To The Graduate School:

The members of the Committee approve the dissertation of Li Wang presented on April 21, 2009.

Dr. Dimitri Mavriplis, Chairman

Dr. Paul Dellenback

Dr. Jonathan Naughton

Dr. Douglas Smith

Dr. Frederico Furtado

Dr. David Darmofal, External Examiner

APPROVED:

Dr. Paul Dellenback, Head, Department of Mechanical Engineering

Don A. Roth, Dean, The Graduate School

Wang, Li, Techniques for High-order Adaptive Discontinuous Galerkin Discretizations in Fluid Dynamics, Ph.D., Department of Mechanical Engineering, April, 2009.

The use of high-order discontinuous Galerkin (DG) discretizations has become more widespread over the last decade for solving convection-dominated computational fluid dynamics problems. The appeal of these methods relates to their favorable asymptotic accuracy properties, combined with compact stencils and favorable scalability properties on parallel computing architectures. This work covers advances in several areas of high-order DG discretizations, including the development of implicit solvers, discrete adjoint methods for shape optimization, and output-based error estimation and mesh and time-step adaptation.

For time-dependent problems, high-order implicit time-integration schemes are considered exclusively to avoid the stability restrictions of explicit methods, with particular emphasis on balancing spatial and temporal accuracy of the overall approach. In order to make the high-order schemes competitive, efficient solution techniques consisting of a p-multigrid approach driven by element Jacobi smoothers are investigated and developed to accelerate convergence of the non-linear systems, in which the results demonstrate h independent convergence rates, while remaining relatively insensitive to time-step sizes.

A framework based on discrete adjoint sensitivity analysis has also been developed for applications in shape optimization and goal-oriented error estimation. An adaptive discontinuous Galerkin algorithm driven by an adjoint-based error estimation procedure has been developed, which incorporates both h-, p- and combined hp-adaptive schemes, for producing accurate simulations at optimal cost in the objective functional of interest. Current results show superior performance of these adaptive schemes over uniform mesh refinement methods, as well as the potential of the hp refinement approach to capture strong shocks without limiters. Finally, the adjoint-based error estimation strategy is successfully extended to unsteady flow problems, where the time-dependent flow solution is solved in a forward manner in time but the corresponding unsteady adjoint solution is evaluated as a backward time integration.

Results demonstrate that this methodology provides accurate global temporal error prediction, and may be employed to drive an adaptive time-step refinement strategy for improving the accuracy of specified time-dependent functionals of interest.

TECHNIQUES FOR HIGH-ORDER ADAPTIVE DISCONTINUOUS GALERKIN DISCRETIZATIONS IN FLUID DYNAMICS

by

Li Wang

A dissertation submitted to the Department of Mechanical Engineering
and The Graduate School of The University of Wyoming
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
MECHANICAL ENGINEERING

Laramie, Wyoming
April, 2009

To My Family:

Emily and Ming

Contents

List of Figures

List of Tables

Acknowledgments

This thesis would not have been possible without the support of many exceptional people.

First, I would like to express my deepest gratitude to my advisor, Professor Dimitri Mavriplis, for guiding me to these studies, for providing me with the most interesting topics, for his invaluable support and consistent consideration, and for his enormous patience and guidance in my writing skills.

Second, I am grateful for the grant from the Office of Naval Research ONR N0014-04-10602 and the support from NASA Grant NX07AC31A for making this work possible. I would like to thank Dr. Amik St-Cyr from the National Center for Atmospheric Research (NCAR) and the 2008 SIParCS program for providing me with the interesting opportunity to work on high-order limiting and shock-capturing for discontinuous Galerkin methods.

I would also like to thank my colleagues in our CFD lab, Cristian Nastase, who developed the steady-state discontinuous Galerkin discretizations used in this work, and Zhi Yang, for his countless pointers to interesting problems. I would like to thank all my committee members for their reviews, and special thanks are given to the thesis external reader, Professor David Darmofal from Massachusetts Institute of Technology, for his constructive suggestions.

Further, many thanks to my parents-in-law for their support and helping me to take care of my daughter in the past few years. I would certainly like to thank my parents for their consistent encouragement. Without them I would not be the person I am.

Finally, and the most of all, I would like to thank my beloved husband, Ming. I thank him for supporting me to pursue what I like, and for sharing with me my worries and his joys. Also I would like to devote this work to my lovely daughter Emily who was born in July 2006.

Li Wang

The University of Wyoming
April 2009

Chapter 1

Introduction

The continuous growth of computer resources has led to the emergence of Computational Fluid Dynamics (CFD) as an indispensable technology for the analysis of real flow problems, often encountered in many engineering analysis and design problems. While numerical simulations allow the study and analysis of complex flow processes without resorting to expensive and time-consuming experimental measurements, experimental methods continue to play an important role in fluid mechanics due to inaccuracies and uncertainties in many numerical simulations. Therefore, the focus in current CFD technology improvement has concentrated on the development and improvement of computational methods and numerical techniques to make modern numerical simulations more accurate, efficient, stable and robust.

Early CFD approaches concentrated on finite-difference methods [1–3] for solving the governing partial differential equations of flow fields and these became widely used in the context of structured meshes for simple geometries. However, one of the main drawbacks of these methods lies in difficulties involved in extending these discretizations to unstructured meshes, which are critical for handling problems with complex geometries. Finite volume methods [4–6], on the other hand, can be used for arbitrary geometries, using structured or unstructured meshes, and furthermore, can easily be formulated to be discretely conservative [7], which makes these methods highly desirable for modeling problems where conservation is important, such as problems with shock waves or problems which must conserve a particular property exactly such as mass or energy transfer. However, most finite-volume discretizations have traditionally relied on low-order accurate (at most second-order) formulations, which in turn result in either low accuracy, or require extensively refined computational meshes for achieving high accuracy.

Over the last decade, much effort in the computational fluid dynamics community has concentrated on developing higher-order accurate discretizations [8–16] in order to improve the accuracy and efficiency of simulations with demanding accuracy requirements. The advantage of high-order discretization approaches is that they alter the asymptotic relationship between solution accuracy and resolution in a beneficial manner. On the other hand, these methods are more computationally expensive, and thus the increase in delivered accuracy must outweigh the additional computational expense to make these techniques practical [16–20]. At the same time, the incorporation of sensitivity analysis techniques [21–23] has become the subject of increased interest in many numerical simulations. The calculation of the sensitivities of specific simulation outputs with respect to simulation inputs can be used to deliver important information in a simulation which in turn can be used to drive an optimization process, or to estimate simulation error in the outputs [24–27], thus enabling an adaptive refinement strategy for reducing simulation error.

This thesis is focused on the investigation and development of novel techniques for advancing the state-of-the-art of computational fluid dynamics through the combination of both high-order discretizations and sensitivity analysis techniques. The thesis work relies exclusively on the use of higher-order discontinuous Galerkin discretizations, due to their ability to maintain discrete conservation in the presence of high-order discretizations on unstructured meshes about complex configurations. For sensitivity analysis techniques, this work relies exclusively on discrete adjoint techniques, which can be implemented in a relatively straight-forward manner, and through which sensitivities and/or error estimates can be computed in a robust and efficient manner.

1.1 Higher-order Methods

The original discontinuous Galerkin method was first introduced and analyzed through the work of Reed and Hill [28] as a technique to solve the scalar hyperbolic neutron transport equation. Lesaint and Raviart [29] presented the first numerical analysis for the method for linear hyperbolic problems. They showed that the L_2 norm of the error convergence in the DG method with an assumption of a smooth exact solution is $\mathcal{O}(h^p)$ for interpolating polynomials of order p and an average element size h. More numerical analysis on convergence rates was further studied by Johnson and Pitkaranta [30], Richter [31] and Peterson [32] for linear equations. The optimal convergence rate of $\mathcal{O}(h^{p+1})$ was proved to be achievable for structured two-dimensional non-Cartesian grids where the characteristic direction is not exactly aligned with the grids. The first extension of DG methods to nonlinear hyperbolic problems was attributed to the work of Chavent and Salzano [33], based on the use of a Riemann solver to evaluate fluxes across elemental interfaces. More fundamental work for DG methods applied to non-linear hyperbolic equations was further made by Cockburn and Shu through a series of works [12, 34–37] using explicit time-integration schemes to achieve high-order accuracy in space and in time. These methods also have been recently applied to second-order elliptic problems [9, 38–41] notably with extensions to the Navier-Stokes equations. More recently, efficient solution strategies [16–20, 42] have been investigated for improving the performance of high-order DG methods.

The general appeal of DG methods is due to their favorable properties for delivering higher accuracy at reasonable computational cost. Over the last decade, significant growth in computational resources and increasing requirements of high accuracy levels have made high-order accurate discontinuous Galerkin methods a popular choice for many current CFD simulations [8–18, 42]. DG methods rely on a set of polynomial basis functions defined only on each individual mesh element. The resulting accuracy order of these methods depends only on the approximating order of the polynomial basis functions. However, because these basis functions are only defined on individual elements, the solution approximation is discontinuous across elements. The resulting discontinuities can be handled using approximate Riemann solver techniques [43–47], which are now well understood and have been developed extensively in the context of finite-volume methods. Furthermore, a very attractive property of DG methods lies in the fact that the increase of the degree of approximating polynomials does not affect the numerical stencils since each element only communicates with its direct neighbors, regardless of the order of discretizations. The compactness of DG methods has clear advantages in boundary condition treatment and parallel computing [48]. In addition, the communication at shared element interfaces can easily be formulated to accommodate different discretization orders in neighboring elements, which makes DG methods ideally suited for implementing *h*-*p* adaptive refinement strategies [49–51].

For time-dependent problems, high-order time-integration schemes are required to work in conjunction with high-order spatial schemes to achieve an overall high error tolerance. While most current implementations focus on the use of explicit schemes, such as explicit Runge-Kutta discontinuous Galerkin (RKDG) methods [13, 18, 52], relatively little work has been done with regards to implicit time-integration schemes for DG methods, particularly higher-order implicit schemes, such as implicit Runge-Kutta schemes. Explicit schemes are well suited for problems with similar spatial and temporal scales, however, they are notoriously inefficient for problems with disparate temporal and spatial scales, such as low reduced frequency problems and for steady-state problems. The CFL stability limit of explicit schemes applied to high-order spatial discretizations becomes more restrictive as the order of the spatial discretization increases [13, 53]. As a consequence, implicit time-integration strategies, which are unconditionally stable and allow the selection of the time-step size based purely on temporal accuracy considerations [54–56], can be expected to provide a more effective approach for problems with disparate length and time scales or stiff problems. The use of high-order implicit time-integration methods [55, 57, 58] has been recently investigated for finite-volume methods. Thus, one of the goals of this work is to implement high-order implicit schemes for DG discretizations and to develop efficient solution strategies to make these schemes competitive.

1.2 Adjoint-based Sensitivity Analysis

In addition to the development of CFD analysis capabilities, sensitivity analysis capabilities have become indispensable in modern research areas related to aerodynamic design optimization and error estimation for functional outputs. Adjoint-based shape optimization for elliptic systems was first introduced by Pironneau [59] and was applied to transonic flow by Jameson [23]. Subsequently, Jameson et al. [21,22] pioneered this method for Euler and Navier-Stokes problems. In the context of unstructured meshes, a discrete adjoint approach was developed for shape optimization by Newman and Taylor [60, 61] and Elliot and Peraire [62, 63] for compressible inviscid flows and laminar viscous flows, respectively. In addition, Anderson and Venkatakrishman [64] developed a continuous adjoint approach using unstructured grids. While the majority of work in aerodynamic shape optimization has been focused on the design of aerospace vehicles in a steady flow environment, unsteady shape optimization techniques [65–68] have been developed as well for time-dependent flow problems. Additionally, adjoint-based sensitivity analysis strategies initially developed for design optimization have been extended to enable error estimation for quantifying solution error in specific simulation outputs. Practical formulations for output-based a posteriori error estimation based on adjoint techniques have been developed and demonstrated successfully, initially within the context of a variational framework [27,49,69–72], and more recently for finite-volume discretizations [24–26,73]. Mani and Mavriplis [74,75] further extended this methodology to temporal error estimation in conjunction with adaptive time-step refinement to reduce errors in time-dependent functionals in the context of finite-volume methods.

This work considers applications of the adjoint technique in the context of discontinuous Galerkin discretizations for both shape optimization problems and for a posteriori error estimation. In the context of design optimization problems, adjoint methods enable the calculation of the sensitivity of the objective with respect to any number of design variables at a cost which is essentially independent of the number of design variables, and roughly equivalent to an additional flow analysis problem. For optimization problems, when design variables produce changes in the shape of the computational domain boundaries, the linearization of the discretized flow equations with respect to the mesh geometry must be considered. In the case of DG discretizations, this involves the variation of element and face quadrature points as well as the element mappings, which may involve geometrically higher-order curved elements. Techniques for including these effects are formulated in this work and demonstrated on a simple airfoil optimization problem.

