \mathbf{f}(\mathbf{x}) \quad \mathbf{x} \in \partial\Omega, \tag{3.9}$$

where $\mathbf{f}$ is the normal force, and $\boldsymbol{Q}$ is a generalized spring constant. In the curved mesh generation process, only the Dirichlet boundary conditions are of interest, which can be recovered by setting $\boldsymbol{Q} = \frac{1}{\epsilon}\boldsymbol{I}$ and $\mathbf{f}(\mathbf{x}) = \frac{1}{\epsilon}\mathbf{u}_b(\mathbf{x})$ in Equation (3.9). Here, $\mathbf{u}_b(\mathbf{x})$ is the prescribed boundary displacement, and ϵ is a small number such as 10^{-10}.

The solution method for the Navier equations and their boundary conditions will now be presented. To ease the notation, the Einstein summation convention is used for indices i, j, k, and l, which are reserved only for the geometric dimensions. In the first solution step, Equations (3.6) and (3.7) are

combined to yield:

$$-\frac{\partial}{\partial x_k}\left(K_{ijkl}\frac{\partial u_j}{\partial x_l}\right)=0 \qquad i=1\cdots N_{\text{dim}}. \tag{3.10}$$

where K_{ijkl} are the components of the fourth-order stiffness tensor:

$$K_{ijkl}=\lambda\delta_{ik}\delta_{jl}+\mu(\delta_{ij}\delta_{kl}+\delta_{il}\delta_{jk}). \tag{3.11}$$

The next step involves the use of the continuous Galerkin finite element method to solve the governing equations. The numerical solution $\mathbf{u}_h(\mathbf{x})$ will be defined as the linear superposition of a set of basis functions,

$$\mathbf{u}_h(\mathbf{x})=\sum_{r=1}^{N_{\text{DOF}}/N_{\text{dim}}}\mathbf{U}_{h,r}\psi_r(\mathbf{x}), \tag{3.12}$$

where ψ_r is the rth finite element basis function. Note that the finite element basis functions are not necessarily the same as Lagrange polynomials that are used to represent the curved geometry in Equation (3.1). Moreover, $\mathbf{U}_{h,r}$ is the rth sub-component of the vector of degrees of freedom, $\mathbf{U}_h$, and is expressed as:

$$\mathbf{U}_{h,r}=[(U_{h,r})_1,(U_{h,r})_2,(U_{h,r})_3]^T. \tag{3.13}$$

Subsequently, $\mathbf{u}$ is replaced with $\mathbf{u}_h$ in Equation (3.10), and the result is multiplied by every basis function ψ_r, which when integrated over the whole domain gives:

$$-\int_\Omega\frac{\partial}{\partial x_k}\left(K_{ijkl}\frac{\partial u_{h,j}}{\partial x_l}\right)\psi_r d\Omega=0 \quad i=1\cdots N_{\text{dim}} \quad r=1\cdots N_{\text{DOF}}.$$

Then, using the integration by parts theorem and Equation (3.7), we arrive at the discretized weak form:

$$\sum_{s=1}^{N_{\text{DOF}}}\left(\int_\Omega\frac{\partial\psi_r}{\partial x_k}K_{ijkl}\frac{\partial\psi_s}{\partial x_l}d\Omega+\int_{\partial\Omega}\psi_r Q_{ij}\psi_s dS\right)(U_{h,s})_j=\int_{\partial\Omega}f_i\psi_r dS, \tag{3.14}$$

for every $i\leq N_{\text{dim}}$ and $r\leq N_{\text{DOF}}$, which is a linear system of equations for finding $\mathbf{U}_h$.

The described solution scheme is implemented using the libMesh finite element library [36] in this thesis. The hierarchical polynomials [22] of order p are chosen as the basis functions, where p is an input parameter. LibMesh supports quadrature rules of arbitrary orders of accuracy for assembling the discretized system of equations. Therefore, a quadrature scheme accurate up to order $(2p+1)$ is used, which ensures exact evaluation of all integrals, so that we do not have to worry about the effect of quadrature error on the final solution.

To illustrate the performance of the implemented method, a model problem is presented, with a simple geometry resembling a three-dimensional airfoil. This geometry along with its anisotropic mesh are shown in Figure 3.3. The mesh is constituted of 1288 hexahedra. The curved part of the boundary is analytically parameterized as:

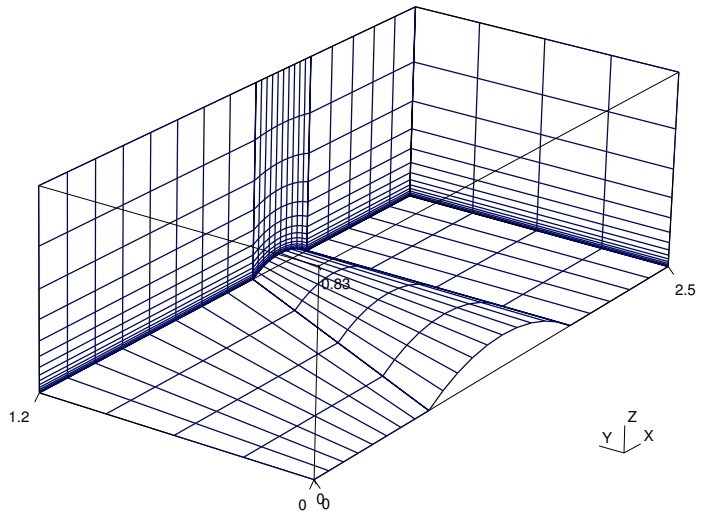


Figure 3.3: Geometry of the curved mesh generation test case

$$\begin{aligned}
&\mathbf{y}(u,v) = \left[u + (R_0 - bu)\sin(v), cu, (R_0 - bu)\cos(v) - H_0\left(1 - \frac{bu}{R_0}\right)\right]^T \\
&0 < u < 1.25 \quad -\frac{\pi}{6} < v < \frac{\pi}{6} \\
&R_0 = 1 \quad H_0 = \cos(\pi/6) \quad b = 2 \quad c = 4,
\end{aligned} \tag{3.15}$$

where u and v are parameterization variables, and $\mathbf{y}$ is a point on the surface. To correctly capture the curvature of the boundary, we need to project every point $\mathbf{x}$ on the boundary of the faceted mesh to the surface of the original geometry. Then, the boundary condition simply becomes that of Dirichlet with $\mathbf{u}_b(\mathbf{x}) = Proj(\mathbf{x}) - \mathbf{x}$. The projection operator would be identity on the planar parts of the boundary. On the curved part, however, we seek to find the point $Proj(\mathbf{x}) = \mathbf{y}(u,v)$, such that the line $\mathbf{xy}$ is perpendicular to the surface, i.e., we seek the unknowns $[u,v,d]^T$ that satisfy:

$$\mathbf{x} - \mathbf{y}(u,v) - d\mathbf{n}(u,v) = 0, \tag{3.16}$$

where d is the distance between the points $\mathbf{x}$ and $\mathbf{y}$, and $\mathbf{n}$ is the surface normal vector,

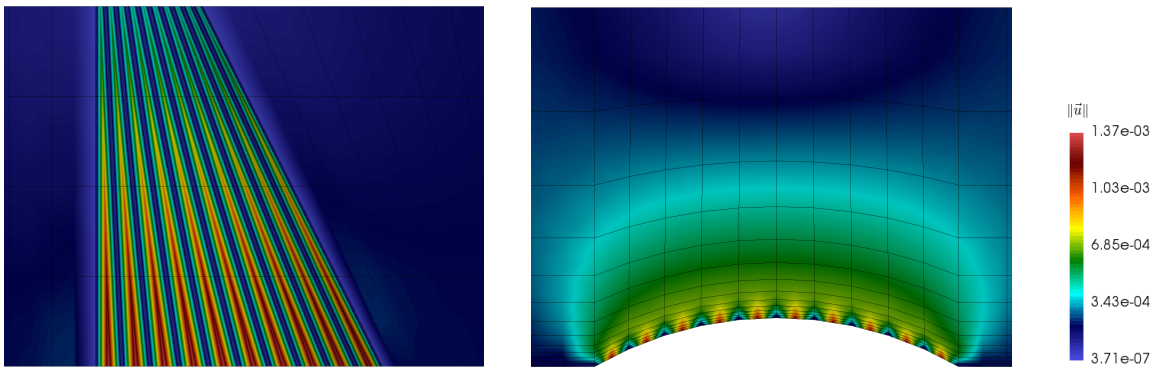
$$\mathbf{n}(u,v) = \frac{\partial_u \mathbf{y} \times \partial_v \mathbf{y}}{\|\partial_u \mathbf{y} \times \partial_v \mathbf{y}\|_2}. \tag{3.17}$$

The projection problem can be solved using the Newton's method, globalized by line search, which fully defines the boundary conditions. Finally, the values $E = 1$ and $\nu = 0.3$ have been used in the Navier equations. Note that due to the use of only Dirichlet boundary conditions, the value of E cancels out and is insignificant as long as it is uniform throughout the domain. However, changing ν in the admissible range $(0, 0.5)$ will result in slightly different displacement fields.

To solve the presented model problem, we have used basis functions of orders $p = 2, 3$, and a Con-

Table 3.1: Performance of the linear elasticity solver

p	N_{DOF}	Assembly Time(s)	Linear Solve Time(s)	Total Time(s)
2	36,801	4.1	7.8	12.5
3	117,390	52.4	98.0	153.6



(a) (b)

Figure 3.4: Curved mesh displacement magnitude: (a) $z = 0$ view, (b) $y = 0.25$ cross section.

jugate Gradient (CG) solver, to set up and solve Equation (3.14), respectively. The problem size and solution time are listed in Table 3.1. Due to the nature of higher-order finite element methods, it is not surprising that the number of DOF and computational time have increased substantially for the $p = 3$ scheme. Figure 3.4 shows the magnitude of the displacement field obtained using quadratic hierarchical basis functions. As expected, the displacement field has the largest magnitude near the curved wall, yet vanishes as one moves towards the other planar parts of the boundary. Moreover, there is no displacement at the vertices of the mesh, as they are already on the true curved boundary. On the other hand, there is a big displacement in the middle of the high aspect ratio cells that are near the curved wall, i.e., where the difference between the true boundary and its planar approximation is the greatest. The residual of the linear system (3.14) per CG iteration, is shown in Figure 3.5. Convergence is rather smooth, and the residual drops below 10^{-11} in 18 and 25 iterations for the quadratic and cubic basis functions, respectively. The increase in number of linear iterations for the $p = 3$ scheme is due to its stiffer linear system and higher number of degrees of freedom.

3.3 Modified Basis Functions for Highly Anisotropic Meshes

If Cartesian coordinate monomials, $\phi^i(\mathbf{x})$, are used as basis functions for the k-exact reconstruction scheme, numerical difficulties arise when dealing with anisotropic meshes. The problem shows itself in two situations. First, the condition number of the left hand side matrix of Equation (2.9) soars drastically, even up to $O(\frac{1}{\epsilon})$, where ϵ is the machine zero. This can in turn introduce an error of order $O(1)$ in the reconstruction coefficients, and deteriorate the accuracy of $\mathbf{u}_h$. Secondly, the k-exactness

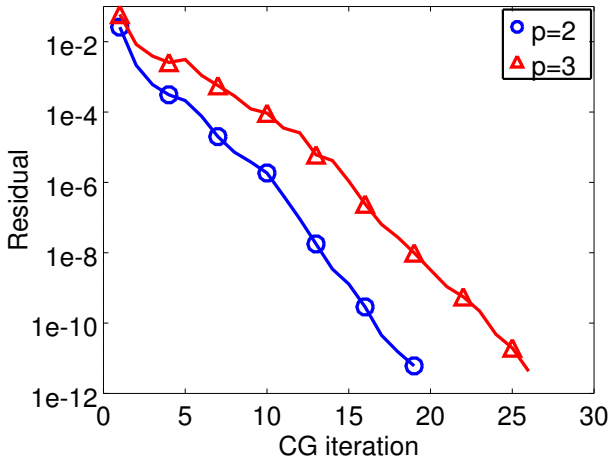


Figure 3.5: Residual per iteration for the linear elasticity solver

of the reconstruction scheme is lost, i.e., for a function v and its projection on the discrete solution space v_h, the equality $\|v - v_h\| = O(h^{k+1})$ might not hold anymore. In this section, we will introduce the remedies proposed by Jalali and Ollivier-Gooch [32] to mitigate these difficulties, along with their generalization to three dimensions.

The key is to use two new sets of reconstruction basis functions that have better conditioning and interpolation properties for anisotropic meshes. The first group is the set of principal coordinate basis functions $\varphi^{+i}(\mathbf{x})$, which are defined as:

$$\left\{ \varphi_\tau^{+i}(\mathbf{x}) \,\middle|\, i = 1 \dots N_{\text{rec}} \right\} = \left\{ \frac{1}{a!b!c!} (w_{\tau 1}(\mathbf{x}))^a (w_{\tau 2}(\mathbf{x}))^b (w_{\tau 3}(\mathbf{x}))^c \,\middle|\, a + b + c \leq k \right\}. \tag{3.18}$$

In this equation, $\mathbf{w}_\tau(\mathbf{x})$ is the principal coordinate of control volume τ evaluated at point $\mathbf{x}$. To find this coordinate transformation, the moment of inertia tensor $\boldsymbol{I}_\tau$ must be found,

$$(I_\tau)_{ij} = \int_\tau (x_i - x_{\tau i})(x_j - x_{\tau j}) d\Omega. \tag{3.19}$$

As this tensor is symmetric, it can be diagonalized, $\boldsymbol{I}_\tau = \boldsymbol{Q}_\tau \boldsymbol{\Lambda}_\tau \boldsymbol{Q}_\tau^T$, where $\boldsymbol{Q}$ and $\boldsymbol{\Lambda}$ are its matrix of right eigenvectors, and diagonal matrix of eigenvalues, respectively. Then, the principal coordinates are evaluated from: $\mathbf{w}_\tau(\mathbf{x}) = \boldsymbol{Q}_\tau(\mathbf{x} - \mathbf{x}_\tau)$. Although these basis functions span the same polynomial space as the Cartesian monomials, $\phi^i(\mathbf{x})$, they greatly reduce the condition number of the left hand side matrix in Equation (2.9). Our heuristic approach is to use these basis functions for control volumes which have a high aspect ratio, yet their faces are almost planar.

For control volumes that are close to wall boundaries and have high distortion, a more complicated approach must be adopted. As a starting point, let us assume that for every control volume τ with such property, we somehow can find a vector function $(d_\tau, t_{\tau 1}, t_{\tau 2})$, such that at every point $\mathbf{x}$ in the vicinity of the control volume, $\nabla d_\tau(\mathbf{x})$ is perpendicular to the closest wall boundary. Furthermore, $\nabla t_{\tau 1}(\mathbf{x})$ and $\nabla t_{\tau 2}(\mathbf{x})$ must be perpendicular to $\nabla d_\tau(\mathbf{x})$, and form an orthogonal basis for $\mathbb{R}^3$. The vector $(d_\tau, t_{\tau 1}, t_{\tau 1})$

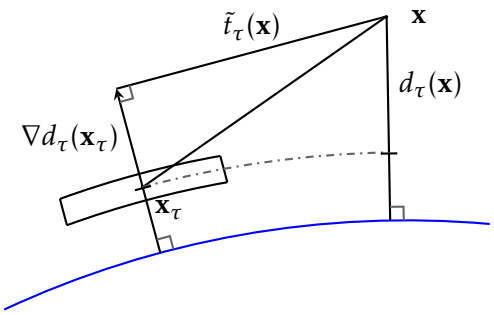


Figure 3.6: Construction of approximate wall coordinates for a two-dimensional control volume τ

is then denoted as the wall coordinates of control volume τ. In two dimensions, the wall coordinate vector shrinks to (d_τ, t_τ), so a rather simple strategy can be used to find it. The normal coordinate, $d_\tau(\mathbf{x})$, is set to the distance to wall of point $\mathbf{x}$ minus that of the reference point of τ. For the tangential coordinate t_τ, however, a compact analytic relation does not exist. Thus, the approximate value $\tilde{t}_\tau$ is used,

$$\tilde{t}_\tau(\mathbf{x}) = \|(\mathbf{x}-\mathbf{x}_\tau) - ((\mathbf{x}-\mathbf{x}_\tau)\cdot\nabla d_\tau(\mathbf{x}_\tau))\nabla d_\tau(\mathbf{x}_\tau)\|_2, \tag{3.20}$$

which is the length of the projection of $\mathbf{x}-\mathbf{x}_\tau$ onto the perpendicular plane of $\nabla d_\tau(\mathbf{x}_\tau)$. Figure 3.6 schematically shows the construction of the approximate wall coordinates for a sample control volume. Although it is possible to find these coordinates for every quadrature point in the vicinity of the control volume of interest, finding their derivatives can be complicated, or even impossible. Thus, yet another coordinate transformation is introduced, namely the curvilinear coordinates $\boldsymbol{\xi}_\tau$, which closely approximates $(d_\tau, \tilde{t}_\tau)$, while having a nice polynomial relationship in terms of $\mathbf{x}$,

$$\boldsymbol{\xi}_\tau = \sum_{i=1}^{N_{\text{rec}}} \mathbf{b}_\tau^i \varphi_\tau^{+i}(\mathbf{x}), \tag{3.21}$$

where φ_τ^{+i} are the same principal coordinate monomials introduced in Equation (3.18). The coefficients $\mathbf{b}_\tau$ are subsequently found by requiring $\boldsymbol{\xi}_\tau$ to be as close as possible to $(d_\tau, \tilde{t}_\tau)$, at the reference point of all the control volumes in $Stencil(\tau)$. Mathematically, the following minimization problem must be solved:

$$\begin{aligned} &\underset{\mathbf{b}_\tau^1 \dots \mathbf{b}_\tau^{N_{\text{rec}}}}{\text{minimize}} && \sum_{\sigma \in Stencil(\tau)\cup\{\tau\}} (h(\mathbf{x}_\sigma) - \xi_{\tau 1}(\mathbf{x}_\sigma))^2 + (\tilde{t}(\mathbf{x}_\sigma) - \xi_{\tau 2}(\mathbf{x}_\sigma))^2 \\ &\text{subject to} && \boldsymbol{\xi}_\tau(\mathbf{x}_\tau) = 0, \end{aligned} \tag{3.22}$$

which has an identical solution process to that of Equation (2.9). Note that it is theoretically possible to use the Cartesian monomials $\phi_\tau^i(\mathbf{x})$ in the definition of the curvilinear coordinates in Equation (3.21), but this results in a poor-conditioned least square problem in finding the $\mathbf{b}_\tau^i$ coefficients, and thus is avoided. In this thesis, the emphasis is on anisotropic three-dimensional meshes that are symmetric in the x_3 direction. Taking advantage of the symmetry, we are able to define $\xi_{\tau 1}$ and $\xi_{\tau 2}$ using the

two-dimensional method, and simply set $\xi_{\tau 3} = x_3 - x_{\tau 3}$.