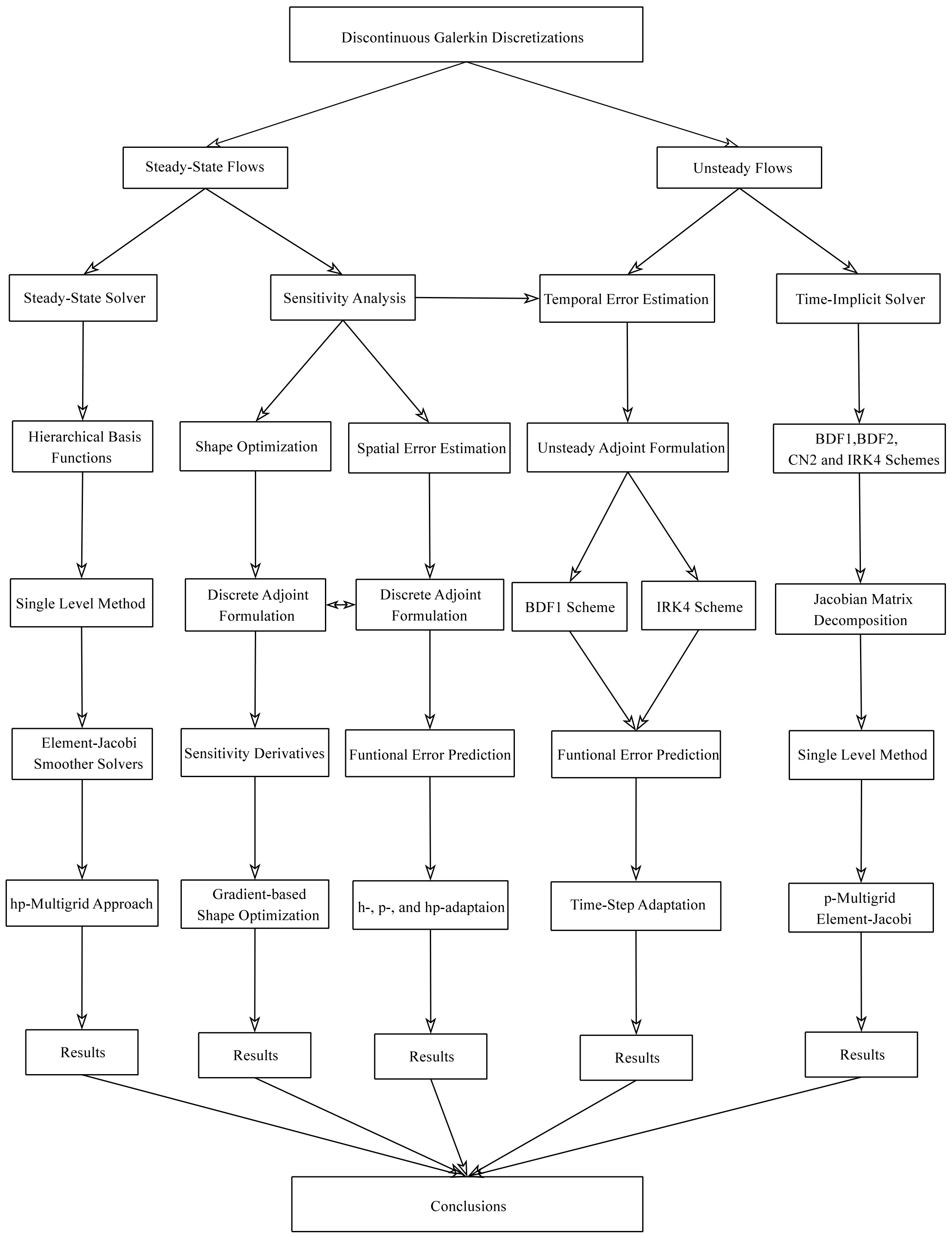
In addition to shape optimization problems, global error estimates for specified functionals of interest can be quantified by using the adjoint method. Furthermore, the resulting functional error estimates can also be divided into elemental contributions, which serve as a guide for automated mesh adaptation in an attempt to decrease and equidistribute the functional error. Since functional accuracy in discontinuous Galerkin methods can be improved by subdividing elements or by increasing the discretization order, an h-refinement scheme, or a p-enrichment scheme or a combined hp-refinement scheme can be employed to improve the quality of a computational mesh. This work develops a framework for estimating spatial functional error and implementing h-p refinement strategies, particularly for difficult problems such as hypersonic flow problems with strong shock waves. Finally, the framework is extended to unsteady problems for temporal error estimation [74, 75], in which a time-dependent adjoint problem must be solved, and a global temporal error estimate is divided into localized error contributions in the time domain and is used to guide an adaptive time-step refinement approach.

1.3 Thesis Overview

The objective of this work is to develop efficient solution strategies for high-order adaptive discontinuous Galerkin methods for steady-state and higher-order accurate time-implicit schemes, and to develop discrete adjoint sensitivity techniques for gradient-based shape optimization, output-based error estimation and mesh refinement algorithms, as well as demonstrating the performance of the methods in two-dimensional compressible flow test cases. The primary contributions of this thesis are as follows:

- Extension of solution methods for high-order steady-state discontinuous Galerkin discretizations to unsteady time-implicit problems, with emphasis on the development of a p-Multigrid approach driven by element-Jacobi solver variants and verification of temporal error convergence for various orders of implicit schemes.
- Derivation of a discrete adjoint sensitivity formulation for DG discretizations and application to shape optimization.
- Development of an adjoint-based spatial error estimation technique for steady-state flows.
- Investigation of h-refinement, p-refinement and hp-refinement schemes and application of an hp-adaptive refinement scheme to hypersonic flow problems.
- Extension of the adjoint-based spatial error estimation technique to temporal error estimation for unsteady flows, including the derivation of an unsteady discrete adjoint formulation for the first-order backwards difference scheme and the fourth-order implicit Runge-Kutta scheme in the context of discontinuous Galerkin discretizations.

While the contributions are intended to be general, this work is restricted to the application of these methods to discontinuous Galerkin discretizations of the compressible Euler equations. The outline of this thesis is illustrated in Fig. 1.1. Chapter 2 reviews the compressible Euler equations and details the DG discretization formulation, as well as the h-p multigrid approach for steady-state problems. In Chapter 3, the discrete adjoint formulation for computing sensitivity derivatives in shape optimization problems is derived using the steady-state DG discretization, with particular emphasis on the treatment of high-order curved surface elements and the resulting sensitivities in the design optimization problem. Chapter 4 develops and investigates the adjoint-based error estimation procedure and implements various mesh adaptation schemes, consisting of an h-refinement, a p-refinement and a combined hp-refinement schemes. Special attention is given to comparing the performance of the adaptation schemes with uniform mesh refinement methods, and to applying the application of hp-adaptation strategies for hypersonic flow problems. Chapter 5 focuses on the implementation of high-order time-implicit schemes in conjunction with high-order DG methods for achieving an overall high level of accuracy, with particular emphasis on the development of efficient implicit non-linear solvers. In Chapter 6, attention turns towards the extension of the adjoint-based spatial error estimation to unsteady problems, the first of which is an unsteady discrete adjoint formulation for the first-order backwards differencing scheme (BDF1) and the second of which is the unsteady adjoint method for the fourth-order implicit Runge-Kutta scheme (IRK4). Finally, conclusions and directions of future work are given in Chapter 7.



Discontinuous Galerkin Discretizations
Steady-State Flows
Unsteady Flows
Steady-State Solver
Sensitivity Analysis
Temporal Error Estimation
Time-Implicit Solver
Hierarchical Basis Functions
Shape Optimization
Spatial Error Estimation
Unsteady Adjoint Formulation
BDF1,BDF2, CN2 and IRK4 Schemes
Single Level Method
Discrete Adjoint Formulation
Discrete Adjoint Formulation
BDF1 Scheme
IRK4 Scheme
Jacobian Matrix Decomposition
Element-Jacobi Smoother Solvers
Sensitivity Derivatives
Funtional Error Prediction
Funtional Error Prediction
Single Level Method
hp-Multigrid Approach
Gradient-based Shape Optimization
h-, p-, and hp-adaptaion
Time-Step Adaptation
p-Multigrid Element-Jacobi
Results
Results
Results
Results
Results
Conclusions

Figure 1.1: Guide to this thesis.

Chapter 2

High-order Steady-state Discontinuous Galerkin Discretizations

This chapter presents high-order discontinuous Galerkin discretizations for steady-state non-linear hyperbolic systems of equations, represented by the two-dimensional compressible Euler equations. Efficient solution strategies consisting of an *hp*-multigrid approach driven by element Jacobi smoothers are investigated and developed to make the overall high-order methods competitive. Results show that a p-order DG discretization scheme achieves a $p+1$ rate of spatial error convergence for steady-state flow problems, demonstrating the favorable asymptotic accuracy properties of high-order discontinuous Galerkin methods. In addition, the *hp*-Multigrid approach exhibits p and h independent convergence rates in achieving steady-state solutions.

2.1 Governing Equations

The governing equations that we consider exclusively in this work are the two-dimensional compressible Euler equations that can be written in the following conservative form:

$$\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t} + \frac{\partial \mathbf{f}_1(\mathbf{u}(\mathbf{x},t))}{\partial x} + \frac{\partial \mathbf{f}_2(\mathbf{u}(\mathbf{x},t))}{\partial y} = 0 \quad \text{in } \Omega \tag{2.1}$$

where Ω is a two-dimensional bounded domain. The vector of conservative flow variables $\mathbf{u}$ and the inviscid Cartesian flux components $\mathbf{f}_1$ and $\mathbf{f}_2$ are defined by

$$\mathbf{u} = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho e \end{array} \right\}, \quad \mathbf{f}_1 = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e + p)u \end{array} \right\}, \quad \mathbf{f}_2 = \left\{ \begin{array}{c} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e + p)v \end{array} \right\} \tag{2.2}$$

respectively, where the notations ρ , p, and e denote the fluid density, pressure and specific total energy per unit mass, respectively. u and v represent the flow velocity components in the x and y coordinate directions. This system of equations is completed by the perfect gas equation of state given as,

$$p = (\gamma - 1)\left[\rho e - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{2.3}$$

where γ is defined as the ratio of specific heats, which is 1.4 for air.

2.2 Discontinuous Galerkin Discretizations

The computational domain Ω is partitioned by a triangulation $\mathcal{T}_H$, of average element size H, into an ensemble of non-overlapping elements, such that $\Omega = \bigcup_{k \in \mathcal{T}_H} \Omega_k$, where Ω_k refers to the volume of an element k ($k \in \mathcal{T}_H$). The discontinuous Galerkin discretization proceeds by formulating a weak statement of the governing equations, by multiplying Eq. (2.1) by a set of test functions, $\{\phi_j, j = 1, \cdots, M\}$, with the maximum polynomial order of p, and integrating within each element, e.g. k, as:

$$\int_{\Omega_k} \phi_j \left[\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}_1(\mathbf{u})}{\partial x} + \frac{\partial \mathbf{f}_2(\mathbf{u})}{\partial y}\right] d\Omega_k = 0 \tag{2.4}$$

Integrating this equation by parts, the weak statement of the problem becomes:

$$\begin{aligned} &\int_{\Omega_k} \phi_j \frac{\partial \mathbf{u}_H}{\partial t} d\Omega_k - \int_{\Omega_k} \left[\frac{\partial \phi_j}{\partial x}\mathbf{f}_1(\mathbf{u}_H) + \frac{\partial \phi_j}{\partial y}\mathbf{f}_2(\mathbf{u}_H)\right] d\Omega_k \\ &+ \int_{\partial\Omega_k \setminus \partial\Omega} \phi_j \mathbf{H}(\mathbf{u}_H^+, \mathbf{u}_H^-, \mathbf{n}) dS + \int_{\partial\Omega_k \cap \partial\Omega} \phi_j \mathbf{H}^b(\mathbf{u}_H^-(\mathbf{u}_H^+, \mathbf{n}), \mathbf{n}) dS = 0 \end{aligned} \tag{2.5}$$

where the unit normal vector $\mathbf{n} = (n_x, n_y)$ is outward to the boundary and points into the computational domain. $\mathbf{u}_H$ represents the Galerkin finite-element approximation for $\mathbf{u}$ and the notations of $\mathbf{u}_H^+$ and $\mathbf{u}_H^-$ refer to interior and exterior solution approximations at shared inter-element interfaces, respectively. Current implementations of the interior boundary flux function $\mathbf{H}(\mathbf{u}_H^+, \mathbf{u}_H^-, \mathbf{n})$ include the Riemann flux approximation of HLLC [45, 46] and Lax-Friedrichs [47]. For edges coinciding with physical boundaries of the computational domain, the approximate flux function, $\mathbf{H}^b(\mathbf{u}_H^-(\mathbf{u}_H^+, \mathbf{n}), \mathbf{n})$, is required to be explicitly dependent on exterior traces to satisfy the dual-consistency condition [76]. Here we set $\mathbf{H}^b(\mathbf{u}_H^-(\mathbf{u}_H^+, \mathbf{n}), \mathbf{n}) = \mathbf{f}_1(\mathbf{u}_H^-)n_x + \mathbf{f}_2(\mathbf{u}_H^-)n_y$, where $\mathbf{u}_H^-$ is determined by the interior flow approximation, $\mathbf{u}_H^+$, as well as the given boundary conditions. In particular, total temperature and pressure are prescribed on the inflow boundary and static pressure is prescribed on the outflow boundary, and at solid walls $\mathbf{u}_H^-$ is set to have the same density, total energy and tangential velocity $(u, v)_\parallel^t$ as $\mathbf{u}_H^+$, given by:

$$(u, v)_\parallel^t = (u^+, v^+)^t - (u^+ n_x + v^+ n_y)(n_x, n_y)^t \tag{2.6}$$