Now that the curvilinear coordinates are introduced, their corresponding basis functions, $\varphi_\tau^{*i}(\mathbf{x})$, can be defined as:

$$\left\{ \varphi_\tau^{*i}(\mathbf{x}) \,\middle|\, i = 1 \dots N_{\text{rec}} \right\} = \left\{ \frac{1}{a!b!c!} (\xi_{\tau 1}(\mathbf{x}))^a (\xi_{\tau 2}(\mathbf{x}))^b (\xi_{\tau 3}(\mathbf{x}))^c \,\middle|\, a + b + c \leq k \right\}. \tag{3.23}$$

Numerical experiments have shown that using these basis functions for near-wall high aspect ratio elements restores the k-exactness of the reconstruction scheme [32].

Chapter 4

Solving the Discretized System of Equations

This chapter begins by introducing the Pseudo Transient Continuation (PTC) method that finds the solution of Equation (2.4) via solving a series of systems of linear equations. Available methods for the solution of these linear systems are introduced, and their shortcoming in handling three-dimensional finite volume problems is addressed. Finally, a new solution scheme is proposed, and its performance is compared to other previously available methods in ANSLib.

4.1 Pseudo Transient Continuation

The PTC method [17, 32, 35, 47] starts from an initial guess for the steady-state solution of Equation (2.4). The initial guess is usually taken from the free-stream conditions or the converged solution of a lower-order accurate scheme. Then, the solution is updated iteratively until the norm of the residual vector, $\|\mathbf{R}(\mathbf{U}_h)\|_2$, falls below a desired threshold. Consider the backward Euler time advance scheme:

$$\frac{\mathbf{U}_h^+ - \mathbf{U}_h}{\Delta t} + \mathbf{R}(\mathbf{U}_h) = 0, \tag{4.1}$$

where Δt is the time step size, and $\mathbf{U}_h^+$ is the the DOF vector at the next time level. Linearizing Equation (4.1) gives:

$$\left(\frac{\boldsymbol{I}}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}_h}\right)\delta\mathbf{U}_h = -\mathbf{R}(\mathbf{U}_h), \tag{4.2}$$

where $\delta\mathbf{U}_h$ is the change in the DOF vector between the current and next time levels. Furthermore, $\frac{\partial \mathbf{R}}{\partial \mathbf{U}_h}$ is the residual Jacobian matrix, and is evaluated using the algorithm proposed by Michalak and Ollivier-Gooch [47].

The PTC method is derived from the backward Euler scheme by making a few modifications. First, a

nonuniform time step is used for every control volume. Thus, Equation (4.2) is changed to:

$$\left(\frac{\boldsymbol{V}}{\mathrm{CFL}}+\frac{\partial \mathbf{R}}{\partial \mathbf{U}_h}\right)\delta \mathbf{U}_h = -\mathbf{R}(\mathbf{U}_h), \tag{4.3}$$

where CFL is the Courant-Friedrichs-Lewy number, and the diagonal matrix $\boldsymbol{V}$ is the time step scaling matrix. The entries of $\boldsymbol{V}$ corresponding to control volume τ are denoted as V_τ, and found from the equation:

$$V_\tau = \frac{\lambda_{\max,\tau}}{h_\tau}, \tag{4.4}$$

where h_τ is the hydraulic diameter of the control volume, and $\lambda_{\max,\tau}$ is the maximum eigenvalue of the inviscid flux Jacobian over all the control volume surface quadrature points. Secondly, after $\delta \mathbf{U}_h$ is solved for, the solution is updated according to the line search algorithm:

$$\mathbf{U}_h \leftarrow \mathbf{U}_h + \omega \delta \mathbf{U}_h, \tag{4.5}$$

where $\omega \in (0 \quad 1]$ is the line search parameter, which must satisfy:

$$\left\| \frac{\omega \boldsymbol{V} \delta \mathbf{U}_h}{\mathrm{CFL}} + \mathbf{R}(\mathbf{U}_h + \omega \delta \mathbf{U}_h) \right\|_2 \leq \kappa \|\mathbf{R}(\mathbf{U}_h)\|_2. \tag{4.6}$$

Here, κ controls the strictness of the line search algorithm, and $\kappa = 1.2$ performs well for both viscous and inviscid flows [17]. In addition, the vector entries corresponding to the turbulence working variable are omitted in the evaluation of the norms in Equation (4.6) to enhance convergence [77]. Finally, the CFL number is updated at each iteration according to the value of ω:

$$\mathrm{CFL} \leftarrow \begin{cases} 1.5\,\mathrm{CFL} & \omega = 1 \\ \mathrm{CFL} & 0.1 < \omega < 1 \\ 0.1\,\mathrm{CFL} & \omega < 0.1 \end{cases}. \tag{4.7}$$

At the beginning of the solution process, the approximate solution is away from its steady-state value, and a small CFL number prevents divergence by making the approximate solution follow a physical transient path. On the other hand, when the approximate solution gets close to its steady-state value, Equation (4.7) will increase the CFL number. Therefore, the effect of the term $\frac{\boldsymbol{V}}{\mathrm{CFL}}$ in Equation (4.3) will be reduced, and Newton iterations would be recovered. Thus, as better solution approximations are obtained, the convergence rate will get closer to the optimum behavior of Newton's iterations.

4.2 Linear Solvers

Every PTC iteration requires the solution of the linear system $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, where:

$$\boldsymbol{A} = \left(\frac{\boldsymbol{V}}{\mathrm{CFL}} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}_h} \right) \quad \mathbf{x} = \delta \mathbf{U}_h \quad \mathbf{b} = -\mathbf{R}(\mathbf{U}_h). \tag{4.8}$$

The exact solution of the linear system is carried out by combinatorial algorithms that typically calculate a lower-upper (LU) factorization of the LHS matrix. Although there has been great progress in the development of such methods, e.g., MUMPS linear solver library [7], they still suffer from huge consumption of memory and time resources, when applied to sufficiently large problems. Furthermore, these methods do not take advantage of any initial guess for the solution, nor the fact that only an approximate solution is of interest. As a result, scientists in the CFD community have resorted to iterative methods for the solution of huge linear systems.

Iterative methods start from an initial guess $\mathbf{x}^{(0)}$, and then generate a sequence of improving approximate solutions. Iterative methods are either categorized as stationary, or a member of the Krylov Subspace (KSP) family. In a stationary method the approximate solution at step $k+1$, $\mathbf{x}^{(k+1)}$, is only dependent on the residual vector of the previous iteration, $\mathbf{r}^{(k)}$, which is defined as:

$$\mathbf{r}^{(k)} = \mathbf{b} - \boldsymbol{A}\mathbf{x}^{(k)}. \tag{4.9}$$

Conversely, a KSP method updates the approximate solution based on all or some of the previous residual vectors. ANSLib uses a KSP method, GMRES [63], as its linear solver because of its successful history in solving aerodynamic problems [17, 41, 49, 56, 78], and the fact that it is readily available as a black box solver in many numerical computation libraries, such as PETSc [10].

The GMRES method iteratively constructs an orthogonal basis for the search space spanned by the vectors $\mathbf{r}^{(0)}, \boldsymbol{A}\mathbf{r}^{(0)}, \ldots, \boldsymbol{A}^{k-1}\mathbf{r}^{(0)}$. Then, it seeks the next approximate solution vector, $\mathbf{x}^{(k)}$, as the member of the search space that minimizes $\|\mathbf{r}^{(k)}\|_2$. Furthermore, the search space is usually restarted after a certain number of iterations to prevent it from becoming too large. The behavior of the GMRES method is strongly dependent on the eigenvalue structure of matrix $\boldsymbol{A}$. The more compact the eigenvalue spectrum, the fewer iterations are required for convergence. As the LHS matrices arising from a k-exact finite volume discretization usually lack a nice eigenvalue distribution, the performance of GMRES must be improved by means of preconditioning, which is introduced in the following section.

4.3 Preconditioning

Preconditioning is the transformation of the original linear system into one which has the same solution, but is easier to solve using iterative methods. Preconditioning is usually done by first finding a matrix $\boldsymbol{P}$, such that the condition number of the product matrix $\boldsymbol{AP}$ is smaller than that of $\boldsymbol{A}$. Then,

the original equation is changed to:

$$\boldsymbol{AP}\mathbf{y} = \boldsymbol{b} \quad \mathbf{x} = \boldsymbol{P}\mathbf{y}. \tag{4.10}$$

The preconditioning matrix can be constructed either directly from $\boldsymbol{A}$, or the LHS matrix resulting from a lower-order discretization scheme. The matrix from which the preconditioner is constructed will be denoted as $\boldsymbol{A}^*$.

The main steps of the preconditioned GMRES algorithm are shown in Algorithm 1 [63], where no restart strategy and a zero initial guess have been assumed for simplicity. In this algorithm, the columns of the $\boldsymbol{V}_m \in \mathbb{R}^{N_{\text{unk}} \times m}$ matrix form an orthonormal basis for the search space spanned by the vectors $\mathbf{b}$, $(\boldsymbol{AP})\mathbf{b}$, …, $(\boldsymbol{AP})^{m-1}\mathbf{b}$. At each iterations m, the $\boldsymbol{H}_m$ and $\boldsymbol{V}_m$ matrices are used to find the solution guess, $\mathbf{x}^{(m)} = \boldsymbol{AP}\boldsymbol{V}_m\boldsymbol{\alpha}_m$, and the norm of its corresponding residual, $\|\mathbf{r}^{(m)}\|_2 = \gamma_m$. Also, n is the maximum number of iterations while *rtol* is the tolerance in the reduction of the residual norm. A smaller tolerance value will result in a more accurate solution, but requires more GMRES iterations. Finally, note that the GMRES solver only requires the matrix-by-vector product of the $\boldsymbol{P}$ matrix, and does not need its explicit form.

In this section, the preconditioners available in ANSLib, namely Point Gauss-Seidel, Block Jacobi, and ILU will be introduced, and their shortcoming in solving large three-dimensional problems will be discussed. More complicated preconditioners, such as multigrid and domain decomposition have not been considered in this thesis because of the inconsistent results available in the literature regarding their performance. For example, Shahbazi et al. [65] reported that the linear multigrid method can be ten times faster than conventional single grid algorithms. On the other hand, Diosady and Darmofal [20], and Wallraff et al. [77] reported cases for which they were not able to achieve a significant speedup factor.

4.3.1 Point Gauss-Seidel

The Point Gauss-Seidel (PGS) method is a stationary iterative solver that can also be used as a preconditioner for the GMRES method. First, $\boldsymbol{A}^*$ is decomposed as:

$$\boldsymbol{A}^* = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}, \tag{4.11}$$

where $\boldsymbol{D}$ is the control volume diagonal part of $\boldsymbol{A}$, $\boldsymbol{L}$ is the strict lower part, and $\boldsymbol{U}$ is the strict upper part. Then, the matrix-by-vector product $\boldsymbol{P}\mathbf{v}$ of the PGS preconditioner is defined as:

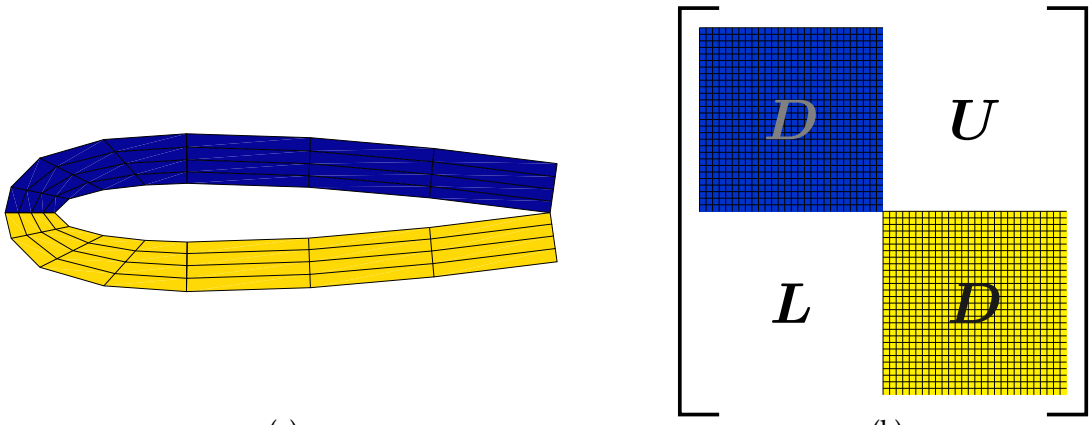
$$\begin{aligned}
&\mathbf{z}^{(0)} = 0 \\
&\mathbf{z}^{(1)} = (\boldsymbol{D} + \boldsymbol{L})^{-1}(\mathbf{v} - \boldsymbol{A}^*\mathbf{z}^{(0)}) \\
&\vdots \\
&\mathbf{z}^{(n)} = (\boldsymbol{D} + \boldsymbol{L})^{-1}(\mathbf{v} - \boldsymbol{A}^*\mathbf{z}^{(n-1)}) \\
&\boldsymbol{P}\mathbf{v} = \mathbf{z}^{(n)},
\end{aligned} \tag{4.12}$$

Algorithm 1 Preconditioned GMRES with no restart and a zero initial guess.

function GMRES($\boldsymbol{A}, \boldsymbol{P}, \mathbf{b}, n, rtol$)
  $\beta = \|\mathbf{b}\|_2$
  $\mathbf{v}_1 = \mathbf{b}/\beta$
  $\boldsymbol{V}_1 = [\mathbf{v}_1]$, $\boldsymbol{H}_0 = []$
  $m = 0$
  repeat
    $m = m + 1$
    $\mathbf{z}_m = \boldsymbol{P}\mathbf{v}_m$
    $\mathbf{w} = \boldsymbol{A}\mathbf{z}_m$
    $\mathbf{h}_m, \mathbf{v}_{m+1}$ = ARNOLDI($\boldsymbol{V}_m, \mathbf{w}$)
    $\boldsymbol{V}_{m+1} = [\boldsymbol{V}_m, \mathbf{v}_{m+1}]$, $\boldsymbol{H}_m = [\boldsymbol{H}_{m-1}, \mathbf{h}_m]$
    $\boldsymbol{\alpha}_m, \gamma_m$ =DENSESOLVE($\beta, \boldsymbol{H}_m$)
  until $m > n$ or $\gamma_m/\beta < rtol$
  Return $\mathbf{x}^{(m)} = \boldsymbol{P}\boldsymbol{V}_m\boldsymbol{\alpha}_m$

function ARNOLDI($\boldsymbol{V}_m, \mathbf{w}$)
  Returns the vector $\mathbf{v}_{m+1}$, which is the unit normal component of $\mathbf{w}$ to the space spanned by the previous $\mathbf{v}_i$ ($i \leq m$) vectors.
  Returns the column vector $\mathbf{h}_m$, where $(h_m)_i$ is the length of the projection of $\mathbf{w}$ over the $\mathbf{v}_i$ vector if $i \leq m + 1$, and is zero otherwise.

function DENSESOLVE($\beta, \boldsymbol{H}_m$)
  Returns $\boldsymbol{\alpha}_m = \text{argmin}_{\boldsymbol{\alpha}} \|\mathbf{b} - \boldsymbol{A}\boldsymbol{P}\boldsymbol{V}_m\boldsymbol{\alpha}\|$ and $\gamma_m = \min_{\boldsymbol{\alpha}} \|\mathbf{b} - \boldsymbol{A}\boldsymbol{P}\boldsymbol{V}_m\boldsymbol{\alpha}\|$.
  The objective of this function is achieved by solving the equivalent problem $\text{argmin}_{\boldsymbol{\alpha}} \left\|\beta\mathbf{e}_1 - \boldsymbol{H}_m\boldsymbol{\alpha}\right\|_2$ using a rotational change of variables. Solving this problem includes the solution of a dense $m \times m$ linear system, and yields both γ_m and $\boldsymbol{\alpha}_m$. Also, $\mathbf{e}_1 = [1, 0, \ldots, 0]^T$.



(a) (b)

Figure 4.1: Schematic of Block Jacobi decomposition: (a) The mesh, and the partition belonging to each processor (b) The lower, upper, and block diagonal matrices.

where n is the number of PGS iterations. Although PGS is a strong preconditioner for solving linear systems arising from discretization on isotropic meshes, it performs poorly when dealing with anisotropic meshes [44]. Therefore, PGS is only used for the solution of Poisson's and Euler equations in this thesis, for which an isotropic mesh correctly captures the solution.

4.3.2 Block Jacobi

The Block Jacobi (BJ) preconditioner also uses a decomposition of matrix $\boldsymbol{A}^*$, with the difference that the diagonal blocks of matrix $\boldsymbol{D}$ must be composed of all the degrees of freedom belonging to the same processor. Figure 4.1a shows a partitioned mesh, and Figure 4.1b shows the corresponding structure and decomposition of the LHS matrix. The matrix-by-vector product for this preconditioner is denoted as:

$$\begin{aligned}
&\mathbf{z}^{(0)} = 0 \\
&\boldsymbol{D}\mathbf{z}^{(1)} = \mathbf{v} - \boldsymbol{A}^*\mathbf{z}^{(0)} \\
&\vdots \\
&\boldsymbol{D}\mathbf{z}^{(n)} = \mathbf{v} - \boldsymbol{A}^*\mathbf{z}^{(n-1)} \\
&\boldsymbol{P}\mathbf{v} = \mathbf{z}^{(n)},
\end{aligned} \tag{4.13}$$

where n is the number of BJ iterations. Note that the block diagonal matrix $\boldsymbol{D}$ cannot be inverted exactly anymore, and each intermediate vector $\mathbf{z}^{(k)}$ must be solved for by using an inner iterative solver. Nevertheless, since every block of $\boldsymbol{D}$ only belongs to a single processor, the inner iterative solver can be serial. Thus, BJ is a means of parallelizing serial preconditioners, and is used for this purpose in ANSLib.

4.3.3 ILU

The incomplete lower-upper factorization with fill level p (ILUp) produces an approximation of the LU factorization of $\boldsymbol{A}^*$, such that the approximate matrices have a much smaller nonzero structure than the exact LU factorization. The factored lower and upper triangular matrices $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ are computed by performing Gaussian elimination on $\boldsymbol{A}^*$, but ignoring certain matrix entries. Then, the preconditioner matrix is set to $\boldsymbol{P} = (\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}})^{-1}$. A larger fill level results in $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices with bigger nonzero structure, but is likely to construct a more effective preconditioner [63]. Nevertheless, the ILU method is inherently a serial algorithm, so it is implemented in conjunction with the BJ method for parallel simulations.

Since the ILU preconditioner is based on matrix factoring, its performance is dramatically affected by the ordering of DOFs in the matrix $\boldsymbol{A}^*$. Quotient minimum degree (QMD) and reverse Cuthill-McKee (RCM) are among the reordering algorithms that are offered in PETSc [10], and are used in ANSLib. The former reduces the fill of the factored matrices, while the latter reduces the fill of matrix $\boldsymbol{A}^*$ itself.

In the course of development of ANSLib, Nejat and Ollivier-Gooch [49] used an ILU preconditioner factored from the LHS matrix of the 0-exact scheme to solve the Euler equations. They showed the good performance of their preconditioning method for 1- and 2-exact finite volume schemes, but concluded that the 3-exact scheme requires a more effective preconditioner. Later, Michalak and Ollivier-Gooch [47] demonstrated that incomplete factorization of the higher-order LHS matrix results in faster convergence compared to factoring the 0-exact LHS matrix for inviscid problems. They further conjectured that factoring the LHS matrix to the full order using a fill level of 3 has a feasible memory consumption, and can be a practical preconditioner for three-dimensional flow problems. Hereinafter, the ILU preconditioner of fill level p factored from the 0-exact and full-order LHS matrices will be denoted as LO-ILUp and HO-ILUp, respectively.

Viscous flow problems are more challenging than their inviscid counterparts. Jalali and Ollivier-Gooch [32] observed that a fill level of three or larger is required for HO-ILU preconditioning of viscous turbulent flow problems. The HO-ILU3 method is a practical preconditioner for two-dimensional problems, but its memory cost soars drastically in three-dimensions, hindering its implementation for such problems. To mitigate this issue, a new ILU based algorithm will be introduced in the next section that not only has a less memory consumption than HO-ILU3, but also does not suffer from poor performance of LO-ILU when used with the 3-exact reconstruction scheme.

4.4 Improved Preconditioning Algorithms

In this section, an effective preconditioning algorithm is proposed based on inner GMRES iterations. Furthermore, the lines of strong unknown coupling are presented for reordering of the LHS matrix. As will be shown in Section 4.5, this reordering method can improve the speed of the solver consider-

ably.

4.4.1 Inner GMRES Iterations

Some researchers have observed that the lower-order LHS matrix can construct a more effective preconditioner compared to its higher-order counterpart because the structure of the lower-order LHS matrix only includes the immediate neighbors of every control volume [41, 56, 78]. Nevertheless, as will be shown in Section 4.5, the LO-ILU method can behave poorly for high-order reconstruction schemes applied to viscous flow problems. Our conjecture is that using the exact inverse of the lower-order LHS matrix as the preconditioner: $\boldsymbol{P} = (\boldsymbol{A}^*)^{-1}$, rather than the product of the ILU matrices: $\boldsymbol{P} = (\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}})^{-1}$, can mitigate the mentioned issue. In this case, the matrix-by-vector product $\mathbf{z} = \boldsymbol{P}\mathbf{v}$ can be found by solving the system:

$$\boldsymbol{A}^*\mathbf{z} = \mathbf{v}. \tag{4.14}$$

Nevertheless, the linear system of Equation (4.14) can be as large as the original linear system $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, and its solution cannot be carried out exactly. Instead of seeking the exact solution, the proposal is to use further inner ILU preconditioned GMRES iterations to find an approximate solution for Equation (4.14). The resulting preconditioner will be referred to as GMRES-LO-ILU in this thesis, and its performance will be examined in Section 4.5.

When an inner GMRES solver is used as the preconditioner, $\boldsymbol{P}$ will not be a linear operator anymore, and its effect changes from iteration to iteration. Therefore, the equation $\mathbf{x}^{(m)} = \boldsymbol{P}\boldsymbol{V}_m\boldsymbol{\alpha}_m$ cannot be used any longer. The solution to this issue is using a Flexible GMRES (FGMRES) outer solver [62]. The resulting combination, i.e., a GMRES-LO-ILU preconditioned FGMRES solver, is shown in Algorithm 2. Unlike normal GMRES, the $\mathbf{z}$ vectors are found by applying the inner GMRES solver to the $\mathbf{v}$ vectors, and are explicitly kept track of as the columns of the $\boldsymbol{Z}$ matrix. Moreover, the final approximate solution is written as $\mathbf{x}^{(m)} = \boldsymbol{Z}_m\boldsymbol{\alpha}_m$ while the ILU factored matrices are still supplied to the algorithm for preconditioning the inner GMRES solver. Also, tolerance values and limits for the maximum number of iterations have to be specified for both the inner and outer solvers which are identified by the subscripts $(.)_i$ and $(.)_o$, respectively.

4.4.2 Lines of Strong Coupling Between Unknowns

Forming non-overlapping lines that contain control volumes with strongly coupled degrees of freedom is beneficial in constructing effective preconditioners. Mavriplis [43] introduced the concept of lines in anisotropic unstructured meshes to enhance the performance of multigrid solvers via implicit smoothing. His approximate algorithm for finding such lines was purely geometric-based, and only considered control volume aspect ratios. Thus, his method only captured coupling through diffusion. Okusanya et al. [52] later developed an algorithm that considered both the advection and diffusion phenomena. Fidkowski et al. [21] proved that such an algorithm results in a set of unique lines, and used

Algorithm 2 GMRES-LO-ILU preconditioned FGMRES with no restart and a zero initial guess.

function FGMRES($\boldsymbol{A}, \boldsymbol{A}^*, \tilde{\boldsymbol{L}}, \tilde{\boldsymbol{U}}, \mathbf{b}, n_o, rtol_o, n_i, rtol_i$)
  $\beta = \|\mathbf{b}\|_2$
  $\mathbf{v}_1 = \mathbf{b}/\beta$
  $\boldsymbol{V}_1 = [\mathbf{v}_1]$, $\boldsymbol{H}_0 = []$, $\boldsymbol{Z}_0 = []$,
  $m = 0$
  repeat
    $m = m + 1$
    $\mathbf{z}_m$ =GMRES($\boldsymbol{A}^*, (\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}})^{-1}, \mathbf{v}_m, n_i, rtol_i$)
    $\mathbf{w} = \boldsymbol{A}\mathbf{z}_m$
    $\mathbf{h}_m, \mathbf{v}_{m+1}$ = Arnoldi($\boldsymbol{V}_m, \mathbf{w}$)
    $\boldsymbol{V}_{m+1} = [\boldsymbol{V}_m, \mathbf{v}_{m+1}]$, $\boldsymbol{H}_m = [\boldsymbol{H}_{m-1}, \mathbf{h}_m]$, $\boldsymbol{Z}_m = [\boldsymbol{Z}_{m-1}, \mathbf{z}_m]$
    $\boldsymbol{\alpha}_m, \gamma_m$ =DenseSolve($\beta, \boldsymbol{H}_m$)
  until $m > n$ or $\gamma_m/\beta < rtol$
  Return $\mathbf{x}^{(m)} = \boldsymbol{Z}_m\boldsymbol{\alpha}_m$

the lines for nonlinear p-multigrid solution of Euler equations. Diosady and Darmofal [20] showed that reordering the unknowns, such that members of a line have consecutive numbers, is an effective reordering strategy for the ILU preconditioner. In this thesis, the line reordering algorithm of [20] is implemented to improve the speed of the linear solver.

In the first step of the line creation algorithm, a weight is assigned to every face inside the mesh. This weight is derived from the Jacobian of the 0-exact discretization of the advection-diffusion equation,

$$\nabla \cdot (\mathbf{v}u - \mu_L \nabla u) = 0, \tag{4.15}$$

where u is the unknown variable, $\mathbf{v}$ is the velocity taken from the initial conditions, and μ_L is an input parameter that controls the sensitivity of the lines to mesh anisotropy. The weight of a face f is then defined as:

$$W(f) = \max\left(\frac{\partial R_\sigma}{\partial u_\tau}, \frac{\partial R_\tau}{\partial u_\sigma}\right), \tag{4.16}$$

where W is the weight of the face, σ and τ are the adjacent control volumes of the face, and $\mathbf{R}$ the residual vector. If a face is located on the boundary, Equation (4.16) will not directly be applicable to it. Thus, a mirrored ghost control volume is placed adjacent to every boundary face. After finding the face weights, calling Algorithm 3 will construct the lines, where $F(\tau)$ is the set of faces of control volume τ, and $\sigma = C(\tau, f)$ is the control volume which has the face f in common with control volume τ. This algorithm ensures that two adjacent control volumes are connected to each other if and only if they have their first or second highly weighted face lying between them [21].

To illustrate the performance of the algorithm, an anisotropic mesh for the geometry of a NACA 0012 airfoil is considered, as shown in Figure 4.2. The velocity has been taken from the 2-exact solution of the flow resulting from a Mach number of 0.15, a Reynolds number of 6×10^6, and an angle of attack of $10°$. The parameter μ_L is set to the heuristic value of 10^{-5}. As desired, the lines follow the direction

Algorithm 3 Line creation

function CreateLines ▷ Calling this function creates all lines.
  Unmark all control volumes.
  while there a exists an unmarked control volume τ **do**
    MakePath(τ) ▷ Forward path creation
    MakePath(τ) ▷ Backward path creation

function MakePath(τ) ▷ Helper function.
  repeat
    For control volume τ, pick the face $f \in F(\tau)$ with the highest weight, such that control volume $\sigma = C(\tau, f)$ is not part of the current line.
    if f is a boundary face **then**
      Terminate
    else if σ is marked **then**
      Terminate
    else if f is not the first or second ranked face of σ in weight **then**
      Terminate
    Mark σ.
    Add σ to the current line.
    $\tau \leftarrow \sigma$
  until Termination

of mesh anisotropy near wall boundaries, while their pattern changes, and follows the flow direction in other parts of the geometry.

4.5 Numerical Comparisons

In this section, the performance of the proposed preconditioning algorithm is studied by considering the turbulent flow around a NACA 0012 airfoil, with $\alpha = 10°$, $Ma = 0.15$, and $Re = 6 \times 10^6$. Three nested meshes with approximately 25K, 100K, and 400K control volumes (N_{CV} = 25K,100K, and 400K) are considered, where the coarsest mesh and its corresponding lines of strong unknown coupling are shown in Figure 4.2. The farfield boundaries are located almost 500 chords away from the airfoil, and the chord length has a nondimensional value of one. This problem is taken from the NASA Turbulence Modeling Resource (TMR) website [60], and has been previously studied by Jalali and Ollivier-Gooch [32], where they used a HO-ILU3 preconditioner with QMD reordering. Only the highest-order discretization scheme of ANSLib (3-exact) is considered here, since it results in the stiffest linear systems that are most difficult to solve. Also, the initial conditions are taken from the steady-state solution of the lower-order 2-exact scheme.

The ILU based preconditioning methods tested are shown in Table 4.1. In all cases, the outer GMRES solver stops when the linear residual of Equation (4.3) is reduced by a factor of 10^{-3}, or a maximum number of 500 iterations is performed. The maximum Krylov subspace size is 100 for the outer GMRES

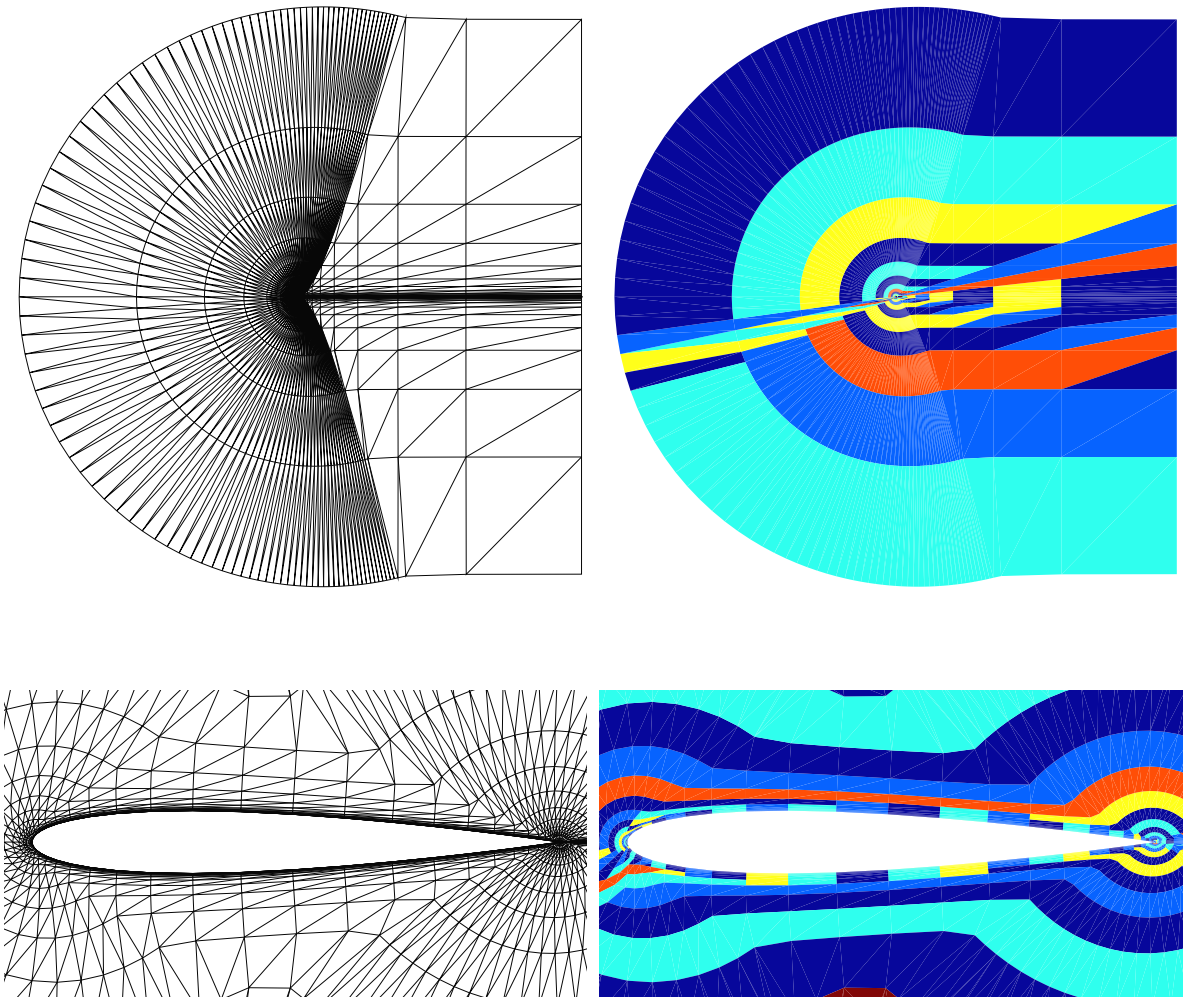


Figure 4.2: Mesh and lines of strong unknown coupling for the geometry of a NACA 0012 airfoil

Table 4.1: Preconditioning methods considered

Case name	Preconditioning method	Reordering algorithm	$rtol_i, n_i$
A	HO-ILU3	QMD	—
B	LO-ILU0	RCM	—
C	LO-ILU0	lines	—
D	GMRES-LO-ILU0	RCM	$10^{-3}, 10$
E	GMRES-LO-ILU0	lines	$10^{-3}, 10$
E*	GMRES-LO-ILU0	lines	$10^{-3}, 60$

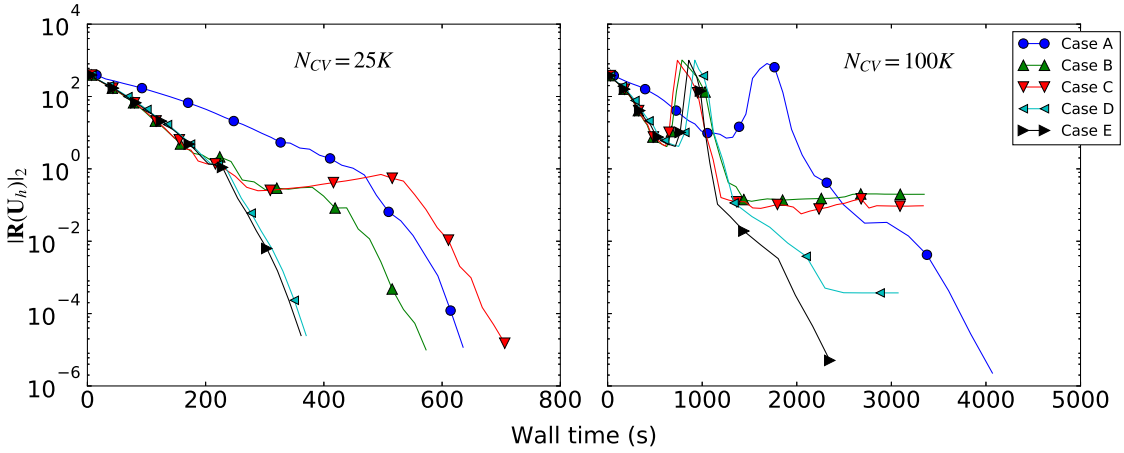


Figure 4.3: Comparison of residual histories for the NACA 0012 problem obtained using different preconditioning algorithms

solver, and the initial CFL number is set to 10^{-2}. Furthermore, the simulation ends when the norm of the residual vector $\|\mathbf{R}(\mathbf{U}_h)\|_2$ is reduced by a factor of 10^{-8}. The command-line options supplied to the ANSLib executable for each case are listed in Appendix A. Cases A-E are compared to each other on the 25K and 100K meshes, while case E* is used to solve the problem on the finest mesh. Note that other combinations of preconditioner and reordering schemes that are not considered in Table 4.1, did not result in convergence even for the coarse mesh. Moreover, an extensive search for finding the optimum number of inner iterations n_i has not been carried out, and the large number of inner iterations for case E* is chosen as a safe measure to ensure convergence on the finest mesh.