2.2.1 Solution Expansion and Geometry Mapping

The discrete solution in the local discontinuous Galerkin formulation can be expanded as a series of basis functions $\{\phi_i, i = 1, \cdots, M\}$ and solution expansion coefficients $\tilde{\mathbf{u}}_H$, written as:

$$\mathbf{u}_H(\mathbf{x}, t) = \sum_{i=1}^{M} \tilde{\mathbf{u}}_{H_i}(t)\ \phi_i(\mathbf{x}) \tag{2.7}$$

where M is the number of modes determined by the degree of the basis functions. Note that the set of test functions selected for the discrete form of Eq. (2.5) is identical to the set of basis functions, since this corresponds to a Galerkin method. In order to simplify the implementation of the subsequent spectral multigrid approach, the particular choice of the basis functions in the current work involves a set of hierarchical basis functions defined on a standard isoparametric triangle $\bar{\Omega}$ spanning between $\{0 < \xi, \eta < 1\}$ [16, 77]. To briefly summarize, the first-order Lagrange polynomials are defined as,

$$L_1 = 1 - \xi - \eta, \qquad L_2 = \xi, \qquad L_3 = \eta \tag{2.8}$$

then, the hierarchical basis set, $\{\phi\}$, is fully described by *vertex*,

$$\phi_1^v = L_1, \quad \phi_2^v = L_2, \quad \phi_3^v = L_3 \tag{2.9}$$

edge,

$$\begin{aligned} \phi_n^{e1} &= L_1 L_2 \psi_{n-2}(L_2 - L_1) \\ \phi_n^{e2} &= L_2 L_3 \psi_{n-2}(L_3 - L_2) \\ \phi_n^{e3} &= L_3 L_1 \psi_{n-2}(L_1 - L_3) \end{aligned} \tag{2.10}$$

and *bubble*,

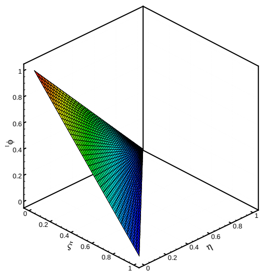
$$\phi_{n1,n2}^b = L_1 L_2 L_3 \psi_{n1-1}(L_2 - L_1) \psi_{n2-1}(L_1 - L_3) \tag{2.11}$$

shape functions, where $2 \leq n \leq p^e$, $n1 + n2 = p^b - 1$ and $n1, n2 \geq 1$. The kernel functions $\psi(z)$ are given as,

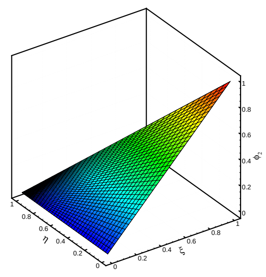
$$\psi_{n-2}(z) = \frac{-2\sigma}{n-1} P_{n-2}^{1,1}(z) \tag{2.12}$$

where $P_n^{\alpha,\beta}$ represents the Jacobi polynomial of order n, with weights α and β. In our discretization, the edge order p^e and the bubble order p^b, are set to be the discretization order within the element (i.e. $p^e = p^b = p$). For $p \geq 2$ the basis functions within the standard triangle, $\{\phi_i, i = 4 \dots M\}$, are normalized Lobatto functions [77] (i.e. $\phi_{n \geq 2} = \sigma \int_{-1}^{x} P_{n-1}^{0,0}(z) dz$) , which take zero values at the end of their definition interval. The normalization factor, σ, can be used to condition the mass or convection matrices. Fig. 2.1 illustrates the basis functions up to $p = 4$.

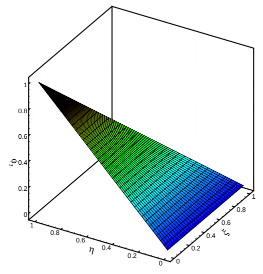
Since the set of basis functions is defined in the standard triangle, a coordinate mapping from the reference to a physical triangle is required for the computation of the first-order derivatives and integrals. The reference-to-physical transformation and the corresponding Jacobian associated with each element k are given by:



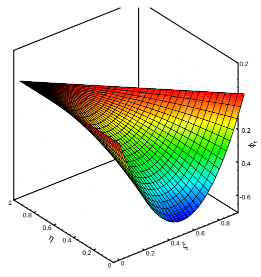
(a) ϕ_1



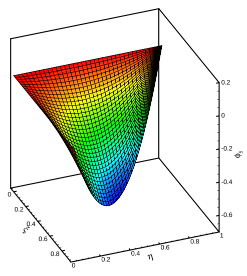
(b) ϕ_2



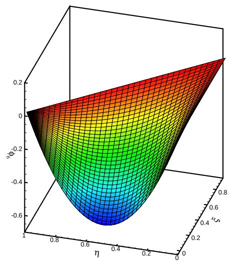
(c) ϕ_3



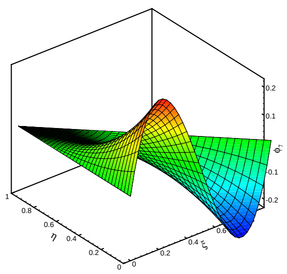
(d) ϕ_4



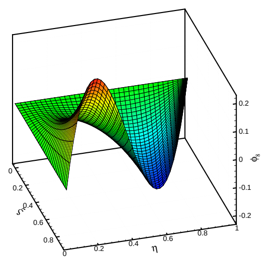
(e) ϕ_5



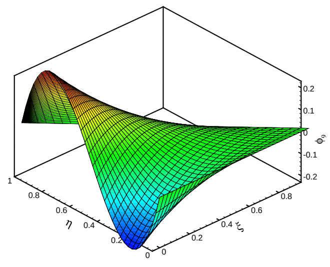
(f) ϕ_6



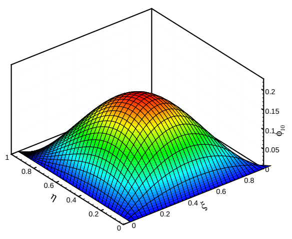
(g) ϕ_7



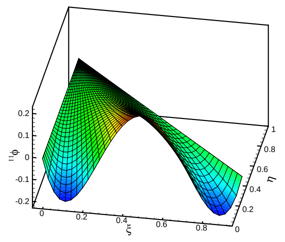
(h) ϕ_8



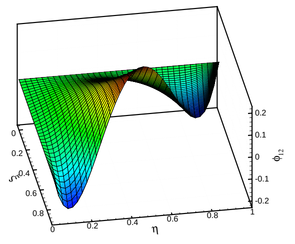
(i) ϕ_9



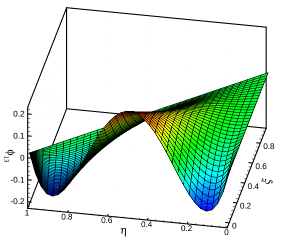
(j) ϕ_{10}



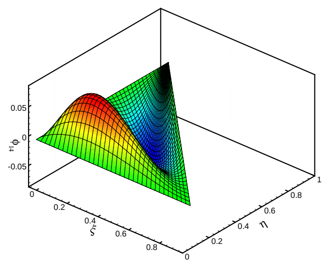
(k) ϕ_{11}



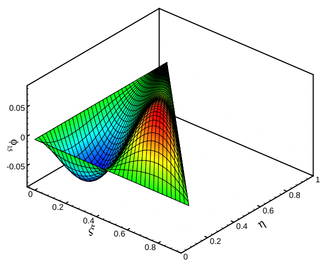
(l) ϕ_{12}



(m) ϕ_{13}

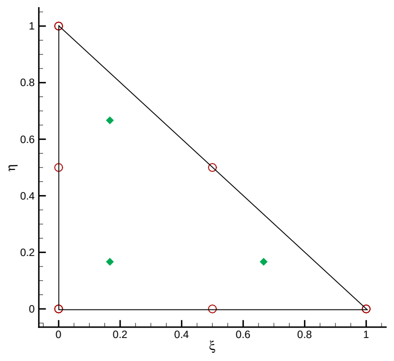


(n) ϕ_{14}

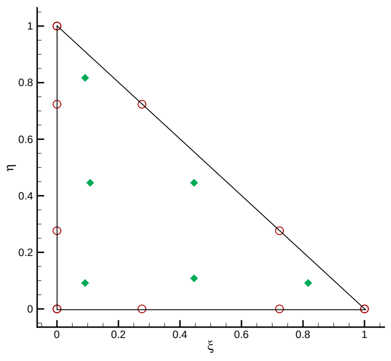


(o) ϕ_{15}

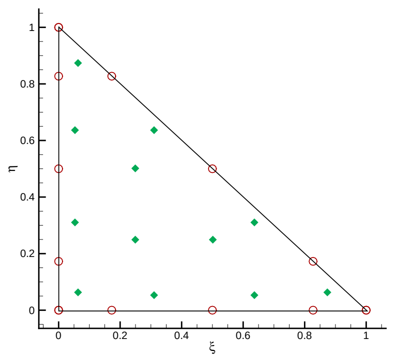
Figure 2.1: Illustration of the basis set used for high-order DG discretizations up to $p = 4$ (i.e. fifth-order accurate).



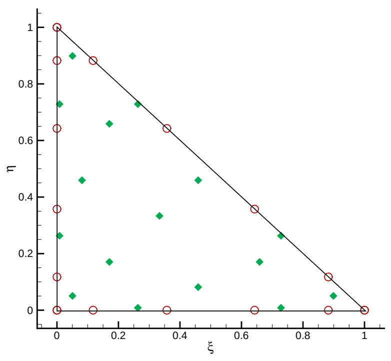
(a) $p = 1$



(b) $p = 2$



(c) $p = 3$



(d) $p = 4$

Figure 2.2: Volume and edge quadrature points on the standard triangle for various discretization orders. Green diamond symbols: Volume; Red circle symbols: Edge.

$$\mathbf{x}_k = \sum_{i=1}^{M} \tilde{\mathbf{x}}_{k_i} \phi_i(\xi, \eta) \qquad J_k = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{2.13}$$

In the simple case of straight-sided elements the transformation is linear thus the geometric mapping can be evaluated just by using the element vertex coordinates. However, in the more complex cases of high-order curved elements, which are typically required at curved boundaries, additional physical nodes [48] are required for determining the higher-order modes ($p > 1$) of the geometric mapping coefficients, $\tilde{\mathbf{x}}$, obtained by:

$$\tilde{\mathbf{x}}_k = \mathbf{\Phi}^{-1} \hat{\mathbf{x}}_{p_k} \tag{2.14}$$

where $\hat{\mathbf{x}}_{p_k} = \{\mathbf{x}_{c_k}, \mathbf{x}_{q_k}\}$ consists of the coordinates of element vertices, $\mathbf{x}_{c_k}$, as well as additional quadrature points, $\mathbf{x}_{q_k}$, for the element k. $\mathbf{\Phi}$ denotes the projection mapping matrix which is constituted by the basis functions evaluated at the corresponding points ($\boldsymbol{\xi}_k \leftarrow \hat{\mathbf{x}}_k$) in the reference triangle. The evaluation of the volume integrals in Eq. (2.5) is computed by use of Gauss quadrature rules [78, 79] which are exact for polynomial degree $2p$, and the surface integrals use Gauss-Legendre-Lobatto quadrature rules [77] which are exact for polynomial degree of $2p+1$ [16, 20]. Fig. 2.2 illustrates the volume and edge quadrature points on the standard triangle for various discretization orders ranging from $p = 1$ to $p = 4$. Neglecting the temporal derivative term (i.e. the first term) in Eq. (2.5), the spatially discretized system of equations for steady-state problems becomes:

$$\begin{aligned}\mathbf{R}_{Hj}(\tilde{\mathbf{u}}_H) = &-\int_{\Omega_k}\left[\frac{\partial\phi_j}{\partial x}\mathbf{f}_1(\tilde{\mathbf{u}}_{H_i}\phi_i)+\frac{\partial\phi_j}{\partial y}\mathbf{f}_2(\tilde{\mathbf{u}}_{H_i}\phi_i)\right]d\Omega_k+\int_{\partial\Omega_k\backslash\partial\Omega}\phi_j\mathbf{H}(\tilde{\mathbf{u}}^+_{H_i}\phi_i,\tilde{\mathbf{u}}^-_{H_i}\phi_i,\mathbf{n})dS\\ &+\int_{\partial\Omega_k\cap\partial\Omega}\phi_j\mathbf{H}^b(\mathbf{u}^-_H(\tilde{\mathbf{u}}^+_{H_i}\phi_i,\mathbf{n}),\mathbf{n})dS = 0 \qquad (j = 1, 2, \cdots, M)\end{aligned} \tag{2.15}$$

where $\mathbf{R}_H(\tilde{\mathbf{u}}_H) = \{\mathbf{R}_{Hj}(\tilde{\mathbf{u}}_H)\}$ represents the non-linear steady-state residual and the repeated index i implies summation over the range of 1 to M. Chapter 5 will revisit the spatially discretized equations of Eq. (2.5) for the solution of unsteady flow problems, where additional temporal discretizations will be considered.

2.2.2 Single Level Solution Method

In this section, we seek an efficient solution strategy for accelerating convergence of the non-linear equations (Eq. (2.15)) to steady state. We first describe a single level solution method including an approximate Newton linearization method and various element Jacobi smoothing schemes. Then, in the next section, the emphasis is placed on applying the same techniques used in the single level method to each level of a multigrid approach.

Element Jacobi Smoother

A Newton linearization scheme for solving the non-linear system (2.15) is taken as,

$$\tilde{\mathbf{u}}_H^{k+1} = \tilde{\mathbf{u}}_H^k - \left[\frac{\partial \mathbf{R}_H}{\partial \tilde{\mathbf{u}}_H}\right]^{-1} \mathbf{R}_H(\tilde{\mathbf{u}}_H^k) \tag{2.16}$$

where k refers to the iteration index. Instead of directly computing the inverse of the Jacobian matrix, $\left[\frac{\partial \mathbf{R}_H}{\partial \tilde{\mathbf{u}}_H}\right]$ in Eq. (2.16), we consider the use of approximate Newton schemes, consisting of a non-linear element Jacobi (NEJ), a quasi-non-linear element Jacobi (qNEJ), a linearized element Jacobi (LEJ) and a linearized element Gauss-Seidel smoother. First, the full Jacobian entries are decomposed into diagonal, $[D]$, and off-diagonal, $[O]$, block components, i.e. $[\partial \mathbf{R}_H / \partial \tilde{\mathbf{u}}_H] = [D] + [O]$. Each diagonal block corresponds to the modal coupling between all modes within a given element, which can be easily inverted using a lower-upper (LU) factorization at each block level and treated implicitly in the element Jacobi smoothers. The off-diagonal components arise from the inter-element flux computation between neighboring elements and are treaty explicitly in a given element Jacobi smoother. In particular, the present procedure for implementing the solver variants can be explained as follows:

- ▶ Outer iteration (with iteration index k): $\tilde{\mathbf{u}}_H^k$

  Inner iteration (with iteration index m, $m = 1, \cdots, \mathcal{M}$): $\hat{\mathbf{u}}_H^1 = \tilde{\mathbf{u}}_H^k$

  - ▷ NEJ : $\Delta\hat{\mathbf{u}}_H^{m+1} = [D]^{-1}_{\hat{\mathbf{u}}_H^m}[-\mathbf{R}_H(\hat{\mathbf{u}}_H^m)]$
  - ▷ qNEJ: $\Delta\hat{\mathbf{u}}_H^{m+1} = [D]^{-1}_{\tilde{\mathbf{u}}_H^k}[-\mathbf{R}_H(\hat{\mathbf{u}}_H^m)]$
  - ▷ LEJ : $\Delta\hat{\mathbf{u}}_H^{m+1} = [D]^{-1}_{\tilde{\mathbf{u}}_H^k}[-\mathbf{R}_H(\tilde{\mathbf{u}}_H^k) - [O]_{\tilde{\mathbf{u}}_H^k}\Delta\hat{\mathbf{u}}_H^m]$
  - ▷ LGS : $\Delta\hat{\mathbf{u}}_H^{m+1} = [\mathrm{D} + \mathcal{L}]^{-1}_{\tilde{\mathbf{u}}_H^k}[-\mathbf{R}_H(\tilde{\mathbf{u}}_H^k) - [\mathcal{U}]_{\tilde{\mathbf{u}}_H^k}\Delta\hat{\mathbf{u}}_H^m]$

  Model solution update: $\hat{\mathbf{u}}_H^{m+1} = \hat{\mathbf{u}}_H^m + \alpha\Delta\hat{\mathbf{u}}_H^{m+1}$

- ▶ Residual vector and diagonal/off-diagonal block updates:
  $\tilde{\mathbf{u}}_H^{k+1} = \hat{\mathbf{u}}_H^{\mathcal{M}}, \quad \mathbf{R}_H(\tilde{\mathbf{u}}_H^{k+1}), \quad [D]_{\tilde{\mathbf{u}}_H^{k+1}}, \quad [O]_{\tilde{\mathbf{u}}_H^{k+1}} \quad \text{or} \quad [O]_{\tilde{\mathbf{u}}_H^{k+1}} = [\mathcal{L} + \mathcal{U}]_{\tilde{\mathbf{u}}_H^{k+1}}$

- ▶ The procedure is performed repeatedly until $\mathbf{R}_H(\tilde{\mathbf{u}}_H^{k+1}) = 0$.

In the NEJ solver, the off-diagonal blocks are discarded, and the residual vector and diagonal components are updated using the latest available solution approximation at each inner iteration; The qNEJ approach represents a modified NEJ solver where the diagonal block Jacobians are frozen over a certain number of subiterations and thus only the steady-state residual is updated at each smoothing iteration. On the other hand, the LEJ method requires the storage of both diagonal and off-diagonal blocks, which are frozen in the inner iterations. The LGS smoothing strategy further splits the off-diagonal components of the Jacobian matrix into lower, $[\mathcal{L}]$, and upper, $[\mathcal{U}]$, contributions (denoted as $[O] = [\mathcal{L}] + [\mathcal{U}]$), treats $[\mathcal{U}]$ explicitly and $[\mathcal{L}]$ implicitly through forward substitution, and thus follows an ordered sweep across elements using the latest available neighboring information. Additionally, α is a relaxation parameter designed to keep $\|\alpha \Delta \tilde{\mathbf{u}}_H^{m+1}\|_{L_\infty} / \|\tilde{\mathbf{u}}_H^{m+1}\|_{L_\infty} \leq 10\%$. The number of smoothing passes (i.e. the inner iterations) is generally set equal to 5 in the current work. The performance of these solver variants for steady flow problems is demonstrated in Section 2.3.2. More detailed comparisons pertaining to unsteady flow problems are discussed in Chapter 5.

2.2.3 V-Cycle *p*-, *hp*-Multigrid Approach

Implementation of the element Jacobi smoother variants on the highest discretization order level results in a single-level solution method, which can be further accelerated by employing a V-cycle *p*- or *hp*-Multigrid approach [16–18,20,42,80,81]. This section is concerned with the use of efficient multigrid methods for solving the spatially discretized non-linear problem denoted in Eq. (2.15) to a final steady-state solution. Chapter 5 will reconsider this approach to obtain solutions for the corresponding unsteady problem for a series of arbitrarily large time-step sizes.

General Description

The motivation for the multigrid approach stems from an examination of the error of the numerical solution in the frequency domain. A multigrid scheme begins by eliminating the high-frequency errors associated with local variations in the solution on the fine grid, and then transferring the fine grid solution to a coarse grid, where the low-frequency errors of the fine grid manifest themselves as high-frequency errors and thus can be eliminated efficiently using the same smoothing method [82, 83]. Therefore, the basic idea behind all multigrid strategies is to accelerate the solution of a fine grid problem by computing corrections on coarser grids and then interpolating them back to the fine grid problem. Here we briefly summarize the principal procedure [82–84] to formulate a two-level multigrid scheme and this procedure can be used recursively for multiple levels of a multigrid scheme. First consider a discrete problem denoted as:

$$J_h u_h = f_h \tag{2.17}$$

where J_h represents the discretization of the continuous problem with a mesh size of h, and u_h represents the exact solution for the corresponding discretization scheme. If $\bar{u}_h$ refers to an approximation to the exact solution solved by using an iterative scheme, then Eq. (2.17) becomes:

$$J_h \bar{u}_h - f_h = r_h \tag{2.18}$$

Since $\bar{u}_h$ does not satisfy the discrete problem exactly, we have a residual term, r_h. If $r_h = 0$, then $\bar{u}_h = u_h$. Subtracting Eq. (2.18) from Eq. (2.17), we obtain:

$$J_h u_h - J_h \bar{u}_h = -r_h \tag{2.19}$$

If J_h denotes a linear operator, Eq. (2.19) can be written as:

$$J_h(u_h - \bar{u}_h) = -r_h \quad \text{or} \quad J_h v_h = -r_h \tag{2.20}$$

where v_h represents the required solution correction, $u_h - \bar{u}_h$, on the fine mesh. Assuming high frequency errors in the solution have been eliminated by using an efficient local solution or error smoothing strategy, we next transfer the solution correction to a coarse mesh. Thus the problem on the coarse mesh reads:

$$J_H v_H = -I_h^H r_h \tag{2.21}$$

where v_H denotes the solution corrections on the coarse mesh, obtained by solving Eq. (2.21) either exactly or approximately. I_h^H refers to the restriction operator for transferring the residual vector from a fine mesh to a coarse mesh. The fine level solution is then updated as:

$$u_h^* = \bar{u}_h + I_H^h v_H \tag{2.22}$$

where I_H^h refers to the prolongation operation for interpolating the solution corrections from the coarse mesh back onto the fine mesh, and u_h^* denotes new solution updates.

If the operator J_h is non-linear, $J_h u_h - J_h \bar{u}_h$ is no longer equal to $J_h(u_h - \bar{u}_h)$ and thus Eq. (2.20) is not valid. To modify the above multigrid scheme, a new variable on the coarse mesh is introduced, written as:

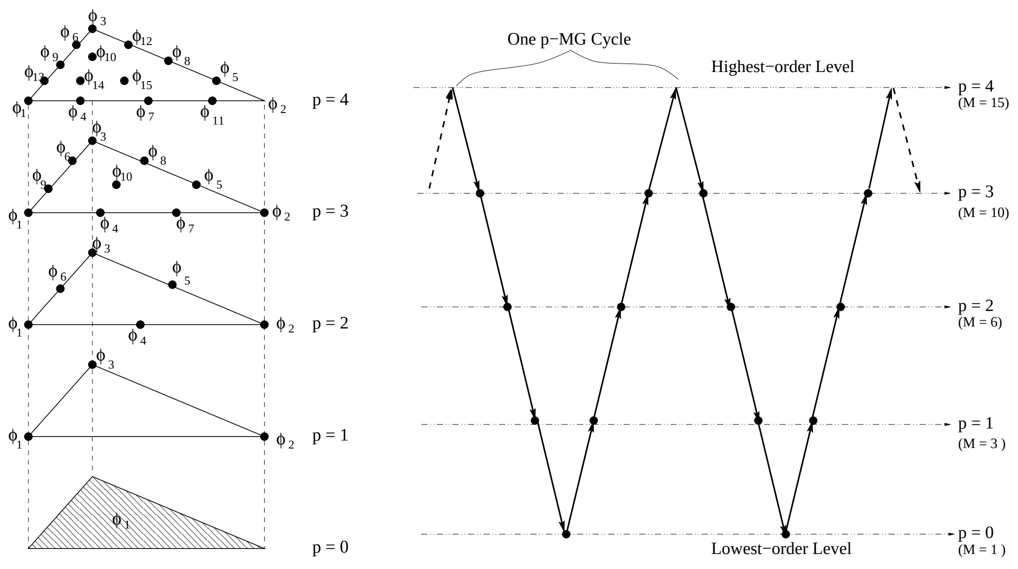
$$\bar{u}_H = \bar{I}_h^H \bar{u}_h + v_H \tag{2.23}$$

where $\bar{I}_h^H$ denotes an operator which restricts the solution from the fine mesh to the coarse mesh. Note that, in the traditional h-Multigrid approach [82, 83], this operator may be distinct from the aforementioned restriction residual operator I_h^H which transfers the fine level residual variables to the coarse mesh, whereas in the spectral or p-Multigrid approach, these operators may be identical. The coarse grid problem equivalent to Eq. (2.21) is thus expressed as:

$$J_H \bar{u}_H = S_H \quad \text{where} \quad S_H = J_H \bar{I}_h^H \bar{u}_h - I_h^H r_h \tag{2.24}$$

where S_H is often called the defect correction for the coarse grid problem. $\bar{u}_H$ is obtained by solving Eq. (2.24) either exactly or approximately on the coarse mesh. Then the fine level solution is updated as:

$$u_h^* = \bar{u}_h + I_H^h(\bar{u}_H - \bar{I}_h^H \bar{u}_h) \tag{2.25}$$



(a) (b) M, number of modes

Figure 2.3: Illustration of a five level V-cycle p-Multigrid algorithm.

This constitutes one two-level multigrid cycle and can be used recursively on multiple coarser levels. Moreover, the multigrid solution procedure is performed repeatedly until u_h^* satisfies Eq. (2.17). The current work makes use of the non-linear multigrid approach and concentrates on the use of a spectral multigrid (i.e. p-Multigrid) strategy [17,18,20,81] or a combined spectral-and-geometric multigrid (i.e. hp-Multigrid) approach [16]. More details are given in the following sections.

p-Multigrid

The p-Multigrid algorithm is based on the standard geometric multigrid method [84], but instead of using physically fewer elements on coarser levels, lower-order approximations serve as "coarse" levels while the same spatial grid elements are used on all levels, as illustrated in Fig. 2.3(a). Therefore, in this methodology, no additional grid information needs to be stored. Fig. 2.3(b) shows an example of a five-level V-cycle p-Multigrid approach. Here, the highest level consists of the original $p = 4$ spatial discretization, while the intermediate levels consist of lower $p = 3$, $p = 2$ and $p = 1$ discretizations, while a $p = 0$ (i.e. first-order accurate) scheme is employed at the lowest level. The procedure of a simplified two-level V-cycle p-Multigrid scheme used for solving the spatially discretized Euler equations (2.15) is described as follows:

- **Step 1:** Perform $\mathcal{M}$ subiterations on the high-order approximation level (p) to solve the problem: $\mathbf{R}_p(\tilde{\mathbf{u}}_p^m) = \mathbf{S}_p$, using one of the element Jacobi smoothers described previously; Calculate residual: $\mathbf{r}_p = \mathbf{S}_p - \mathbf{R}_p(\tilde{\mathbf{u}}_p^{\mathcal{M}})$.