The residual history of the solver for each preconditioning algorithm is shown in Figure 4.3, where the wall time is obtained from a single core of an Intel i7-4790 (3.60 GHz) CPU. Our proposed preconditioning algorithm (case E) outperforms all the other methods on both mesh sizes, and takes only half the time of the HO-ILU3 algorithm (case A) to find the steady-state solution. Furthermore, the line reordering algorithm shows its prominence on the 100K control volume mesh, where the RCM reordering algorithm fails. The effectiveness of the new preconditioning algorithm is further demonstrated by solving the problem on a 400K control volume mesh. The residual history for this case is shown in Figure 4.4.

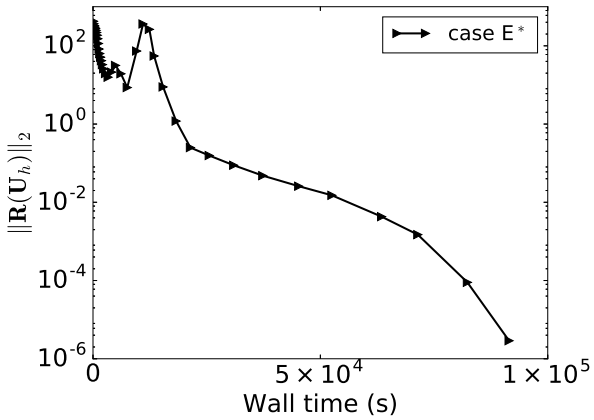


Figure 4.4: Residual history for the NACA 0012 problem on the finest mesh of 400K control volumes

Table 4.2 shows the number of PTC iterations (N-PTC), the number of outer GMRES iterations (N-GMRES), total memory consumption, CPU time spent on the linear solver (LST), and total computational time (TST). The proposed preconditioning scheme (case E) speeds up the linear solve process by a factor of three, and results in a twofold reduction in memory consumption, compared to the HO-ILU3 method. The shortcomings of the LO-ILU methods (cases B and C) are evident because of their large number of PTC iterations and failure in convergence, on the coarse and medium meshes, respectively. Also, the memory consumption scales linearly with respect to the problem size for the proposed preconditioning scheme, whereas the linear solve time seems to be increasing at a much more rapid rate. Although the proposed preconditioning scheme has a huge linear solve time for $N_{\text{CV}} = 400\text{K}$, it still outperforms the HO-ILU3 method, which cannot handle this mesh size at all.

The viscous drag (C_{Dv}), the pressure drag (C_{Dp}), and the lift (C_L) coefficients computed from the numerical solution along with their reference values are listed in Table 4.3. The reference values are taken from the NASA TMR website, and are obtained by the FUN3D [2] solver running on a very fine mesh (approximately 2M quadrilaterals). As expected, the change in the coefficients between the fine and medium meshes is much smaller than that of the medium and coarse meshes. Furthermore, the gap between the coefficients and their reference values gets smaller with mesh refinement, with the exception of C_L. Solution singularities, different boundary condition implementations between ANSLib and FUN3D, or inexact evaluation of the distance from wall function may partly be accountable for the non-ideal behavior of the lift coefficient. Nevertheless, the purpose of this section was the solution of the discretized system of equations, which according to Figure 4.4 has been performed correctly.

Table 4.2: Performance comparison for different preconditioning schemes. Cases with entries marked by "–" did not converge to the steady state solution.

Preconditioner	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
N_{CV} = 25K					
A	37	956	1.4	416	635
B	44	10,893	0.7	264	572
C	51	13,692	0.7	328	705
D	34	1,856	0.7	134	379
E	34	1,787	0.7	118	361
N_{CV} = 100K					
A	39	4,640	5.9	3,122	4,068
B	–	–	2.8	–	–
C	–	–	2.8	–	–
D	–	–	3.2	–	–
E	36	4,396	3.2	1,308	2,348
N_{CV} = 400K					
E*	38	6,869	13.1	87,312	91,502

Table 4.3: Computed drag and lift coefficients for the NACA 0012 airfoil

N_{CV}	C_{Dp}	C_{Dv}	C_L
25K	0.00545	0.00583	1.0951
100K	0.00615	0.00623	1.0905
400K	0.00609	0.00619	1.0922
NASA TMR	0.00607	0.00621	1.0910

Chapter 5

Three-Dimensional Results

This chapter presents the k-exact finite volume solution of four three-dimensional problems. The problems are studied in order of difficulty, and the solutions are verified to assess the accuracy and performance of the solver. First, Poisson's equation is solved in a cubic domain, where the source term is obtained from a manufactured solution [64]. Then, the Euler equations are solved to simulate the subsonic inviscid flow around a unit sphere. Subsequently, the solution of Navier-Stokes + negative S-A equations are presented to simulate the subsonic viscous turbulent flow over a flat plate and the extruded NACA 0012 airfoil. All the command-line options provided to the solver for running each case are provided in Appendix A.

All the flow simulations were performed using the Grex cluster from the WestGrid computing facilities [3]. Each node of the cluster consists of two 6-core Intel Xeon X5650 2.66GHz processors, and has a minimum of 48GB memory. All compute nodes are connected by a non-blocking Infiniband 4X QDR network. The wall time and memory cost of all the flow problems are reported using 8 processors. In addition, a strong parallel scaling test is conducted for the flat plate and the sphere test cases, i.e., the same problem is solved by using different numbers of processors ranging from 1 to 10 while the resulting speedup is recorded.

5.1 Poisson's Equation in a Cubic Domain

For this test case, Poisson's equation is considered inside the domain $\Omega = [0,1]^3$. The manufactured exact solution

$$u = \sinh(\sin(x_1))\sinh(\sin(x_2))\sinh(\sin(x_3)) \tag{5.1}$$

is used to find the source term, while homogeneous Dirichlet conditions are imposed over all the boundaries. Slice plots of the exact solution are shown in Figure 5.1. The solution has a maximum value of $(\sinh(1))^3$ at the point $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, and decays to zero on the boundaries.

A study of the solution accuracy is performed on four families of unstructured grids. The first grid

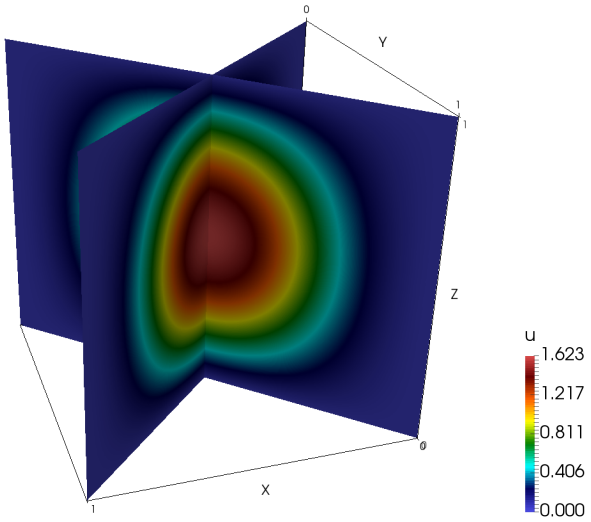


Figure 5.1: Exact solution of Poisson's problem

family is only composed of hexahedra while the other families are composed of only prisms, mixture of pyramids and tetrahedra, and only tetrahedra, respectively. The hexahedral grids are constructed by perturbing the vertices of a uniform $N \times N \times N$ structured mesh. Other grid families are subsequently created by decomposing each hexahedron into the desired element types. Figure 5.2 shows all the grids for $N = 5$.

The problem is solved on grid sizes: $N = 10, 20, 40$, and reconstruction orders: $k = 1, 2, 3$. Figure 5.3 shows the L_2 norm of the discretization error $\|e_h\|_2$ as a function of the mesh length scale $h = \frac{1}{N}$ and the number of degrees of freedom. Optimal accuracy order of $k + 1$ is attained for the $k = 1$ and 3 reconstruction schemes as expected, whereas the $k = 2$ scheme only achieves a second-order accurate solution. Nevertheless, the non-optimal behavior of $k = 2$ is expected for Poisson's problem, and was previously addressed and theoretically explained by Ollivier-Gooch and Van Altena [54].

5.2 Inviscid Flow Around a Sphere

The second test problem is the inviscid flow around a sphere with unit radius, and centered at the point $(0, 0, 0)$. The Mach number is equal to $Ma = 0.38$, and the free-stream flow is in the x_1 direction. This problem is chosen as a three-dimensional extension of the inviscid flow around a circle, the higher-order solution of which was studied by Bassi and Rebay [14]. Three prismatic grids (64K, 322K, and 1M control volumes) are used, where the farfield is located at a distance of 100 dimensionless units from the sphere surface. A symmetric cut of the coarsest mesh near the sphere surface is shown in Figure 5.4. The faces and control volumes adjacent to the wall boundaries are curved using only second-order Lagrange polynomials, as quadratic functions are sufficient to represent a spherical surface. To ensure that numerical quadrature does not introduce additional error in the solution, a quadrature scheme accurate up to order $k + 2$ is employed.

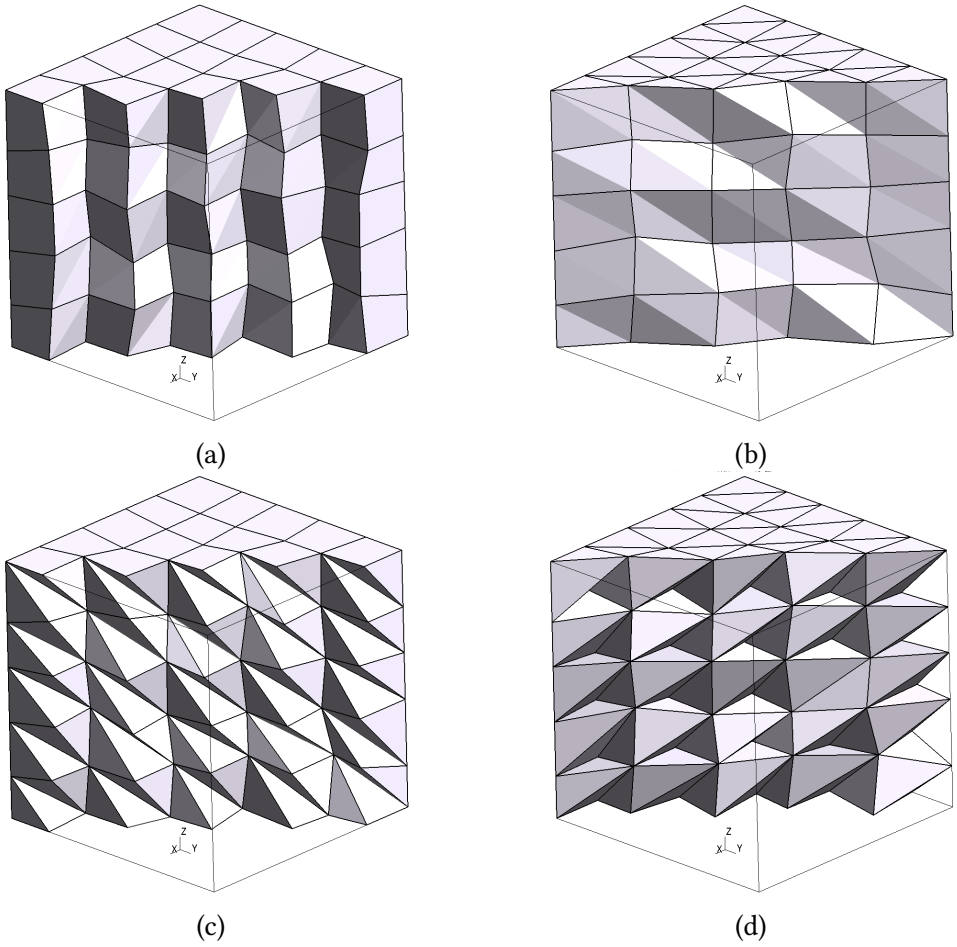


Figure 5.2: Different meshes for the solution of Poisson's problem: (a) hexahedra, (b) prisms, (c) mixture of pyramids and tetrahedra, (d) tetrahedra.

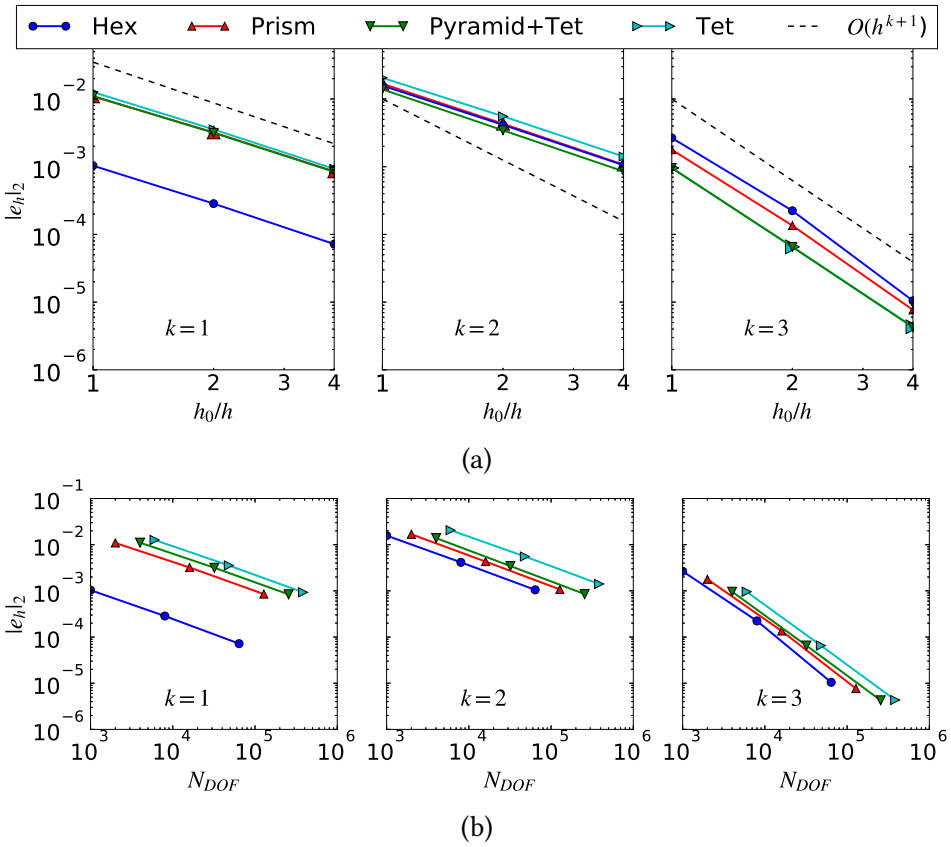


Figure 5.3: Discretization error versus mesh size for Poisson's problem: (a) Error versus mesh length scale, (b) Error versus number of degrees of freedom.