- **Step 2:** Restrict both solution and residual to the low-order approximation level ($p-1$): $\tilde{\mathbf{u}}_{p-1} = \bar{I}_p^{p-1}\tilde{\mathbf{u}}_p^{\mathcal{M}}$; $\mathbf{S}_{p-1} = \mathbf{R}_{p-1}(\tilde{\mathbf{u}}_{p-1}) + I_p^{p-1}\mathbf{r}_p$.

- **Step 3:** Solve the low-order approximation level problem, $\mathbf{R}_{p-1}(\tilde{\mathbf{u}}_{p-1}^m) = \mathbf{S}_{p-1}$, by using the same element Jacobi solver selected on the previous level with $\mathcal{M}$ subiterations; Obtain the low-order level error: $\mathbf{e}_{p-1} = \tilde{\mathbf{u}}_{p-1}^{\mathcal{M}} - \bar{I}_p^{p-1}\tilde{\mathbf{u}}_p^{\mathcal{M}}$.

- **Step 4:** Prolongate this low-order level error to correct the high-order approximation level: $\tilde{\mathbf{u}}_p = \tilde{\mathbf{u}}_p^{\mathcal{M}} + I_{p-1}^p\mathbf{e}_{p-1}$.

- **Step 5:** Repeat steps 1 through 4 for each V-cycle of the p-Multigrid method until $\mathbf{R}_p(\tilde{\mathbf{u}}_p^{\mathcal{M}}) = 0$ (machine zero or a suitably determined tolerance).

The replacement of the notation $\mathbf{R}_H$ with $\mathbf{R}_p$ is denoted to specify the approximation level of the p-Multigrid approach. The source term $\mathbf{S}_p$, which represents the residual restriction term from the finer multigrid levels, vanishes on the finest level (highest-order approximation level) in the multigrid formulation, but is retained on all levels in the description for generality. The p-multigrid approach fits naturally with the present DG discretizations since the use of a hierarchical basis set simplifies the formulation of interpolation operators between high-order and low-order approximation levels. In particular, the restriction operator (I_p^{p-1}) from high-order levels to low-order levels is obtained by simply deleting the corresponding higher order modal coefficients and transferring lower order modes exactly. The prolongation operator (I_{p-1}^p) from lower-order to high-order levels is obtained by injecting lower order modal coefficients exactly. The main reason for this simple projection is due to the fact that the lower order basis functions are a subset of the higher order basis functions. Moreover, the same restriction operator (i.e. $I_p^{p-1} = \bar{I}_p^{p-1}$) is used for both residual and solution restriction in the p-Multigrid scheme, although this is not valid in the h-Multigrid method described subsequently. The two-level multigrid scheme described above is used recursively to solve the coarse level problem, resulting in the full multilevel algorithm. Furthermore, a constant number of five smoothing sweeps (i.e. $\mathcal{M} = 5$) is employed at all multigrid levels using one of the element Jacobi smoothers. For the qNEJ, LEJ and LGS smoothing schemes, this requires only one non-linear Jacobian (and non-linear residual for the LEJ and LGS schemes) evaluation for each visit to a given mesh level.

hp-Multigrid

For relatively fine meshes, the coarse problem at the $p = 0$ level of each multigrid cycle can be expensive to solve and inadequate solution of the $p = 0$ sub-problem may result in deterioration of the overall convergence rate of the multigrid approach. In order to solve the $p = 0$ sub-problem more accurately, the traditional h-Multigrid approach [82–84], which makes use of a sequence of physically coarser agglomerated grids, can be utilized once the $p = 0$ approximation level has been reached (as illustrated in Fig. 2.4), thus constituting the hp-Multigrid approach. For the hp-Multigrid approach, five smoothing passes on each level are still used for the p-Multigrid scheme. However, note that different operators for restricting the residual and solution variables are required. In particular, the restriction of the solution variables and the residual to a coarse level of the h-Multigrid algorithm is obtained as [85]:

$$\mathbf{u}_H = \bar{I}_h^H \mathbf{u}_h = \frac{1}{A_H} \sum_{k=1}^{N_k} (\mathbf{u}_h A_h)_k \quad \text{and} \quad \mathbf{R}_H = I_h^H \mathbf{R}_h = \sum_{k=1}^{N_k} (\mathbf{R}_h(\mathbf{u}_h))_k \tag{2.26}$$

where N_h is the number of elements used in an individual agglomeration cell between two grid levels, A_h refers to the fine level elemental area, and $A_H = \sum_{k=1}^{N_k} (A_h)_k$ is the coarse level area.

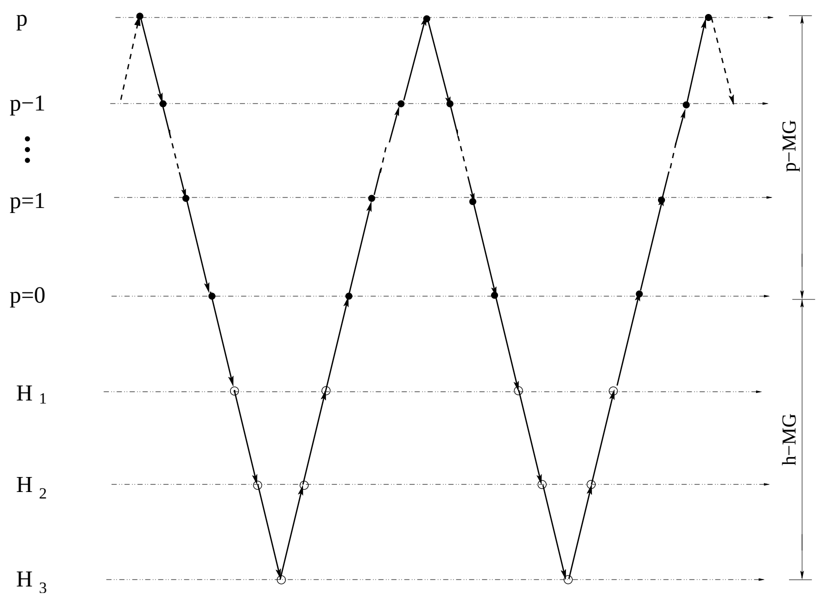


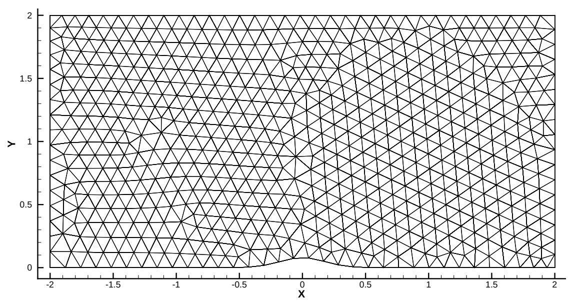
Figure 2.4: Illustration of a V-cycle *hp*-Multigrid algorithm.

2.3 Steady-state Results

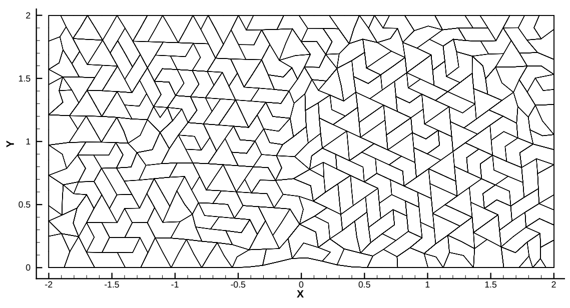
This section presents results of spatial accuracy and algorithm performance for a smooth two-dimensional compressible channel flow problem. The *hp*-Multigrid algorithm described above is employed to accelerate convergence of the flow solver to a steady-state solution, where a constant 5 smoothing iterations are used on each level of the *hp*-Multigrid approach, including the agglomerated ($p = 0$) levels.

2.3.1 Subsonic Channel Flow over a Gaussian Bump

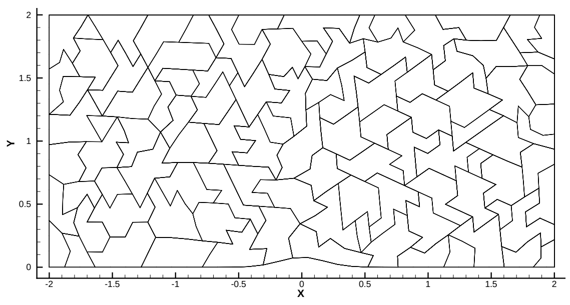
The computational domain and an intermediate mesh with a mesh size of $N = 1248$ are depicted in Fig. 2.5(a). This test case involves the compressible flow with freestream Mach number of 0.35 passing over a Gaussian bump perturbation. The boundary nodes are generated using the bump geometry configuration, given by: $y = \frac{1}{5\sqrt{2\pi}}e^{-16(x-x_0)^2}$. Wall boundary conditions are enforced on the top and bottom of the channel and the initial condition is set to a steady-state solution obtained with the $p = 0$ scheme. The approximate Riemann flux function of HLLC [45, 46] is used to resolve discontinuities at inter-element boundaries and the spatial residual is con-



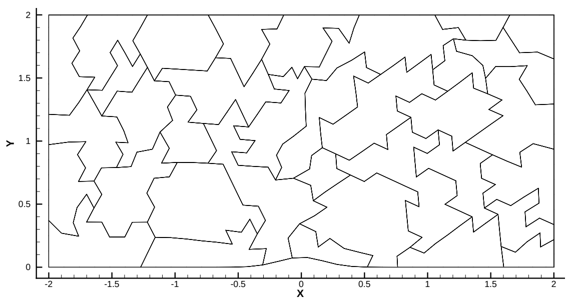
(a) H_0



(b) H_1



(c) H_2



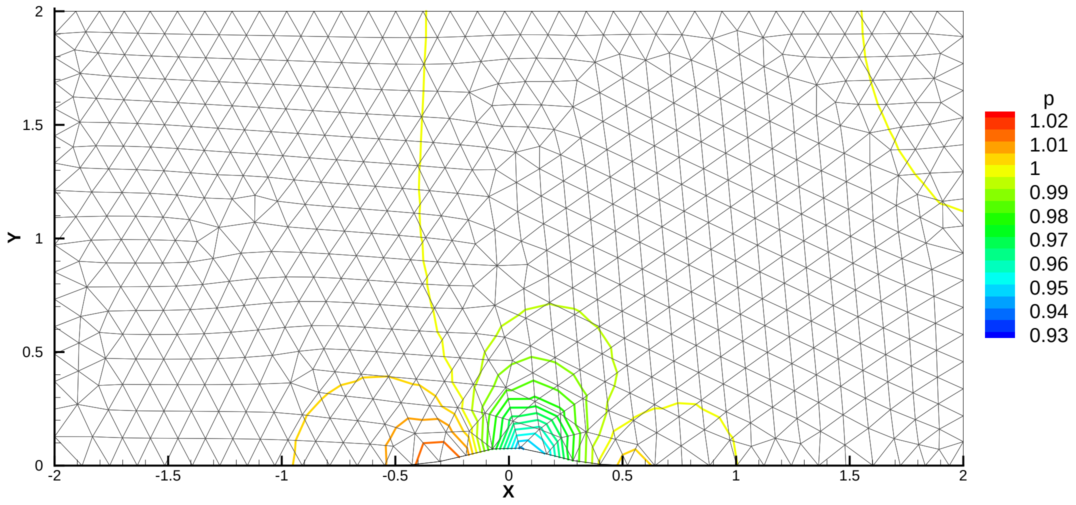
(d) H_3

Figure 2.5: Illustration of four levels of the h-Multigrid agglomerated mesh configuration for a mesh size of $N = 1248$ in the two-dimensional channel flow test case.

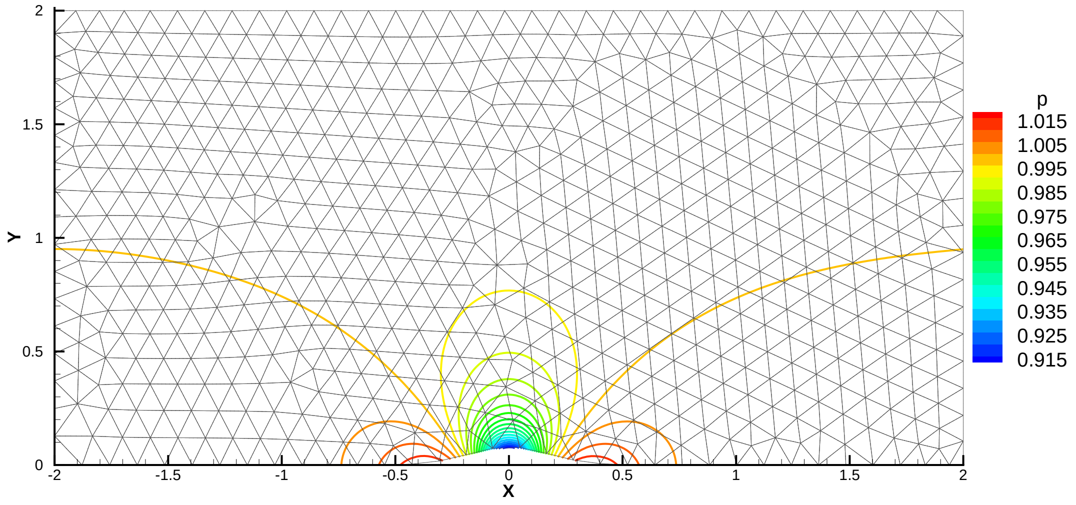
verged to machine precision of 10^{-14} for all cases. Fig. 2.5 illustrates four levels of the h-Multigrid agglomerated mesh configuration for the intermediate mesh ($N = 1248$).

Numerical Solutions

Figures 2.6(a) and 2.6(b) illustrate the pressure contours of the steady-state solution using the $p = 0$ and $p = 4$ spatial discretizations, respectively, on the mesh of 1248 elements. Due to the low mesh resolution on the top of the bump surface, the $p = 0$ discontinuous Galerkin scheme, which corresponds to a first-order accurate finite volume scheme, fails to obtain a smooth solution for this channel flow test case. Based on the same mesh resolution, the $p = 4$ scheme, in contrast, demonstrates the superior ability of a higher-order scheme in delivering a very smooth and symmetric solution for this test case.



(a) First-order accurate scheme with $p = 0$ boundary elements



(b) Fifth-order accurate scheme with $p = 4$ boundary elements

Figure 2.6: Computed pressure contours using the $p = 0$ and $p = 4$ schemes on a unstructured triangular mesh of 1248 elements.

Spatial Accuracy

A series of four grids, consisting of 573, 1248, 2522 and 5088 unstructured triangular elements, have been employed to study solution error convergence of the discontinuous Galerkin discretizations of various orders, ranging from $p = 0$ to $p = 3$. The standard L_2 norm of the entropy error, $\|e_s\|_{L_2} = \|s - s_\infty\|_{L_2}$, where s_∞ denotes the free-stream entropy, is used to measure the discretization error. Fig. 2.7 depicts the L_2 entropy error norm convergence of the steady-state solution for various discretization orders as a function of grid spacing (h). For two-dimensional configurations, the number of elements in the computational domain is inversely proportional to the square of the grid spacing (i.e. $N \sim 1/h^2$). Fig. 2.7 indicates that the optimal error convergence $(\sim h^{p+1})$ is approximately attained in this study. Specifically, the asymptotic slopes of the $p = 0, p = 1, p = 2$ and $p = 3$ schemes are 0.7, 1.7, 3.0, and 3.8, respectively, which are close to the respective design values of 1, 2, 3, and 4.

Efficiency Comparison

The results of the mesh refinement study discussed in the section on spatial accuracy are re-explored in Fig. 2.8 for an efficiency study, where the discretization error is plotted as a functional of the required CPU time, which is obtained using the V-cycle *hp*-Multigrid algorithm driven by the LGS smoothing strategy. The computational cost for the $p = 0$ scheme is not included in the figure since the converged $p = 0$ solutions serve as the initial conditions for higher-order schemes. The spatial residuals of all discretization schemes are fully converged to machine precision of 10^{-14}, so that the relative comparisons between CPU time for the various discretizations remain valid. As illustrated in Fig. 2.8, to achieve a given level of accuracy, the CPU time decreases when the discretization order is increased, and the benefits of using higher-order schemes become more significant with higher levels of accuracy.

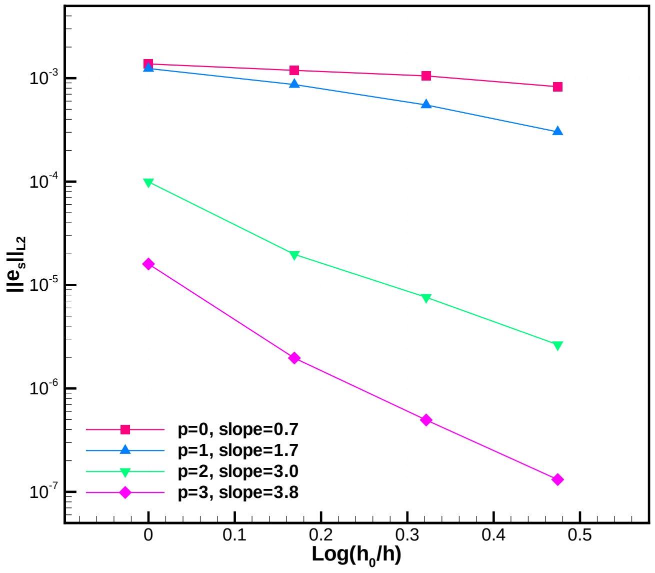


Figure 2.7: Convergence of the L_2 norm of the entropy error for various discretization orders ($0 \leq p \leq 3$) as a function of grid spacing in the case of a 2D compressible channel flow. h_0 represents the grid spacing of the coarsest mesh.

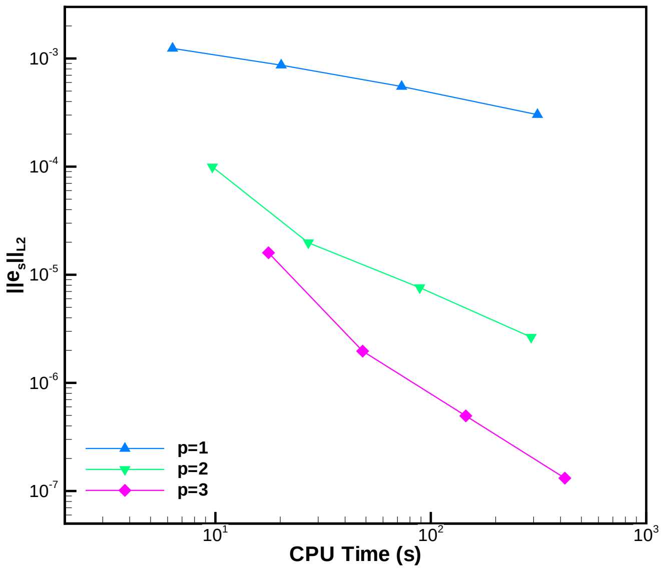


Figure 2.8: Convergence of the L_2 norm of the entropy error for various discretization orders ($1 \leq p \leq 3$) as a function of computational cost in the case of a 2D compressible channel flow.

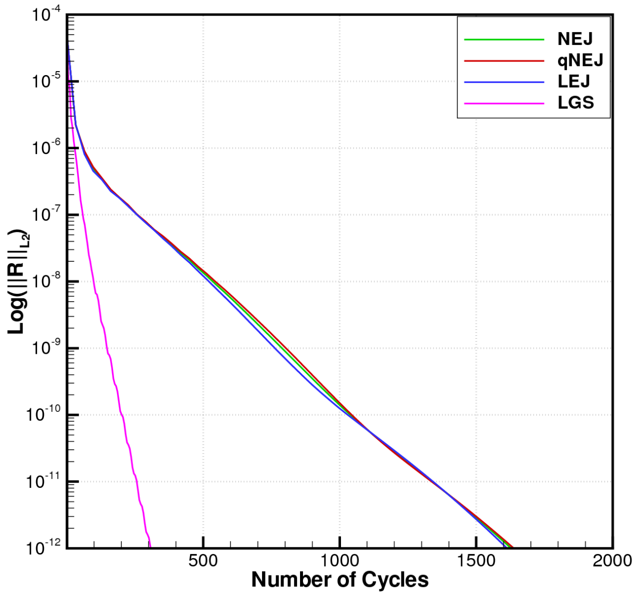
2.3.2 Performance of the *hp*-Multigrid Approach

As mentioned previously, the use of efficient solution strategies is particularly important for the high-order DG discretizations to deliver steady-state flow solutions in an efficient manner. Therefore, this section is concerned with demonstration of the performance of the element-Jacobi solver variants and demonstration of the superior efficiency of the *hp*-Multigrid approach over the single level method.

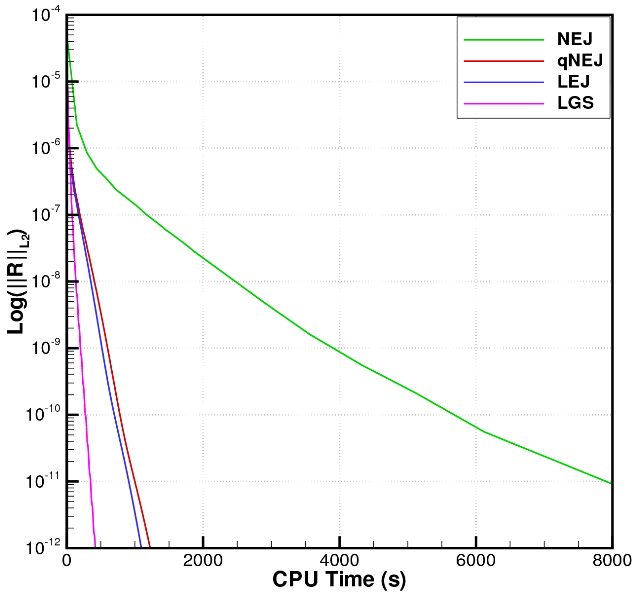
Comparison of Element Jacobi Smoothers

The performance of the non-linear element Jacobi (NEJ), the quasi-non-linear element Jacobi (qNEJ), the linearized element Jacobi (LEJ) and the linearized element Gauss-Seidel (LGS) smoothing strategies is demonstrated using a $p = 4$ spatial discretization and the intermediate mesh with a mesh size of $N = 1248$ for the 2D channel flow problem. The single level method described in Section 2.2.2 is used for all runs and the convergence is measured in terms of overall number of cycles. Recall that the qNEJ, LEJ and LGS smoothers effectively freeze the non-linear Jacobians (and residuals for the LEJ and LGS schemes) over five smoothing iterations, as opposed to the NEJ smoother where both Jacobians and residuals are required to be evaluated at each iteration.

Fig. 2.9(a) depicts the convergence profiles of the NEJ, qNEJ, LEJ and LGS solvers in terms of overall number of cycles. The results are consistent with those obtained in Reference [85]: the NEJ, qNEJ and LEJ solvers converge at similar rates in terms of numbers of cycles, while the LGS delivers a significantly faster convergence rate. When compared in terms of computational cost, as shown in Fig. 2.9(b), the LGS, LEJ and qNEJ schemes are seen to be substantially more efficient than the NEJ scheme. This is because the linearized schemes utilize five iterations between non-linear updates, and therefore result in five times fewer non-linear Jacobian and residual evaluations than the non-linear element Jacobi scheme. The qNEJ scheme is shown to be similar in terms of computational efficiency as the LEJ scheme, however, since the off-diagonal blocks of the Jacobian matrix are not stored in the qNEJ



(a)



(b)

Figure 2.9: Comparison of convergence of non-linear element-Jacobi (NEJ), quasi-non-linear element-Jacobi (qNEJ), linearized element-Jacobi (LEJ) and linearized element Gauss-Seidel (LGS) smoothers on a mesh size of $N = 1248$ and discretization order of $p = 4$. (a) Convergence vs. number of cycles. (b) Convergence vs. computational cost.

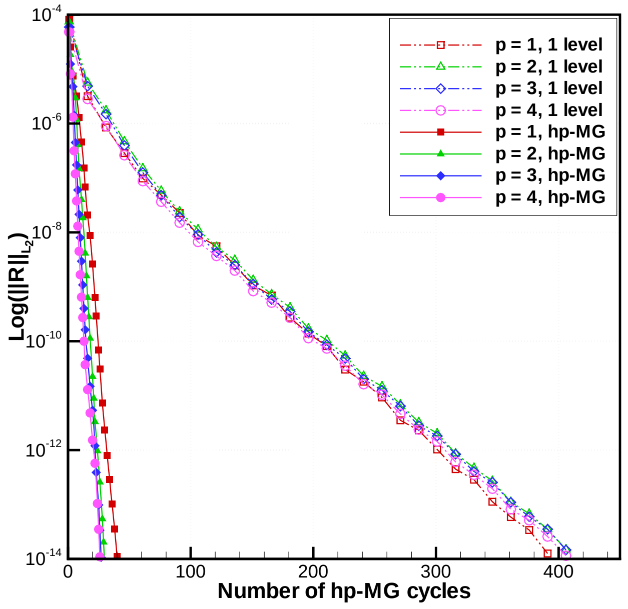
scheme, the qNEJ scheme may be an appropriate compromise for cases where memory limitations are dominant.