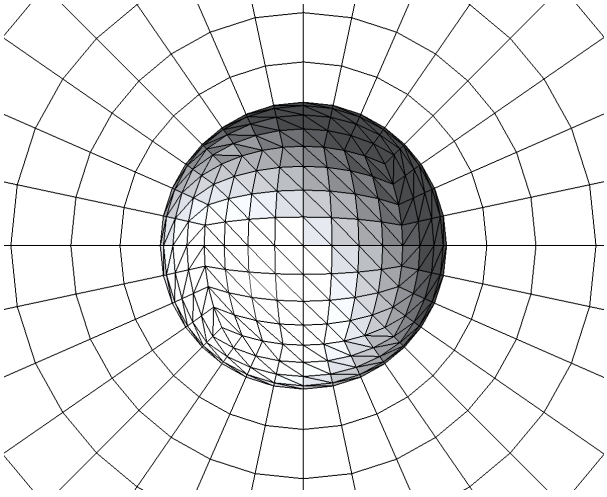


Figure 5.4: Symmetric cut of the coarsest mesh near the sphere surface

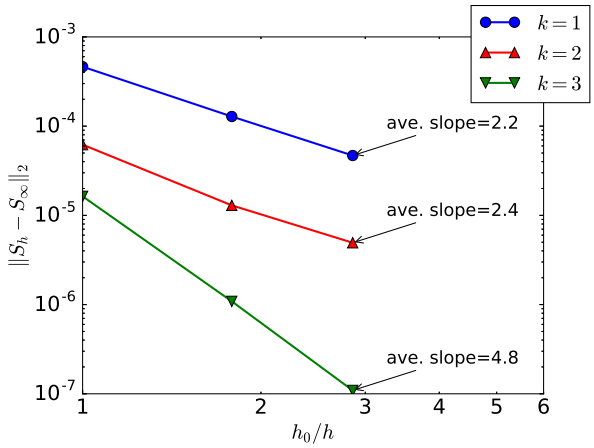


Figure 5.5: Relative entropy norm versus mesh size for the sphere problem

In all cases, the initial solution is the uniform free-stream condition everywhere, and the initial CFL number is set to 10. The GMRES linear solver stops when the linear residual of Equation (4.3) is 10^{-3} times its initial value, or a maximum of 500 iterations is performed. Furthermore, the GMRES solver has a maximum Krylov subspace size of 100, and is preconditioned using the BJ preconditioner with $n_o = 4$ iterations. The inner solver for the BJ preconditioner is PGS with $n_i = 4$ iterations. Although the LHS matrix of the linear system is calculated to full-order, its preconditioner is constructed from the LHS matrix of the 0-exact reconstruction scheme.

As the flow does not contain any discontinuity, the entropy must be constant throughout the domain. Therefore, the L_2 norm of the entropy relative to the free-stream conditions, $\|S - S_\infty\|_2$, has an exact value of zero, and is used to study the solution accuracy. The convergence of $\|S - S_\infty\|_2$ with mesh refinement is shown in Figure 5.5, where the mesh length scale is defined as $h = (N_{\text{CV}})^{-1/3}$. The $k = 1$ and 3 reconstruction schemes converge even faster than their expected ratios of 2 and 4, respectively. The convergence ratio for the $k = 2$ reconstruction scheme, however, is half an order smaller than its nominal value. The slight deviation of the convergence orders from their theoretical values can partly be attributed to the fact that the meshes employed are not nested.

The effect of the reconstruction order k on solution accuracy is also evident from the qualitative Mach contours on the $x_3 = 0$ symmetry plane, shown in Figure 5.6. Not surprisingly, only the $k = 3$ reconstruction scheme has been able to capture the symmetry of the flow on the finest mesh. Furthermore, the artificial wake behind the sphere seems smaller for the $k = 2$ solution compared to that of $k = 1$.

The norm of the residual vector per PTC iteration is shown in Figure 5.7. Convergence is independent of the reconstruction order k, but the number of iterations increases as the mesh is refined. It is noteworthy to mention that the increase in the initial norm corresponding to mesh refinement is due to the definition of the residual vector $\mathbf{R}(\mathbf{U}_h)$ in Equations (2.3) and (2.4). Each entry of the residual vector is divided by the size of its corresponding control volume. Therefore, when the mesh is refined,

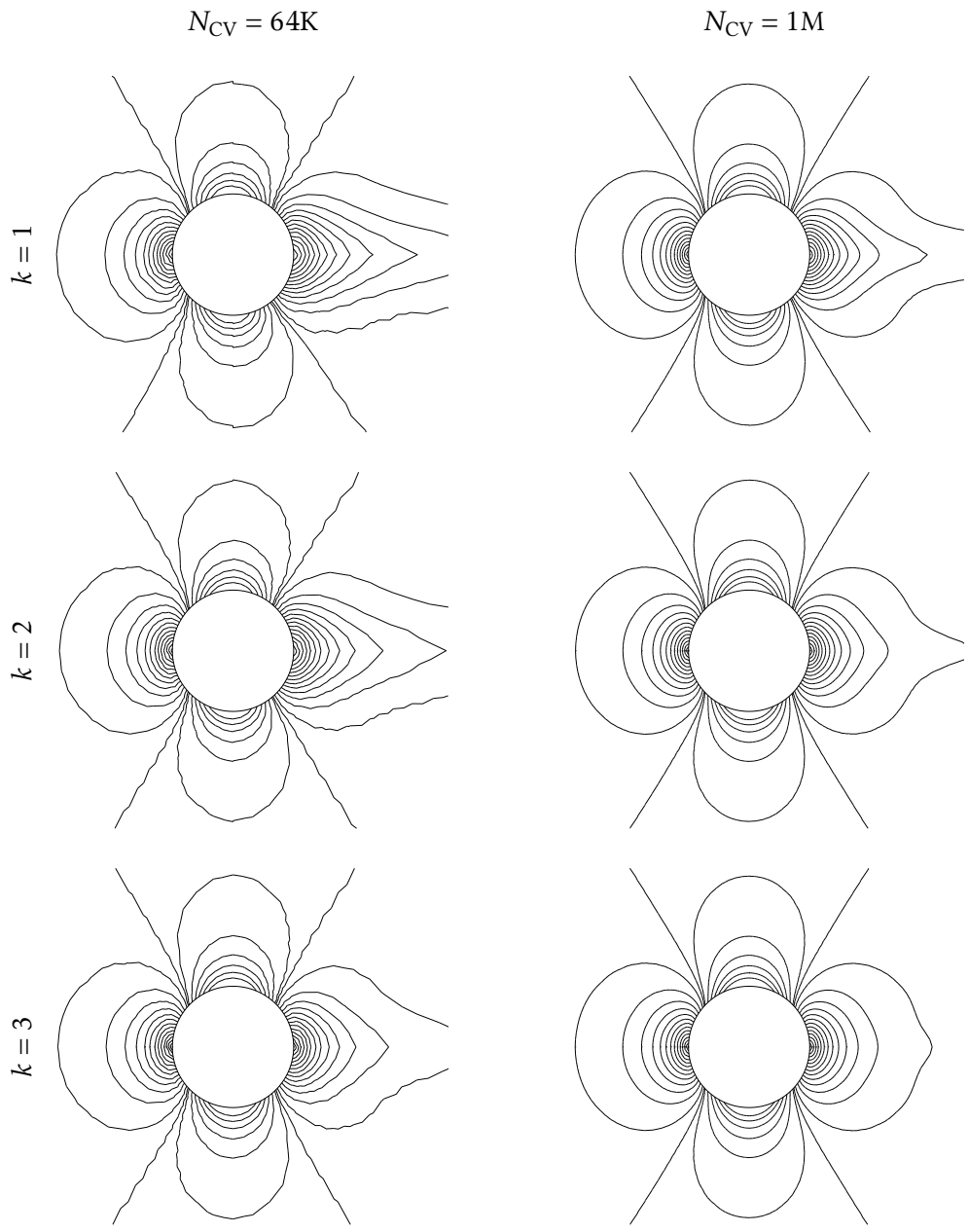


Figure 5.6: Computed Mach contours on the $x_3 = 0$ symmetry plane for the sphere problem

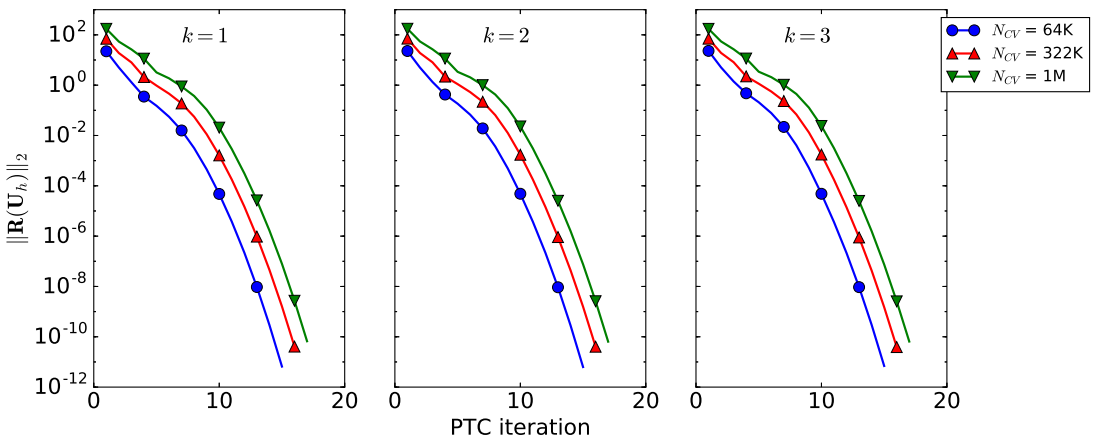


Figure 5.7: Norm of the residual vector per PTC iteration for the sphere problem

Table 5.1: Number of iterations and the resource consumption of the flow solver for the sphere problem

k	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
			N_{CV} = 64K		
1	15	182	2.94	24	107
2	15	181	3.91	20	112
3	15	255	6.00	31	320
			N_{CV} = 322K		
1	16	207	13.24	134	579
2	16	212	14.05	132	620
3	16	300	28.39	271	1,879
			N_{CV} = 1M		
1	17	275	39.48	486	1,969
2	17	277	45.52	536	2,150
3	17	385	85.78	793	5,904

the control volume sizes become smaller, and the residual vector entries will increase.

The number of iterations and the resource consumption of the flow solver are listed in Table 5.1. The time spent on linear solves makes up less than twenty percent of the computational time, suggesting that the Jacobian evaluation is the major computational bottleneck for this problem. Either increasing the reconstruction order, or the number of control volumes results in stiffer linear systems, which require more GMRES iterations to be solved. Nevertheless, both the solution time and memory consumption increase linearly with the number of control volumes for all k. Also, the advantage of using a high-order reconstruction scheme can be clearly seen: According to Figure 5.5, the $k = 1$ scheme needs at least one level of uniform mesh refinement on the finest grid to achieve an entropy norm of 10^{-7}. While such a mesh refinement would increase the resource consumption by a minimum factor of eight, the $k = 3$ scheme achieves the same level of accuracy by consuming only two times the same resources.

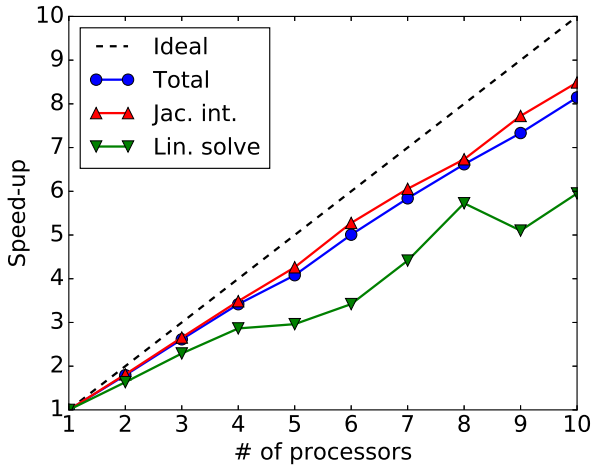


Figure 5.8: The parallel speedup of the solver for the sphere problem: $N_{\text{CV}} = 322\text{K}$ and $k = 3$.

To test the parallel scalability of the flow solver, the sphere problem was solved on different numbers of processors ranging from 1 to 10 with $N_{\text{CV}} = 322\text{K}$ and $k = 3$. The parallel speedup of the total solution process, the Jacobian evaluation algorithm, and the linear solver scheme were computed separately, and are shown in Figure 5.8. Not surprisingly, the linear solver algorithm is less scalable than that of the Jacobian integrator mainly because the performance of the BJ preconditioner decreases with an increase in the number of processors. The sudden dip of the parallel speedup for 9 processors may be due to a less favorable mesh partitioning, or communication problems on the cluster. Regardless, the overall solution algorithm scales very well in parallel, as the linear solver only constitutes a small part of the overall process.

5.3 Turbulent Flow Over a Flat Plate

For this test case, we consider a three-dimensional extension of the flat plate verification case from the NASA TMR website. The computational domain is $\Omega = [-0.33, 2] \times [0, 1] \times [0, 1]$, and the nondimensional parameters have the values: $Re = 5 \times 10^6$, $Ma = 0.2$, $\alpha = 0$, and $\psi = 0$. The flat plate is located on the $(0 \leq x_1 \leq 2) \wedge (x_2 = 0)$ boundary, where adiabatic solid wall conditions are imposed. Symmetry boundary conditions are imposed on the $(x_3 = 0)$, $(x_3 = 1)$, and $(-0.33 \leq x_1 \leq 0) \wedge (x_2 = 0)$ boundaries, while other boundaries are considered as farfield. As symmetry boundary conditions are imposed on all the boundaries normal to the x_3-axis, the exact solution of this problem must be constant in the x_3 direction. Therefore, the solutions obtained in this section can be compared to the two-dimensional results from the NASA TMR website, which are obtained by the CFL3D solver [1] on a 544×384 grid.

The two-dimensional meshes provided by the NASA TMR website are extruded in the x_3 direction to construct a series of nested three-dimensional grids with dimensions: $60 \times 34 \times 7$, $120 \times 68 \times 14$, and $240 \times 136 \times 28$. The coarsest mesh is depicted in Figure 5.9. An anisotropic mesh is necessary

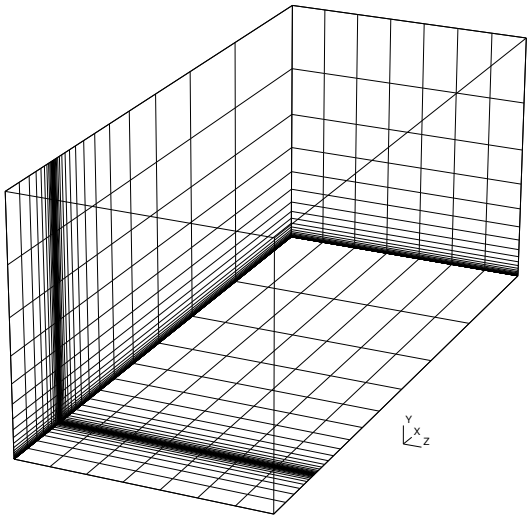
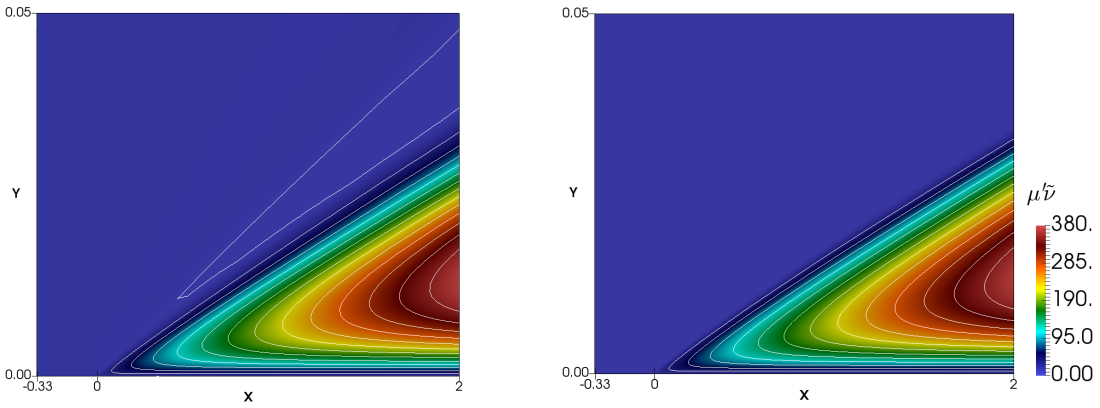


Figure 5.9: Coarsest mesh for the flat plate problem.

to correctly capture the flow pattern in the boundary layer and at the plate leading edge. On each mesh, the problem is solved for $k = 1$, then for $k = 2$, and finally for $k = 3$. The free-stream conditions are used as the initial conditions for $k = 1$, while the initial solution for each $k \geq 2$ is taken from the converged solution of the $(k-1)$-exact scheme. The GMRES-LO-ILU2 method is used as the preconditioner, with 10 inner GMRES iterations, and the RCM reordering algorithm. Moreover, the ILU factorization is performed only on the diagonal block of each processor. Other solver parameters are the same as Section 5.2.

To schematically demonstrate the correctness of the solution, the distribution of the turbulence working variable obtained from $k = 3$ on the finest mesh is plotted on the $x_3 = 0.5$ plane, and shown in Figure 5.10a. The results for a 544×384 grid provided by the NASA TMR are also shown in Figure 5.10b. There is a close agreement between the reference solution of NASA TMR and the solution of this thesis, although a few undershoots of small magnitude are present in the latter. The next quantity of interest is the distribution of the nondimensional eddy viscosity $\frac{\mu_T}{\mu}$ on the line $(x_1 = 0.97) \wedge (x_3 = 0.5)$, which is depicted in Figure 5.11. Note how the solution discontinuity at control volume boundaries is quite evident for $k = 1$, while higher-order reconstruction schemes result in much smoother solutions. The reference solution from the NASA TMR website is also shown in Figure 5.11. An acceptable agreement is observable between our computed results and the reference values. Also, it is evident that a higher-order reconstruction scheme leads to a more accurate estimate of the eddy viscosity.

To further verify the numerical solution, the convergence of the drag coefficient C_D and the skin friction coefficient C_f at the point $\mathbf{x} = (0.97, 0, 0.5)$ are studied with mesh refinement. Table 5.2 shows the computed values on different meshes. The computed values converge faster for a higher-order reconstruction scheme, such that only $k = 3$ offers an accurate solution on the medium mesh. The reference values from the NASA TMR library are also shown in the same table. As desired, there is a good agreement between the computed values using the method of this thesis and the reference values



(a) 3-exact: min : 7×10^{-3}, max : 379

(b) NASA TMR: min : 0, max : 378

Figure 5.10: Distribution of the turbulence working variable on the plane $x_3 = 0.5$ for the flat plate problem

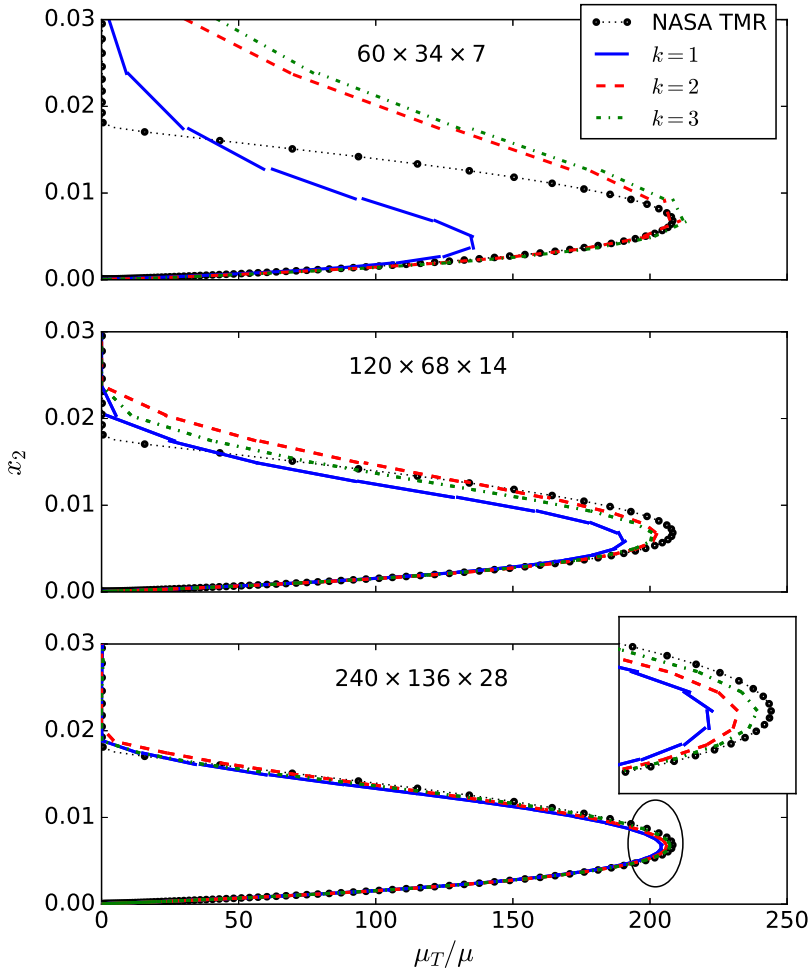


Figure 5.11: Distribution of the nondimensional eddy viscosity on the line $(x_1 = 0.97) \wedge (x_3 = 0.5)$ for the flat plate problem

Table 5.2: Computed value and convergence order of the drag coefficient and the skin friction coefficient at the point $\mathbf{x} = (0.97, 0, 0.5)$ for the flat plate problem

	C_D			C_f		
NASA TMR	0.00286			0.00271		
Mesh \ k	1	2	3	1	2	3
$60 \times 34 \times 7$	0.00396	0.00233	0.00233	0.00350	0.00228	0.00222
$120 \times 68 \times 14$	0.00301	0.00281	0.00285	0.00283	0.00268	0.00271
$240 \times 136 \times 28$	0.00287	0.00286	0.00286	0.00274	0.00273	0.00273
Convergence order	2.8	3.3	5.4	3	3	5.1

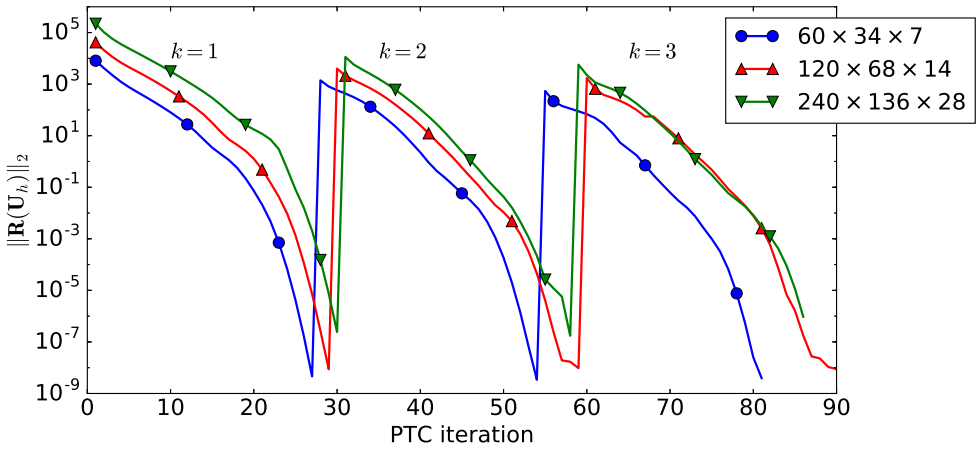


Figure 5.12: Norm of the residual vector per PTC iteration for the flat plate problem

of the NASA TMR website. The convergence orders of C_D and C_f are computed using the procedure of Celik et al. [16], and are also shown in Table 5.2. All the reconstruction schemes have attained optimal convergence rates. Nevertheless, care must be taken when interpreting the estimated convergence rates because the procedure of Celik et al. is most reliable when the error has already achieved an asymptotic behavior on the coarsest mesh, which might not necessarily be the case here.

The norm of the residual vector per PTC iteration is shown in Figure 5.12. Similar to Section 5.2, convergence is not affected by the reconstruction order k, but is slightly degraded as the number of degrees of freedom increases. The number of iterations and the resource consumption of the flow solver are listed in Table 5.3. The majority of the total time is spent on linear solves, showing that the linear systems arising from viscous turbulent flows are much stiffer compared to their inviscid counterparts. As desired, memory consumption increases linearly with mesh refinement, whereas the linear solve time does not scale as nicely as for the inviscid sphere problem. Most importantly, the benefit of higher-order methods can once again be observed: While the $k = 3$ solution on the medium mesh offers the same level of accuracy as the $k = 1$ solution on the fine mesh, the computational time of the former is smaller than that of the latter by a factor of four.

Table 5.3: Number of iterations and the resource consumption of the flow solver for the flat plate problem

k	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
$60 \times 34 \times 7$ mesh					
1	26	844	0.42	31	55
2	26	1,009	1.35	42	124
3	26	1,071	2.10	57	202
$120 \times 68 \times 14$ mesh					
1	28	1,436	5.24	510	713
2	29	1,864	8.30	742	1,489
3	29	2,041	14.63	825	2,124
$240 \times 136 \times 28$ mesh					
1	29	2,492	38.77	6,222	7,884
2	27	3,305	60.70	12,353	18,137
3	27	2,906	121.81	23,141	35,412

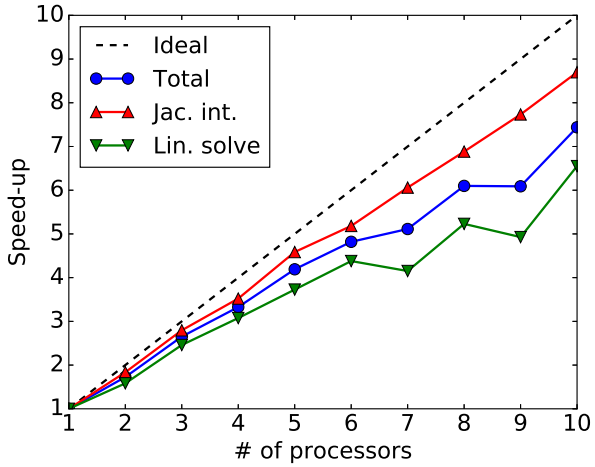
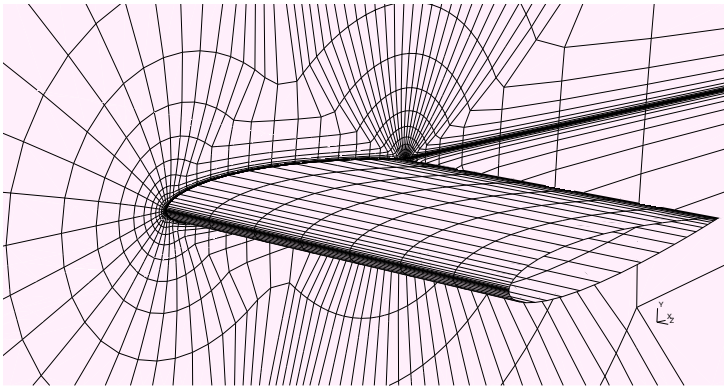


Figure 5.13: The parallel speedup of the solver for the flat plate problem: $120 \times 68 \times 14$ mesh and $k = 3$.

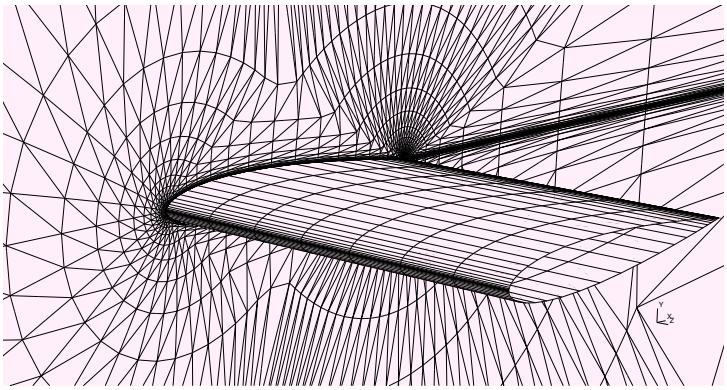
A strong parallel scaling test is conducted for the $k = 3$ scheme on the $120 \times 68 \times 14$ mesh. The resulting speedup for different parts of the solver is shown in Figure 5.13, which is quite similar to the results of the sphere problem, and suggests that the preconditioning algorithm is performing well for viscous flow problems. Nevertheless, the net scaling of the solver degrades marginally compared to the sphere problem, as linear solves take up a bigger portion of the total solution time.

5.4 Turbulent Flow Over an Extruded Airfoil

For the last test case, we consider a three-dimensional extension of the flow around the NACA 0012 airfoil. The computational domain of Section 4.5 is extruded in the x_3 direction with an extrusion length of one nondimensional unit. Symmetry boundary conditions are imposed on the two boundaries normal



(a) Hexahedral mesh



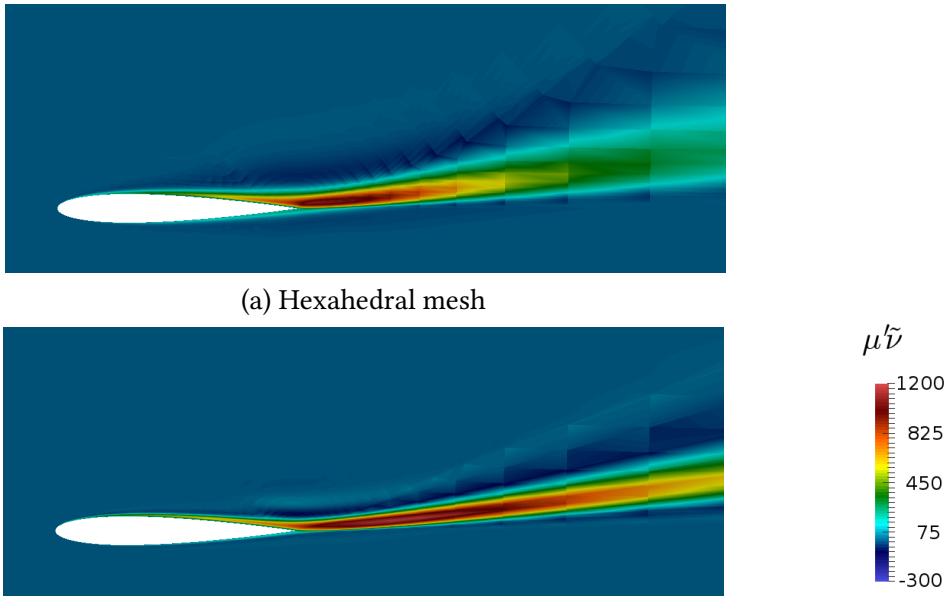
(b) Mixed prismatic-hexahedral mesh

Figure 5.14: Meshes for the extruded NACA 0012 problem

to the x_3-axis, while other boundaries retain their conditions from Section 4.5. The nondimensional parameters are: $Re = 6 \times 10^6$, $Ma = 0.15$, $\alpha = 10$, and $\psi = 0$. As the exact solution must be constant in the x_3 direction, the solutions are compared to the reference values provided by NASA TMR, which are obtained by the FUN3D [2] solver on a 7169×2049 grid.

Two meshes are employed: a hexahedral mesh with $N_{\text{CV}} = 100\text{K}$, and a mixed prismatic-hexahedral mesh with $N_{\text{CV}} = 176\text{K}$. In both cases, quadrature points are obtained from the tensor product of one-dimensional quadrature formulas and the quadrature points of the curved two-dimensional meshes. A portion of the meshes is depicted in Figure 5.14. In both cases, there are 7 layers of extruded control volumes in the x_3 direction. A value of $MinNeigh(3) = 30$ performed acceptably for the hexahedral mesh, whereas the same value resulted in an ill-conditioned version of Equation (2.7) for the mixed mesh. Therefore, $MinNeigh(3) = 60$ was used in the latter case as a safe measure to ensure a well-posed minimization problem for Equation (2.7). Other solver parameters, including the order ramping procedure for solution initialization, are the same as Section 5.3.

The computed contours of turbulence working variable on the $x_3 = 0$ plane are shown in Figure 5.15 for the $k = 3$ solutions. The mixed-mesh resolves the wake region more accurately, while providing a smoother distribution of the turbulence working variable. The superior performance of the mixed



(a) Hexahedral mesh

(b) Mixed prismatic-hexahedral mesh

Figure 5.15: Distribution of the turbulence working variable for the extruded NACA 0012 problem on the $x_3 = 0$ plane

mesh can be explained by its bigger number of degrees of freedom, and the presence of extra flow aligned faces that are missing in the hexahedral mesh.

The distribution of the surface pressure coefficient on the intersection of the airfoil and the $x_3 = 0.5$ plane is computed using the $k = 1$ and 3 solutions, and depicted in Figure 5.16. Generally, the computed distributions are consistent with the reference values obtained by FUN3D. Closer views of the plots are shown for four distinct points to better compare the results. As expected, the $k = 3$ scheme predicts the most accurate values, particularly on the mixed mesh. For example, the FUN3D results show that the pressure coefficient on the lower surface becomes bigger than that of the upper surface, near the trailing edge of the airfoil. This phenomenon is captured by the $k = 3$ solutions, but not observed by those of $k = 1$.

The computed and the reference lift and drag coefficients are shown in Table 5.4. It seems that the hexahedral mesh is too coarse, as none of the schemes can obtain accurate enough solutions. For the mixed mesh, the $k = 3$ scheme gives satisfactory results with less than 10% error for all the coefficients. The solution of the other schemes, however, is quite off, particularly for the pressure drag coefficient C_{Dp}. Note that the reference results are obtained using a 7169×2049 mesh, which is more than a thousand times finer than the meshes employed in this thesis.

The norm of the residual vector per PTC iteration is shown in Figure 5.17. Convergence is marginally affected by either the mesh type or the reconstruction order k. The latter is a desirable result of using the steady-state solution of a k-exact scheme as the initial guess for the $(k+1)$-exact solution.

Finally, parameters related to the iterative convergence of the solver are listed in Table 5.5. The linear

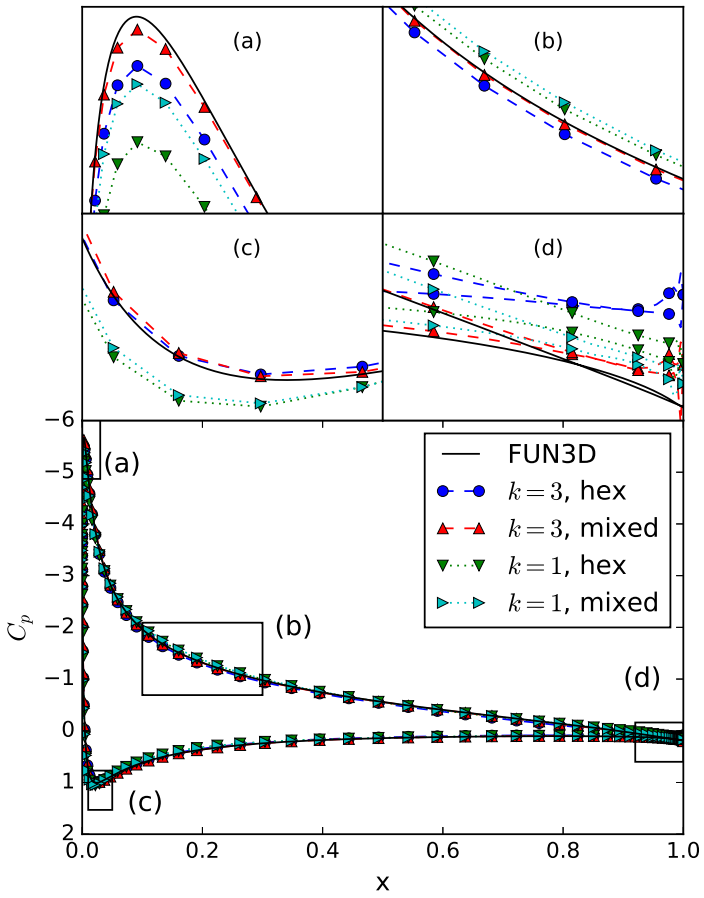


Figure 5.16: Distribution of surface pressure coefficient on the intersection of the extruded NACA 0012 airfoil and the $x_3 = 0.5$ plane.