Efficiency of the *hp*-Multigrid Approach

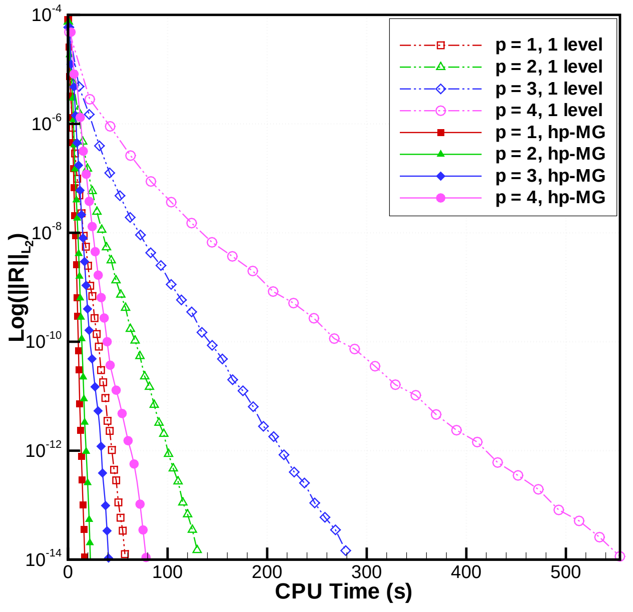
First, the performance of the *hp*-Multigrid scheme as a function of discretization order is considered, and then the behavior of the *hp*-Multigrid scheme for various mesh sizes is examined. Specifically, the mesh sizes vary from $N = 573$, $N = 1248$, $N = 2522$ and $N = 5088$, and discretization orders range from $p = 1$ to $p = 4$, for the two-dimensional compressible channel flow problem. The linearized element Gauss-Seidel smoother with 5 smoothing passes is used at each *hp*-Multigrid level in the following test cases.

Fig. 2.10(a) illustrates the *hp*-Multigrid convergence for various discretization orders, compared with the single level solver on a fixed mesh size of $N = 1248$. Due to the fact that the mesh size has been fixed for all schemes, this figure illustrates the effect of discretization order on convergence rate. Clearly, both single level solver and *hp*-Multigrid approach yield a convergence rate which is independent of the order of discretization for a fixed grid size (i.e. *p*-independence), and the *hp*-Multigrid approach delivers an overall much faster convergence rate than the single level method. Fig. 2.10(b) shows that the computational cost for a higher-order discretization scheme increases progressively using the single level method. For example, to obtain a level of 10^{-14}, the computational cost required by the $p = 4$ scheme in the single level method is roughly 4.3 and 5.6 of those required by the $p = 2$ and $p = 1$ schemes. On the other hand, although an *hp*-Multigrid cycle requires more computational time than a single level cycle, the *hp*-Multigrid approach is still shown to be significantly more efficient than the single level method in terms of computational cost, and the benefit of the *hp*-Multigrid solver can be expected to be more evident for higher approximation orders and larger meshes.

Since mesh resolution is another factor which affects convergence and computational time, the behavior of the *hp*-Multigrid approach and corresponding single level solver is examined with repsect to various mesh sizes. The convergence his-

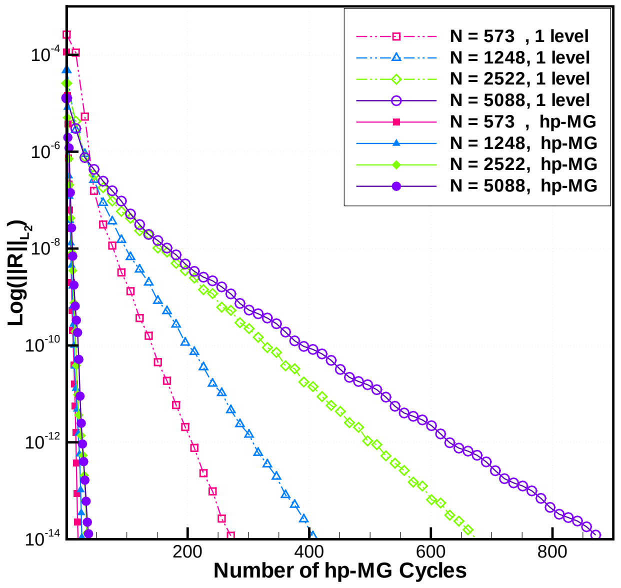


(a)

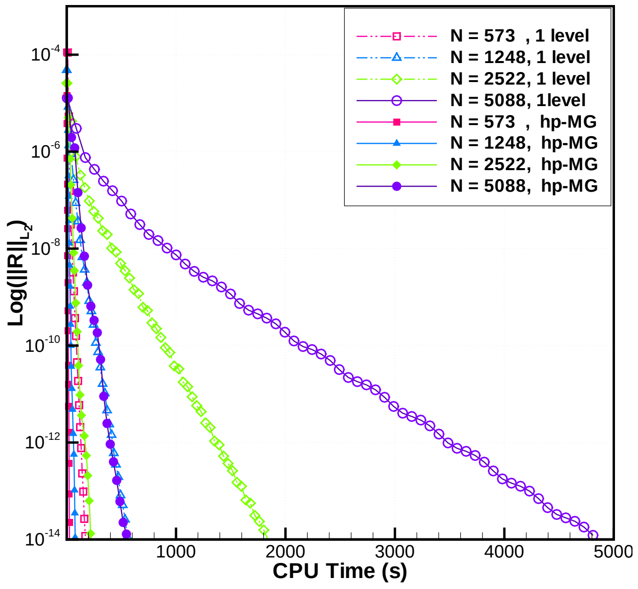


(b)

Figure 2.10: Comparison of convergence for the *hp*-Multigrid and the single level solvers, using various discretization orders on a fixed mesh size of $N = 1248$, for the 2D channel flow test case. (a) Convergence history vs. the number of *hp*-Multigrid cycles. (b) Convergence history vs. computational cost.



(a)



(b)

Figure 2.11: Comparison of convergence for the *hp*-Multigrid and the single level solvers, using various mesh sizes of $N = 573$, $N = 1248$, $N = 2522$ and $N = 5088$ and a fixed discretization order of $p = 4$, for the 2D channel flow test case. (a) Convergence history vs. the number of *hp*-Multigrid cycles. (b) Convergence history vs. computational cost.

tories obtained for various mesh sizes are illustrated in Fig. 2.11 using the fixed discretization order of $p = 4$ for all runs. In terms of number of cycles, increasing the number of elements, N, has an adverse effect on the convergence rate of the single level solver, showing that the element Gauss-Seidel solver is in fact h-dependent. However, the hp-Multigrid solver is seen to provide essentially h-independent convergence rates, achieving convergence to 10^{-14} within 40 multigrid cycles for all meshes. Fig. 2.11(b) shows the computational costs for this test case. It is clearly shown that the hp-Multigrid solver provides the most efficient mechanism for solving the discrete steady-state problem, and this advantage can be expected to increase for finer meshes. For example, using the mesh size of $N = 573$ and the fixed discretization order of $p = 4$, the hp-Multigrid approach achieves a speedup of 6.5 over the single level solver, while a speedup of 8.4 is obtained for the mesh size of $N = 5088$.

Chapter 3

Derivation of Discrete Adjoint Formulations and Application to Shape Optimization

In this chapter, a discrete adjoint-based sensitivity analysis technique is developed and investigated based on the steady-state discontinuous Galerkin discretization. This approach is well-suited for many applications in computational fluid dynamics, such as shape optimization and error estimation. Here a framework for design-optimization problems using an adjoint-based sensitivity formulation is first established. Chapter 4 and Chapter 6 will further explore the application of adjoint methods for error estimation and adaptive refinement strategies for steady-state and unsteady flow problems.

3.1 Introduction

In aerodynamic design optimization techniques, most efficient optimization strategies are gradient based [21,63,76,86–89], where the vehicle shape is parametrized by a set of design variables, **D**, and an optimum achievable shape is obtained by minizing an appropriate cost functional, L, such as drag coefficients, target pressure distributions, lift to drag ratios, etc. Much focus of this gradient-based optimization is placed on obtaining the sensitivity derivatives of the cost functional with respect to the design variables, $dL/d\mathbf{D}$, which provides the search directions for optimization. Several approaches may be useful for obtaining the sensitivity derivatives. The most straightforward approach is via the finite-difference method, in which each component of the gradients, dL/dD_m, is evaluated individually by varying the value of D_m by a small amount and recomputing the value of the objective functional based on this effect. However, the sensitivity result obtained by the finite-difference method is very sensitive to the selected perturbation size, and moreover each gradient component requires one extra analysis solution, and thus this method results in an inefficient approach when there is a large number of design parameters. An alternate approach to obtain objective sensitivities in a more accurate manner is via the tangent method, in which a linearization of the analysis problem is directly taken based on the analysis code, thus yielding accurate sensitivity results. However, the evaluation of each sensitivity derivative requires the solution of a linear system, which again becomes impractical when a large number of design variables is involved. A more popular approach for this task involves the use of an adjoint method [23, 64, 86, 89], in which all the sensitivity derivatives relevant to one objective functional are obtained by solving one flow problem and one adjoint problem, thus the total cost of this method is essentially independent of the number of design variables, leading to an efficient approach when there is a large number of design parameters while only a single or a small number of objectives. The solution of the adjoint problem should exhibit a similar convergence rate as the primary or flow problem due to the fact that both problems contain the same eigenvalues. However, the linear nature of the adjoint problem simplifies the solution procedure and often results in the cost of the adjoint problem being lower than the cost of the flow problem.

There are two main approaches for formulating the adjoint problem, namely the continuous approach and the discrete adjoint approach. In the continuous adjoint approach, the governing equations are first linearized and then discretized, while in the discrete adjoint approach the same steps are performed but in reverse order. Since the linearization of the discretized equations for the primal (i.e. flow) problem has been already performed for the element Jacobi solvers, the current work concentrates on constructing objective gradients based on the discrete adjoint approach for shape sensitivity problems.

The remainder of this chapter is organized as follows. In Section 3.2, some preliminary descriptions, including shape parametrization, surface and mesh deformation and objective or cost functionals, are first given. Then the discrete adjoint-based sensitivity derivatives are formulated in the context of discontinuous Galerkin methods. In Section 3.3, the shape optimization procedure used in this work is described. Computational results are shown in Section 3.4 to demonstrate the performance of the adjoint approach in a typical design problem.

3.2 Sensitivity Calculation

The adjoint sensitivity formulation starts with the formulation of the tangent problem, in which the discretized governing equations are linearized and the sensitivity derivatives of an objective functional are formulated. The discrete adjoint formulation is then derived by transposing each matrix of the tangent problem and performing the operations in reverse order. A detailed derivation with references to related approaches can be found in [89] and therefore our emphasis in this section is to give a summary and to extend the basic idea to discontinuous Galerkin discretizations on meshes involving curved elements.

3.2.1 Preliminary Description

Design Variables and Shape Parametrization

Assume that an initial shape geometry, represented by the coordinates of boundary nodes $\mathbf{x}_s$ and possibly additional surface quadrature points $\mathbf{x}_q$ for curved elements, and a computational mesh are given. The surface geometry is modified through surface node displacements which are determined by a set of design variables, $\mathbf{D} = \{D_m, m = 1, \cdots, N_d\}$ (where N_d represents the total number of design variables). The Hicks-Henne sine bump function [90] is employed in order to ensure smooth

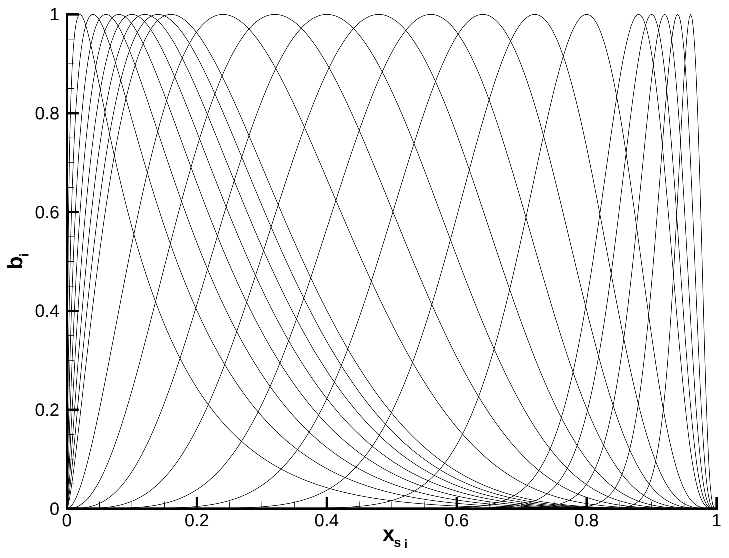


Figure 3.1: Illustration of the Hicks-Henne bump functions for various x_{sm} parameters varying from 0.02 to 0.96.

designed surface shapes, and the design variables are set to be the magnitudes of the bump functions placed in the normal directions of the surface nodes, expressed as:

$$b_i(x_{si}, D_m) = D_m sin^4(\pi {x_{si}}^{ln\frac{1}{2}/lnx_{sm}}), \quad x_{si}, x_{sm} \in [0,1] \tag{3.1}$$

where x_{sm} denotes the x-coordinate of the surface node where the bump function is placed. b_i denotes the surface node displacement at x_{si} due to the displacement of the surface node at x_{sm}, and D_m is the m^{th} component of the design variables associated with the surface node at x_{sm}. Fig. 3.1 depicts a sample plot of the bump functions with a magnitude of 1, as x_{sm} takes on a range of 0.02 to 0.96. For multiple design variables ($N_d > 1$), the surface displacement at a particular node location of x_{si} is the superposition of all the bump functions placed at each designed surface node, and new surface coordinates are computed based on all the surface displacements, denoted as:

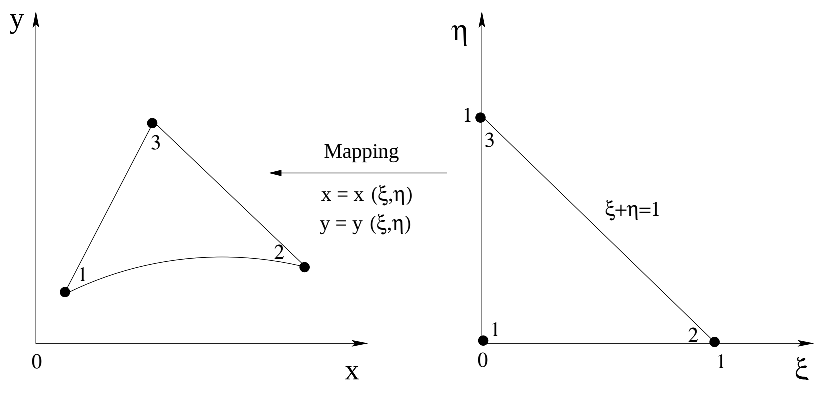


Figure 3.2: Mapping between a physical curvilinear triangle and the reference triangle.

$$\Delta \mathbf{x}_{si} = \mathbf{n}_{si} \sum_{m=1}^{N_d} b_i(x_{si}, D_m)$$
$$\mathbf{x}_{s_i}^{\text{new}} = \mathbf{x}_{s_i}^{\text{old}} + \Delta \mathbf{x}_{si} \tag{3.2}$$

where $\Delta \mathbf{x}_{si}$ represents the displacement at the surface node i. $\mathbf{n}_{si}$ denotes the normal direction at the surface node i. $\mathbf{x}_{s_i}^{(\text{new})}$ and $\mathbf{x}_{s_i}^{(\text{old})}$ represent the new surface coordinates and the old ones from the previous design step, respectively.