Table 5.4: Computed drag and lift coefficients for the extruded NACA 0012 airfoil problem

k	C_{Dp}	C_{Dv}	C_L
NASA TMR			
–	0.00607	0.00621	1.0910
Hex mesh			
1	0.01703	0.00582	1.0619
2	0.01702	0.00497	1.0507
3	0.00301	0.00472	1.0417
Mixed mesh			
1	0.01129	0.00574	1.0735
2	0.00365	0.00565	1.0776
3	0.00550	0.00536	1.0869

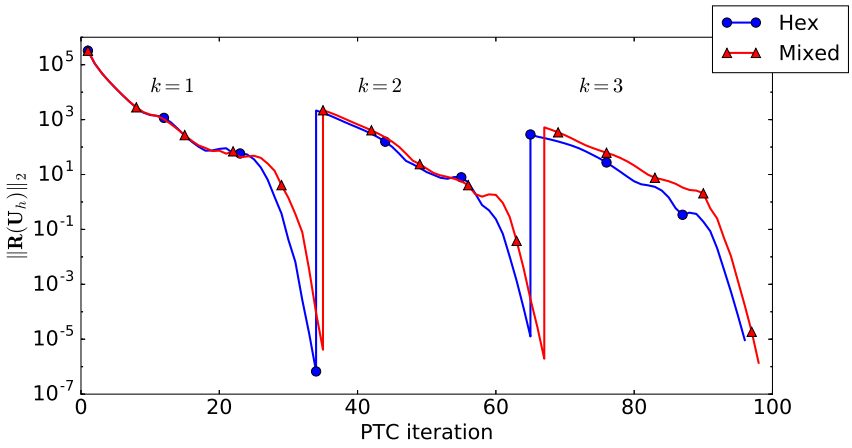


Figure 5.17: Norm of the residual vector per PTC iteration for the extruded airfoil problem

Table 5.5: Resource consumption of the flow solver for the extruded NACA 0012 problem

k	N-PTC	N-GMRES	Memory(GB)	LST(s)	TST(s)
Hex mesh, N_{CV} = 100K					
1	33	1,154	4.77	317	744
2	31	1,788	6.82	730	2,097
3	31	2,415	12.23	1,057	3,215
Mixed mesh, N_{CV} = 176K					
1	34	1,132	8.70	458	1,164
2	32	1,769	10.87	800	2,427
3	31	2,185	26.47	1,311	4,666

solve time makes up around 30% of the total computation time. The total computation time is strongly dependent on k, but does not differ considerably between the two meshes for the same reconstruction order. Moreover, the memory consumption seems to be linearly increasing with both k and N_{CV}.

Chapter 6

Conclusions

A higher-order solution framework was presented for steady-state three-dimensional compressible flows. The highlights of this solution framework include: a k-exact finite volume discretization, a PTC solution algorithm, a memory lean preconditioner based on inner GMRES iterations, and an ILU reordering algorithm based on lines of strong coupling between them.

Higher-order accuracy was achieved by constructing a piecewise continuous representation of the control volume average values. This piecewise continuous representation, also referred to as the discrete solution, was defined as the superposition of a set of basis functions. Monomials of Cartesian, principal, or curvilinear coordinates were used as the basis functions while the coefficient of each monomial in the superposition was found by solving a constrained minimization problem. In the finite volume method, the convective fluxes were evaluated by the Roe flux function. The viscous fluxes were obtained by adding a stabilizing damping term to the averaged gradients of the two adjacent states. Specifically, the flux functions were presented for the RANS equations fully coupled with the negative S-A turbulence model.

The curvature of the domain boundaries must be correctly accounted for in high-order numerical discretization schemes, yet simply replacing the boundary faces of highly anisotropic meshes with higher-order representations can result in invalid intersecting elements. Thus, a three-dimensional finite element elasticity solver was developed to propagate the curvature of the boundary throughout the domain, and to prevent mesh tangling. The developed solver was tested by solving a model problem that imitated the geometry of a three-dimensional airfoil. Also, the principal coordinates and the curvilinear coordinates basis functions were presented to prevent the k-exact reconstruction scheme from behaving poorly on curved anisotropic meshes.

The PTC method was revisited for the solution of the discretized nonlinear equations, where each iteration required the solution of a linear system. The challenges involved in the solution of these linear systems were addressed for three-dimensional problems, and a novel method was introduced to mitigate these issues. In this method, the FGMRES linear solver was preconditioned using inner

GMRES iterations, which were subsequently preconditioned by the ILU method. The $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices were factored from the LHS matrix of the 0-exact discretization scheme, and the lines of strong unknown coupling were employed to reorder the unknowns. By considering the 2-D fully turbulent flow around the NACA 0012 airfoil, it was shown that the proposed preconditioning algorithm is more efficient both in terms of solution time and memory consumption, compared to the other ILU based preconditioning methods considered.

The correct implementation of element quadrature and connectivity information in the solver was verified by solving Poisson's equation inside a cubic domain. Subsequently, inviscid and viscous turbulent test problems were considered, where the solution was verified against reference values either obtained analytically or provided by the NASA TMR website. In all cases, a satisfactory agreement was observed between the solutions of this thesis and the reference values. Moreover, accuracy tests were performed with mesh refinement, and optimal convergence order was attained for reconstruction orders $k = 1$ and 3. The $k = 2$ scheme, however, was slightly inferior in performance compared to its theoretical convergence order.

Timing results demonstrated the benefit and practicality of using higher-order methods for obtaining a certain level of accuracy. A second-order discretization scheme required a finer grid and more computational time to attain the same level of accuracy compared to a higher-order discretization scheme. Furthermore, strong parallel scaling tests were performed, and excellent scaling was observed for the Jacobian integration algorithm. The linear solver algorithm also demonstrated satisfactory parallel scaling results, although the performance in this area can still be improved. It was also observed that the required number of PTC iterations for convergence is independent of the reconstruction order while only slightly affected by the grid size.

Much work remains to be done to extend the solution method of this thesis to solve industrially practical problems:

1. A theoretical study can be conducted to investigate the inferior performance of the $k = 2$ discretization scheme. The theoretical approach for unstructured mesh-based discretization methods is usually through the functional analysis framework. A possible starting point can be the work of Barth and Larson [13] where they prove theoretical convergence orders for the k-exact finite volume method.

2. The boundary condition implementations of this work, though functional, might not be the best choice. An investigation of other possible boundary condition implementations has the potential to improve convergence properties and solution quality [29].

3. The proposed preconditioning algorithm does not require the explicit form of the full-order LHS matrix. Thus, a matrix-free method can be implemented, where the matrix-by-vector product of the full-order LHS matrix is evaluated by finite difference Fréchet derivatives while only the 0-exact LHS matrix is assembled. This strategy can reduce the memory cost considerably [55], while possibly increasing the computation time. Sacrificing computation time for less memory

usage might be unavoidable for very large problems.

4. In this thesis, it was possible to directly use the 2-D curvilinear coordinate basis functions for 3-D problems, as the solutions of the test problems considered were constant in the x_3 direction. A practical problem, on the other hand, requires the definition of truly 3-D curvilinear coordinate basis functions. It should be possible to generalize the method of Section 3.3 by providing a definition for the third wall coordinate. Alternatively, a big hexahedral meta-element can be constructed by extruding a portion of the nearest wall surface to include all the members of a target reconstruction stencil. The reference coordinates of the constructed meta-element can then be used as the curvilinear coordinates.

5. The original 0-exact discretization scheme does not take the gradient dependent source terms into account. Modifying this scheme to somehow capture the influence of the gradient dependent source terms can greatly enhance the performance of the preconditioning algorithm [78].

6. Employing a line Gauss-Seidel preconditioner instead of the ILU method, in conjunction with the proposed inner GMRES preconditioner has the potential to improve the parallel scalability of the linear solver [4].

7. As already mentioned, real world problems contain solution discontinuities, and require shock capturing methods to be used in conjunction with the k-exact reconstruction scheme. Shock capturing is even more important for higher-order methods, as they are more prone to developing unstable and spurious solutions.

8. Haider et al. [26] recently introduced a multi-level reconstruction scheme that uses compact reconstruction stencils, and eliminates the need to directly work with large stencils. Their high-order finite volume method is therefore better suited for parallel simulations. Although their work is limited to the linear advection equation, its application in the context of turbulent compressible flows is worth investigating.

9. An hp-adaptive strategy can be much more efficient in achieving a prescribed level of accuracy, compared to uniform mesh refinement, or using high reconstruction orders for all the control volumes [33, 38]. Desirably, hanging nodes resulting from adaptively refined meshes fit naturally into the k-exact finite volume framework. Nevertheless, an effective error estimator must be designed that selects the candidate regions for refinement. In the case of anisotropic mesh refinement, the error estimator must also provide an estimate of the direction in which the error changes most abruptly.

Bibliography

[1] Computational Fluids Laboratory 3-D, 2011. URL https://cfl3d.larc.nasa.gov/.

[2] Fully Unstructured Navier–Stokes in 3-D, 2011. URL http://fun3d.larc.nasa.gov/.

[3] WestGrid computing resources, 2017. URL https://www.westgrid.ca/.

[4] B. R. Ahrabi and D. J. Mavriplis. Scalable solution strategies for stabilized finite-element flow solvers on unstructured meshes. In *Proceedings of the Fifty-Fifth AIAA Aerospace Sciences Meeting*, page 517, 2017.

[5] B. R. Ahrabi, W. K. Anderson, and J. C. Newman III. An adjoint-based hp-adaptive stabilized finite-element method with shock capturing for turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 318:1030–1065, 2017.

[6] S. R. Allmaras, F. T. Johnson, and P. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *Proceedings of the Seventh International Conference on CFD*, 2012.

[7] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. MUMPS multifrontal massively parallel solver version 2.0. Technical report, CERFACS: Centre EuropÃľen de Recherche et de Formation AvancÃľe en Calcul Scientifique, 1998. URL http://mumps.enseeiht.fr/.

[8] J. Andren, H. Gao, M. Yano, D. Darmofal, C. Ollivier Gooch, and Z. Wang. A comparison of higher-order methods on a set of canonical aerodynamics applications. In *Proceedings of the Twentieth AIAA Computational Fluid Dynamics Conference*, page 3230, 2011.

[9] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, 19(4):742–760, 1982.

[10] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015. URL http://www.mcs.anl.gov/petsc.

[11] G. E. Barter and D. L. Darmofal. Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation. *Journal of Computational Physics*, 229(5):1810–1827, 2010.

[12] T. J. Barth and P. O. Frederickson. Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. In *Twenty-eighth AIAA Aerospace Sciences Meeting*, Jan. 1990. AIAA paper 90-0013.

[13] T. J. Barth and M. G. Larson. A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes. *Finite Volumes for Complex Applications III, London*, 2002.

[14] F. Bassi and S. Rebay. High-order accurate discontinuous finite element solution of the 2D Euler equations. *Journal of Computational Physics*, 138(2):251–285, 1997.

[15] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2016.

[16] I. B. Celik, U. Ghia, and P. J. Roache. Procedure for estimation and reporting of uncertainty due to discretization in CFD applications. *Journal of Fluids Engineering-Transactions of the ASME*, 130(7), 2008.

[17] M. Ceze and K. J. Fidkowski. Constrained pseudo-transient continuation. *International Journal for Numerical Methods in Engineering*, 102(11):1683–1703, 2015.

[18] S.-W. Cheng, T. K. Dey, and J. Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.

[19] X. Deng, M. Mao, G. Tu, H. Zhang, and Y. Zhang. High-order and high accurate CFD methods and their applications for complex grid problems. *Communications in Computational Physics*, 11 (04):1081–1102, 2012.

[20] L. T. Diosady and D. L. Darmofal. Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations. *Journal of Computational Physics*, 228(11):3917–3935, 2009.

[21] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 207(1):92–113, 2005.

[22] J. E. Flaherty. Course notes on finite element analysis. URL http://www.cs.rpi.edu/~flaherje/pdf. Accessed 05/12/17.

[23] S. K. Godunov. A finite difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematicheskie Sborniki*, 47:357–393, 1959.

[24] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1983.

[25] F. Haider, J.-P. Croisille, and B. Courbet. Stability analysis of the cell centered finite-volume Muscl method on unstructured grids. *Numerische Mathematik*, 113(4):555–600, 2009.

[26] F. Haider, P. Brenner, B. Courbet, and J.-P. Croisille. Efficient implementation of high order reconstruction in finite volume methods. In *Finite Volumes for Complex Applications VI Problems & Perspectives*, pages 553–560. Springer, 2011.

[27] F. Haider, P. Brenner, B. Courbet, and J.-P. Croisille. Parallel implementation of k-exact finite volume reconstruction on unstructured grids. In *High-Order Nonlinear Numerical Schemes for Evolutionary PDEs*, pages 59–75. Springer, 2014.

[28] A. Harten, P. D. Lax, and B. Van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. In *Upwind and High-Resolution Schemes*, pages 53–79. Springer, 1997.

[29] R. Hartmann and T. Leicht. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. *International Journal for Numerical Methods in Fluids*, 2016.

[30] C. Hirsch. *Numerical Computation of Internal and External Flows: Computational Methods for Inviscid and Viscous Flows*. Wiley, 1990.

[31] H. Huynh, Z. J. Wang, and P. E. Vincent. High-order methods for computational fluid dynamics: a brief review of compact differential formulations on unstructured grids. *Computers and Fluids*, 98:209–220, 2014.

[32] A. Jalali and C. Ollivier-Gooch. Higher-order unstructured finite volume RANS solution of turbulent compressible flows. *Computers and Fluids*, 143:32–47, 2017.

[33] A. Jalali and C. F. Ollivier Gooch. An hp-adaptive ustructured finite volume solver for compressible aerodynamic flows. In *Proceedings of the Fifty-Fifth AIAA Aerospace Sciences Meeting*, page 0082, 2017.

[34] A. Jalali, M. Sharbatdar, and C. Ollivier-Gooch. Accuracy assessment of finite volume discretization schemes on unstructured meshes. *Computers and Fluids*, 101:220–232, 2014. doi:10.1016/j.compfluid.2014.06.008.

[35] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998.

[36] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3-4): 237–254, 2006.

[37] W. M. Lai, D. H. Rubin, D. Rubin, and E. Krempl. *Introduction to Continuum Mechanics*. Butterworth-Heinemann, 2009.

[38] T. Leicht and R. Hartmann. Anisotropic mesh refinement for discontinuous Galerkin methods in two-dimensional aerodynamic flow simulations. *International Journal for Numerical Methods in Fluids*, 56(11):2111–2138, 2008.

[39] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992.

[40] W. Li. *Efficient Implementation of High-Order Accurate Numerical Methods on Unstructured Grids*. Springer, 2014.

[41] H. Luo, J. D. Baum, and R. Löhner. A fast, matrix-free implicit method for compressible flows on unstructured grids. In *Sixteenth International Conference on Numerical Methods in Fluid Dynamics*, pages 73–78. Springer, 1998.

[42] D. Mavriplis, D. Darmofal, D. Keyes, and M. Turner. Petaflops opportunities for the NASA fundamental aeronautics program. In *Proceedings of the Eighteenth AIAA Computational Fluid Dynamics Conference*, page 4084, 2007.

[43] D. J. Mavriplis. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics*, 145(1):141–165, 1998.

[44] D. J. Mavriplis. Directional agglomeration multigrid techniques for high-Reynolds-number viscous flows. *AIAA Journal*, 37(10):1222–1230, 1999.

[45] D. J. Mavriplis. Grid resolution study of a Drag Prediction Workshop configuration using the NSU3D unstructured mesh solver. In *Proceedings of the Twenty-Third AIAA Applied Aerodynamics Conference*, June 2005. AIAA paper 2005-4729.

[46] C. Michalak and C. Ollivier-Gooch. Accuracy preserving limiter for the high-order accurate solution of the Euler equations. *Journal of Computational Physics*, 228(23):8693–8711, 2009. doi:10.1016/j.jcp.2009.08.021.

[47] C. Michalak and C. Ollivier-Gooch. Globalized matrix-explicit Newton-GMRES for the high-order accurate solution of the Euler equations. *Computers and Fluids*, 39:1156–1167, 2010. doi:10.1016/j.compfluid.2010.02.008.

[48] C. R. Nastase and D. J. Mavriplis. High-order discontinuous Galerkin methods using an *hp*-multigrid approach. *Journal of Computational Physics*, 213(1):330–357, 2006.

[49] A. Nejat and C. Ollivier-Gooch. A high-order accurate unstructured finite volume Newton-Krylov algorithm for inviscid compressible flows. *Journal of Computational Physics*, 227(4):2582–2609, 2008.

[50] H. Nishikawa. Beyond interface gradient: A general principle for constructing diffusion scheme. In *Proceedings of the Fortieth AIAA Fluid Dynamics Conference*, 2010. AIAA paper 2010-5093.

[51] H. Nishikawa. Two ways to extend diffusion schemes to Navier-Stokes schemes: Gradient formula or upwind flux. In *Proceedings of the Twentieth AIAA Computational Fluid Dynamics Conference*, page 3044, 2011.

[52] T. Okusanya, D. Darmofal, and J. Peraire. Algebraic multigrid for stabilized finite element discretizations of the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 193(33):3667–3686, 2004.

[53] C. F. Ollivier-Gooch. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. *Journal of Computational Physics*, 133(1):6–17, 1997.

[54] C. F. Ollivier-Gooch and M. Van Altena. A high-order accurate unstructured mesh finite-volume scheme for the advection-diffusion equation. *Journal of Computational Physics*, 181(2):729–752, sep 2002. doi:10.1006/jcph.2002.7159.

[55] P.-O. Persson and J. Peraire. Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 30(6): 2709–2733, 2008.

[56] A. Pueyo and D. W. Zingg. Efficient Newton-Krylov solver for aerodynamic computations. *AIAA Journal*, 36(11):1991–1997, 1998.

[57] P. Roe and J. Pike. Efficient construction and utilisation of approximate Riemann solutions. In *Proceedings of the Sixth International Symposium on Computing Methods in Applied Sciences and Engineering, VI*, pages 499–518. North-Holland Publishing Co., 1985.

[58] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981. doi:10.1016/0021-9991(81)90128-5. URL http://www.sciencedirect.com/science/article/pii/0021999181901285.

[59] A. Rogerson and E. Meiburg. A numerical study of the convergence properties of ENO schemes. *Journal of Scientific Computing*, 5(2):151–167, 1990.

[60] C. Rumsey. Turbulence Modeling Resource, 2014. URL http://turbmodels.larc.nasa.gov/.

[61] V. Rusanov. *Calculation of Interaction of Non-steady Shock Waves with Obstacles*. Technical translation. National Research Council of Canada, 1962.

[62] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[63] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[64] K. Salari and P. Knupp. Code verification by the method of manufactured solutions. Technical report, Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US), 2000.

[65] K. Shahbazi, D. J. Mavriplis, and N. K. Burgess. Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 228(21):7917–7940, 2009.

[66] C.-W. Shu. High order ENO and WENO schemes for computational fluid dynamics. In *High-order Methods for Computational Physics*, pages 439–582. Springer, 1999.

[67] P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods*. CRC Press, 2003.

[68] P. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. In *Thirtieth AIAA Aerospace Sciences Meeting*, Jan. 1992. AIAA paper 92-0439.

[69] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011, Oct. 1984. doi:http://dx.doi.org/10.1137/0721062.

[70] E. Toro and A. Chakraborty. The development of a Riemann solver for the steady supersonic Euler equations. *The Aeronautical Journal*, 98(979):325–339, 1994.

[71] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, 2013.

[72] B. Van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of Computational Physics*, 23(3):276–299, 1977.

[73] B. Van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 135(2):229–248, 1997.

[74] J. Vassberg, M. DeHaan, and T. Sclafani. Grid generation requirements for accurate drag predictions based on OVERFLOW calculations. In *Proceedings of the Sixteenth AIAA Computational Fluid Dynamics Conference*, page 4124, 2003.

[75] M. R. Visbal and D. V. Gaitonde. On the use of higher-order finite-difference schemes on curvilinear and deforming meshes. *Journal of Computational Physics*, 181(1):155–185, 2002.

[76] J. Von Neumann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232–237, 1950.

[77] M. Wallraff, R. Hartmann, T. Leicht, N. Kroll, C. Hirsch, F. Bassi, C. Johnston, and K. Hillewaert. Multigrid solver algorithms for DG methods and applications to aerodynamic flows. *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, 128:153–178, 2015.

[78] P. Wong and D. W. Zingg. Three-dimensional aerodynamic computations on unstructured grids using a Newton–Krylov approach. *Computers and Fluids*, 37(2):107–120, 2008.

[79] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54(1):115–173, 1984.

[80] S. H. Yoo, H. C. No, H. M. Kim, and E. H. Lee. CFD-assisted scaling methodology and thermal-hydraulic experiment for a single spent fuel assembly. *Nuclear Engineering and Design*, 240(12):4008–4020, 2010.

Appendix A

ANSLib Command-Line Options

The command line interface provided by the ANSLib library consists of an executable file Solver, which can be executed through the Linux shell, with the input options provided in a separate text file:

```
$ ./Solver -opt < address of options file >
```

ANSLib achieves parallelism using the Message Passing Interface (MPI). Thus, parallel jobs can be executed via the MPI launcher :

```
$ mpiexec -np < # of processors > ./Solver -opt  < address of options file >
```

In the remainder, the input options are presented for the test problems considered in this thesis. Note that comments are distinguished by the character #, in the options file.

A.1 Two-Dimensional Turbulent Flow over a NACA 0012 Airfoil from Section 4.5

For this problem, several cases of preconditioners were considered. The following options are shared between all the cases:

Common Options

```
# -------------------------- MESH
-d 2                         # Dimensions
-mesh_type c                 # Cell-centered mesh
-f   < Location of .mesh file without extension >
-B   < Location of .bdry file without extension >
-fec < Location of .fec  file without extension >
# -------------------------- HISTORY AND SOLUTION FILE NAME
-sol               < Location of initial solution input file >
-sol_out           < Location of final solution output file >
-ita_history_name  < Location of residual history output file >
```

```
# -------------------------- ACCURACY
-a   < Order of accuracy of the scheme: a=k+1 >
# -------------------------- PHYSICS
-physics RoeTurbSA2D         # 2-D RANS + negative S-A
-reynolds  6e6               # Reynolds number
-mach 0.15                   # Mach number
-angle 10                    # Angle of attack
# -------------------------- NON-LINEAR SOLVER OPTIONS
-exnut 1                     # Ignore \tilde{nu} entries in the line-search
-C 0.01                      # Initial CFL number
-no_distance_weight          # No distance weights in the k-exact
                             # reconstruction least squares problem
-pseudolts_fixed             # No additional CFL ramping
-max_iter 80                 # Maximum number of PTC iterations
-jacobian_type recanalytic   # Evaluate the Jacobian exactly
-ita_target_residual 1e-5    # Target norm for the non-linear residual
```

To implement different preconditioning strategies, distinct sets of options have to be provided to the flow solver in each case:

Extra Options for Case A

```
# -------------------------- LINEAR SOLVER OPTIONS
# -------------------------- KSP
-ksp_max_it 500                  # Maximum # of GMRES iterations
-ksp_rtol 1e-3                   # rtol for the outer KSP
-ksp_type gmres                  # KSP solver type
# -------------------------- PC
-pc_type ilu                     # Preconditioner type: ILU
-pc_factor_levels 3              # ILU fill level: 3
-pc_factor_mat_ordering_type qmd  # ILU reordering type: QMD
```

Extra Options for Case B

```
# -------------------------- LINEAR SOLVER OPTIONS
# -------------------------- KSP
-ksp_max_it 500                  # Maximum # of GMRES iterations
-ksp_rtol 1e-3                   # rtol for the outer KSP
-ksp_type gmres                  # KSP solver type
# -------------------------- PC
-pre_order 1                       # Order of scheme for A*
-pc_type ilu                       # Preconditioner type: ILU
-pc_factor_levels 0                # ILU fill level: 0
-pc_factor_mat_ordering_type rcm   # ILU reordering type: RCM
```

Extra Options for Case C

```
# -------------------------- LINEAR SOLVER OPTIONS
# -------------------------- KSP
```

```
-ksp_max_it 500                   # Maximum # of GMRES iterations
-ksp_rtol 1e-3                    # rtol for the outer KSP
-ksp_type gmres                   # KSP solver type
# -------------------------- PC
-mu 1e-5                            # \mu_L for line construction
-pre_order 1                        # Order of scheme for A*
-pc_type ilu                        # Preconditioner type: ILU
-pc_factor_levels 0                 # ILU fill level: 0
-pc_factor_mat_ordering_type lines  # ILU reordering type: LINES
```

Extra Options for Case D

```
# -------------------------- LINEAR SOLVER OPTIONS
# -------------------------- KSP
-ksp_max_it 500                   # Maximum # of GMRES iterations
-ksp_rtol 1e-3                    # rtol for the outer KSP
-ksp_type fgmres                  # KSP solver type: FGMRES
# -------------------------- PC
-pre_order 1                      # Order of scheme for A*
-pc_type   ksp                    # Preconditioner type: inner-KSP
-ksp_ksp_type gmres               # inner-KSP type: GMRES
-ksp_ksp_max_it 10                # inner-KSP n_i: 10
-ksp_pc_type ilu                  # inner-KSP preconditioner: ILU
-ksp_pc_factor_levels 0                 # ILU fill level: 0
-ksp_pc_factor_mat_ordering_type rcm    # ILU reordering: RCM
```

Extra Options for Case E

```
# -------------------------- LINEAR SOLVER OPTIONS
# -------------------------- KSP
-ksp_max_it 500                   # Maximum # of GMRES iterations
-ksp_rtol 1e-3                    # rtol for the outer KSP
-ksp_type fgmres                  # KSP solver type: FGMRES
# -------------------------- PC
-mu 1e-5                          # \mu_L
-pre_order 1                      # Order of scheme for A*
-pc_type   ksp                    # Preconditioner type: inner-KSP
-ksp_ksp_type gmres               # inner-KSP type: GMRES
-ksp_ksp_max_it 10                # inner-KSP n_i: 10
-ksp_pc_type ilu                  # inner-KSP preconditioner: ILU
-ksp_pc_factor_levels 0                 # ILU fill level: 0
-ksp_pc_factor_mat_ordering_type lines  # ILU reordering: LINES
```

A.2 Poisson's Equation from Section 5.1

```
# -------------------------- MESH
-d 3               # Dimensions
-mesh_type c       # Cell-centered mesh
-f                 < Location of .vmesh file without extension >
# -------------------------- HISTORY AND SOLUTION FILE NAME
-sol_out           < Location of final solution output file >
-ita_history_name  < Location of residual history output file >
#-------------------------- PHYSICS
-physics Poisson                 # Solve Poisson's equation
-poisson_problem_data sinhsin    # Name of the manufactured solution
# -------------------------- ACCURACY
-a                 < Order of accuracy of the scheme: a=k+1 >
# -------------------------- NON-LINEAR SOLVER OPTIONS
-C 1e10                       # Initial CFL number
-jacobian_type recanalytic    # Evaluate the Jacobian exactly
-no_distance_weight           # No distance weights in the k-exact
                              # reconstruction least squares problem
-max_iter 3                   # Maximum number of PTC iterations
-ita_target_residual 1e-10    # Target norm for the residual vector R
# -------------------------- LINEAR SOLVER OPTIONS
-ksp_type gmres               # KSP type: GMRES
-ksp_rtol 1e-12               # GMRES rtol
-pre_order 1                  # Order of scheme for A*
-pc_type sor                  # Preconditioner type: PGS
-pc_sor_its 10                # Number of PGS iterations: 10
```

A.3 Inviscid Flow Over a Sphere from Section 5.2

```
# -------------------------- MESH
-d 3                     # Dimensions
-mesh_type c             # Cell-centered mesh
-f                 < Location of .vmesh file without extension >
# -------------------------- HISTORY AND SOLUTION FILE NAME
-sol_out           < Location of final solution output file >
-ita_history_name  < Location of residual history output file >
---------------------------- PHYSICS
-physics Euler3D         # Solve 3-D Euler equations
-mach 0.38               # Mach number
-angle 0                 # Angle of attack
# -------------------------- ACCURACY
-a                 < Order of accuracy of the scheme: a=k+1 >
-mcell3d_sphere_hack 1  # Curve the boundary of the sphere
-mcell3d_extraq_face 1  # Use quadrature rule of order k+1 for faces
-mcell3d_extraq_cell 1  # Use quadrature rule of order k+1 for cells
# -------------------------- NON-LINEAR SOLVER OPTIONS
```

```
-C 10                       # Initial CFL number
-jacobian_type recanalytic  # Evaluate the Jacobian exactly
-pseudolts_fixed            # No additional CFL ramping
-max_iter 25                # Maximum number of PTC iterations
-ita_target_residual 1e-6   # Target norm for the residual vector R
# -------------------------- LINEAR SOLVER OPTIONS
-ksp_type fgmres            # KSP type: FGMRES
-ksp_rtol 1e-3              # GMRES rtol
-pre_order 1                # Order of scheme for A*
-pc_type sor                # Preconditioner type: PGS parallelized by BJ
-pc_sor_its 4               # Number of local PGS iterations
-pc_sor_lits 4              # Number of BJ iterations
```

A.4 Turbulent Flow Over a Flat Plate from Section 5.3

```
# -------------------------- MESH
-d 3                    # Dimensions
-mesh_type c            # Cell-centered mesh
-f                < Location of .vmesh file without extension >
# -------------------------- HISTORY AND SOLUTION FILE NAME
-sol               < Location of initial solution input file >
-sol_out           < Location of final solution output file >
-ita_history_name  < Location of residual history output file >
---------------------------- PHYSICS
-physics TurbSA3D                 # Solve 3-D RANS + negative SA
-mach 0.2                         # Mach number
-reynolds 5e6                     # Reynolds number
-angle 0                          # Angle of attack
-turbsa3d_problem_data flat_plate  # Specify post-processing operations
# -------------------------- ACCURACY
-a                < Order of accuracy of the scheme: a=k+1 >
# -------------------------- NON-LINEAR SOLVER OPTIONS
-C 0.1                      # Initial CFL number
-jacobian_type recanalytic  # Evaluate the Jacobian exactly
-pseudolts_fixed            # No additional CFL ramping
-max_iter 25                # Maximum number of PTC iterations
-ita_target_residual 1e-8   # Target norm for the residual vector R
-no_distance_weight         # No weight for the k-exact
                            # reconstruction least squares problem
# -------------------------- LINEAR SOLVER OPTIONS
-ksp_type fgmres            # KSP type: FGMRES
-ksp_rtol 1e-3              # GMRES rtol
-pre_order 1                # Order of scheme for A*
-pc_type ksp                # Preconditioner type: inner-KSP
-ksp_ksp_max_it 10          # Inner-KSP n_i: 10
```

```
-ksp_ksp_type gmres         # Inner-KSP type: GMRES
-ksp_pc_type bjacobi        # Inner-KSP preconditioner: BJ
                            # BJ number of iterations defaults to 1
-ksp_sub_pc_type ilu        # BJ inner solver: ILU
-ksp_sub_pc_factor_levels 2                 # ILU fill level: 2
-ksp_sub_pc_factor_mat_ordering_type rcm    # ILU reordering type: RCM
```

A.5 Turbulent Flow Over an Extruded NACA 0012 Airfoil from Section 5.4

```
# -------------------------- MESH
-d 3                    # Dimensions
-mesh_type e            # Cell-centered extruded mesh
-mext_nlayer 7          # Number of extruded layers
-mext_btag 2            # Boundary tag for the extruded walls (symmetry)
-mext_length 1          # Extrusion length
-f                 < Location of .mesh file without extension >
-B                 < Location of .bdry file without extension >
-fec               < Location of .fec  file without extension >
# -------------------------- HISTORY AND SOLUTION FILE NAME
-sol               < Location of initial solution input file >
-sol_out           < Location of final solution output file >
-ita_history_name  < Location of residual history output file >
---------------------------- PHYSICS
-physics TurbSA3D                  # Solve 3-D RANS + negative SA
-mach 0.15                         # Mach number
-reynolds 6e6                      # Reynolds number
-angle 10                          # Angle of attack
-turbsa3d_problem_data naca0012    # Specify post-processing operations
# -------------------------- ACCURACY
-a                 < Order of accuracy of the scheme: a=k+1 >
# -------------------------- NON-LINEAR SOLVER OPTIONS
-C 0.1                      # Initial CFL number
-jacobian_type recanalytic  # Evaluate the Jacobian exactly
-pseudolts_fixed            # No additional CFL ramping
-max_iter 100               # Maximum number of PTC iterations
-ita_target_residual 1e-5   # Target norm for the residual vector R
-no_distance_weight         # No weight for the k-exact
                            # reconstruction least squares problem
# -------------------------- LINEAR SOLVER OPTIONS
-ksp_type fgmres            # KSP type: FGMRES
-ksp_rtol 1e-3              # GMRES rtol
-pre_order 1                # Order of scheme for A*
-pc_type ksp                # Preconditioner type: inner-KSP
-ksp_ksp_max_it 10          # Inner-KSP n_i: 10
```

```
-ksp_ksp_type gmres         # Inner-KSP type: GMRES
-ksp_pc_type bjacobi        # Inner-KSP preconditioner: BJ
                            # BJ number of iterations defaults to 1
-ksp_sub_pc_type ilu        # BJ inner solver: ILU
-ksp_sub_pc_factor_levels 2                 # ILU fill level: 2
-ksp_sub_pc_factor_mat_ordering_type rcm    # ILU reordering type: RCM
```

Appendix B

Sample Script for Running a Parallel Job on Grex

The computational work on WestGrid systems is carried out through a non-interactive batch job system. Rather than directly executing the solver in the interactive mode, a request has to be made in the form of a job script, which specifies the commands to be executed as well as the required memory, computation time, number of nodes, and number of processors. The following shows a sample job script that executes the solver using 8 processors, and redirects the output to a text file:

Sample Job Script

```
#!/bin/bash
#PBS -S /bin/bash             ## This is a request file for job submission
#PBS -l nodes=8:ppn=1         ## Number of nodes and processors per node
#PBS -l walltime=00:40:00     ## Maximum requested wall time
#PBS -l mem=10gb              ## Maximum requested total memory
#PBS -M <your-email>          ## Email the owner after the job is finished
#PBS -m abe                   ## Send the email at any occasion

## Run an external script that correctly sets the environment
## variables and loads Grex's modules.
## For more information on modules see Grex's website.
. ~/scripts/setenv.sh

## On many WestGrid systems a variable PBS_NP is automatically
## assigned the number of cores requested of the batch system.
## On systems where $PBS_NP is not available, one could use:
CORES=$(/bin/awk 'END {print NR}' $PBS_NODEFILE)
echo "Running on $CORES cores"

## Announce start time
echo "Starting run at: $(date)"
```

```
## Recommended MPI options from Petsc for running a parallel job.
NPMAX=${CORES}
MPI_BINDING='--bysocket --bind-to-socket --report-bindings'
MPIEXEC=mpiexec

## Call the ANSLib executable, provide the correct option file,
## and finally redirect the output to a file, so that it can
## be inspected later.
${MPIEXEC} ${MPI_BINDING} -np ${CORES} \
~/code/ANSLib/hooshi/apps/solver/Solver  \
-opt options/h2_p2_facet.opt \
|& tee pp_data/h2_p2_facet.out

## Announce that the job is finished.
echo "pbs/h2_p2_facet.pbs finished at: $(date)"
```

After a script has been created, it should be submitted for execution:

```
$ qsub <address of the script>
```

The batch job system will place the script in the execution queue, where the execution priority is evaluated based on the the requested amount of resources.