Additional Quadrature Points for Surface Shape

As described in the previous chapter, the integrals in Eq. (2.15) are evaluated in the physical triangle but the basis set is defined in the reference triangle, thus a reference-to-physical coordinate transformation is required, as illustrated in Fig. 3.2. The transformation for a specific physical element is determined by its geometric modal mapping coefficients, $\tilde{\mathbf{x}}$ shown in Eq. (2.13). For a linear element mapping (i.e. straight-sided boundary elements), the coordinates of the newly updated surface nodes, $\mathbf{x}_s$, are sufficient to determine the surface geometric modal mapping coefficients, $\tilde{\mathbf{x}}_s$, as well as the Jacobian of the mapping and its metrics for each surface edge or face, due to the linear transformation. However, additional quadrature points are required to determine higher-order geometric mapping modes $\tilde{\mathbf{x}}_s$ for curved surface elements (c.f. Eq. (2.14)), which must be employed in the presence of higher-order solutions ($p \geq 1$). Moreover, surface quadrature points must deform with surface

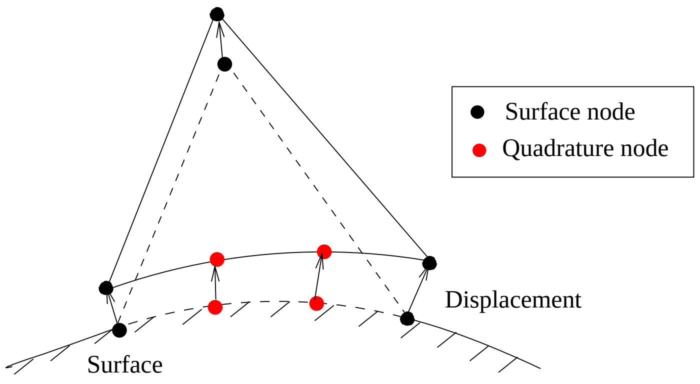


Figure 3.3: Illustration of deformation of curved surface elements with additional quadrature points. The dashed line element represents the initial element shape and the solid line element represents the deformed element shape.

nodes to define new surface mappings, as illustrated in Fig. 3.3. This is achieved by using the same smooth bump functions used for the surface node displacements, shown as:

$$\Delta \mathbf{x}_{q_i} = \mathbf{n}_{q_i} \sum_{m=1}^{N_d} b_i(x_{q_i}, D_m)$$
$$\mathbf{x}_{q_i}^{\text{new}} = \mathbf{x}_{q_i}^{\text{old}} + \Delta \mathbf{x}_{qi} \tag{3.3}$$

where $\Delta \mathbf{x}_{q_i}$ represents the displacement at the quadrature point i and $\mathbf{n}_{q_i}$ denotes its normal direction obtained by the old surface geometry. $\mathbf{x}_{qi}^{(\text{new})}$ and $\mathbf{x}_{qi}^{(\text{old})}$ represent the new surface quadrature coordinate for quadrature node i and the old one from the previous design step, respectively. Therefore, the displacements at the entire set of surface quadrature points are performed similarly to those occurring at the surface grid points.

Interior Mesh Deformation

Once the deformed surface mesh is obtained, the interior mesh must be deformed to prevent the generation of overlapping elements. In the current work, the interior mesh displacements are governed by a discretized set of equations arising from the tension spring analogy [5, 91, 92], in which each edge of the mesh is represented by a spring whose stiffness is related to the length of the edge. The governing equations are expressed as,

$$[K]\Delta\mathbf{x} = \Delta\mathbf{x}_s \tag{3.4}$$

where $\Delta\mathbf{x}$ and $\Delta\mathbf{x}_s$ denote the entire mesh displacements and the surface mesh displacements, respectively. Based on this definition, the vector of the surface mesh displacements $\Delta\mathbf{x}_s$ is a subset of the vector of $\Delta\mathbf{x}$. $[K]$ denotes the stiffness matrix obtained from the discrete mesh motion equations. Thus, the entire set of mesh coordinates, $\mathbf{x}$, and the corresponding coordinate mapping modal coefficients, $\tilde{\mathbf{x}}$, for the aforementioned reference-to-physical mapping can be obtained by solving Eq. (3.4) given the displacements of surface mesh points and surface quadrature points, which are determined by the bump function and design variables as described previously. In order to solve Eq. (3.4), we use several hundred sweeps of a Gauss-Seidel scheme since these equations are relatively simple and inexpensive to solve due to the relatively coarse meshes which are generally employed with high-order discretizations.

Steady-state Flow Equations

The spatially discretized governing equations on the deformed mesh are expressed as:

$$\mathbf{R}\left(\tilde{\mathbf{u}}(\tilde{\mathbf{x}}), \tilde{\mathbf{x}}\right) = 0 \tag{3.5}$$

where $\mathbf{R}$ represents the spatial residual. As described previously, the evaluation of the volume and surface integrals of the discretized governing equations (in Eq. (2.15)) require the reference-to-physical transformation denoted by the coordinate mapping modal coefficients (in Eq. (2.13)), thus the spatially discretized residual is shown as a function of $\tilde{\mathbf{x}}$ rather than $\mathbf{x}$. Additionally, due to the element-based transformation, the coordinate mapping modal coefficients, $\tilde{\mathbf{x}}$, are also element-based. The solution of this system is solved by using the *hp*-Multigrid approach driven by the element Gauss-Seidel smoother described in Section 2.2.3 to accelerate convergence of the non-linear problem, with the understanding that the mesh configuration (and thus $\tilde{\mathbf{x}}$

values) are held fixed throughout the flow solution process.

Objective Functional

Consider a scalar-valued objective functional, L, which refers to the cost functional for the optimization problem and is related to a surface integral of the flow-field variables, such as drag coefficients or target pressure distributions, etc. A general formulation for the objective functional is expressed as:

$$L = L(\tilde{\mathbf{x}}(\mathbf{D}), \tilde{\mathbf{u}}(\tilde{\mathbf{x}}(\mathbf{D}))) \tag{3.6}$$

where $\tilde{\mathbf{u}}$ denotes the converged steady-state flow solution obtained based on Eq. (3.5). Similarly, this functional expression shows the dependence on the coordinate modal coefficients rather than grid node coordinates for the reason described above. Gradient-based shape optimization techniques require the evaluation of the sensitivity of the objective functional with respect to the design variables, i.e. $dL/d\mathbf{D}$. The next section provides a derivation of the discrete adjoint approach for the sensitivity evaluations in the context of discontinuous Galerkin methods.

3.2.2 Discrete Adjoint Approach

The derivation of the discrete adjoint technique for sensitivity analysis starts with the forward tangent problem by taking the derivatives of Eq. (3.6) with respect to the design variables via the chain rule, which leads to the following expression for the sensitivity derivatives,

$$\frac{dL}{d\mathbf{D}} = \left(\frac{\partial L}{\partial \tilde{\mathbf{x}}} + \frac{\partial L}{\partial \tilde{\mathbf{u}}}\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{\mathbf{x}}}\right)\left(\frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{x}_s}\frac{\partial \mathbf{x}_s}{\partial \mathbf{D}} + \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}_q}\frac{\partial \mathbf{x}_q}{\partial \mathbf{D}}\right) \tag{3.7}$$

We next examine each term in the above equation. $\partial \mathbf{x}_s/\partial \mathbf{D}$ represents the sensitivity of surface mesh points with respect to the design variables, which is obtained by linearizing the definition of the bump functions as they appear in Eq. (3.2). Since the surface quadrature points on the geometry surface deform in a similar way as the surface grid points, $\partial \mathbf{x}_q/\partial \mathbf{D}$ refers to the sensitivity of surface quadrature points with respect to the design variables, which is obtained by linearizing Eq. (3.3) with respect to design variables. $\partial \mathbf{x}/\partial \mathbf{x}_s$ represents the sensitivity of the entire set of mesh points with respect to surface node displacements, which is evaluated based on the mesh deformation equation (3.4) as:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{x}_s} = [K]^{-1} \tag{3.8}$$

where $[K]^{-1}$ represents the inverse of the mesh stiffness matrix. We note that the sensitivities $\partial \mathbf{x}/\partial \mathbf{x}_s$ rely only on the grid node information, thus the stiffness matrix has the same formulation as the one used in finite-volume methods [92]. $\partial \tilde{\mathbf{x}}/\partial \mathbf{x}$ refers to the sensitivity of the coordinate mapping coefficients with respect to the entire set of mesh point displacements. Since an individual grid node only affects the coordinate mapping coefficients of the elements sharing this node (as a vertex), $\partial \tilde{\mathbf{x}}/\partial \mathbf{x}$ corresponds to a sparse matrix and only the non-zero entries in the matrix are stored in memory. $\partial \tilde{\mathbf{x}}/\partial \mathbf{x}_q$ refers to the sensitivity of the coordinate mapping coefficients with respect to the surface quadrature displacements, due to the fact that the coordinates of the element vertices as well as the additional quadrature points are required for determining the geometric mapping cooefficients for curved boundary elements (c.f. Eq. (2.14)). In particular, only the elements coinciding with high-order curved boundaries correspond to non-zero blocks in $\partial \tilde{\mathbf{x}}/\partial \mathbf{x}_q$, which are evaluated by linearizing Eq. (2.14) with respect to the coordinates of additional quadrature points. $\partial \tilde{\mathbf{u}}/\partial \tilde{\mathbf{x}}$ denotes the flow sensitivities due to perturbations of geometric mapping modal coefficients, $\tilde{\mathbf{x}}$, which result from the mesh deformation. Due to the fact that the flow variables are defined implicitly by the spatially discretized governing equations expressed in Eq. (3.5), the sensitivities $\partial \tilde{\mathbf{u}}/\partial \tilde{\mathbf{x}}$ can be formulated by linearizing Eq. (3.5) with respect to the design variables via the chain rule, as:

$$\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right] \frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{\mathbf{x}}} \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}} \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{D}} = 0 \quad \text{or} \quad \frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{\mathbf{x}}} = -\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}} \tag{3.9}$$

Substituting Eq. (3.8) and Eq. (3.9) into Eq. (3.7) yields the final expression for the gradient sensitivities, shown as:

$$\frac{dL}{d\mathbf{D}} = \left(\frac{\partial L}{\partial \tilde{\mathbf{x}}} - \frac{\partial L}{\partial \tilde{\mathbf{u}}}\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right]^{-1}\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}}\right)\left(\frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}}[K]^{-1}\frac{\partial \mathbf{x}_s}{\partial \mathbf{D}} + \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}_q}\frac{\partial \mathbf{x}_q}{\partial \mathbf{D}}\right) \tag{3.10}$$

The discrete adjoint problem is then formulated by transposing both sides of Eq. (3.10), which yields:

$$\frac{dL}{d\mathbf{D}}^T = \left(\frac{\partial \mathbf{x}_s}{\partial \mathbf{D}}^T [K]^{-T}\frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}}^T + \frac{\partial \mathbf{x}_q}{\partial \mathbf{D}}^T \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}_q}^T\right)\left(\frac{\partial L}{\partial \tilde{\mathbf{x}}}^T - \frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}}^T\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right]^{-T}\frac{\partial L}{\partial \tilde{\mathbf{u}}}^T\right) \tag{3.11}$$

where $[\cdot]^{-T}$ denotes the inverse of the transposed matrix. Eq. (3.11) corresponds to the formulation for evaluating the discrete adjoint-based sensitivity derivatives in the context of discontinuous Galerkin discretizations. Some notable differences between the above formulation for DG methods and the corresponding adjoint formulation [89] in finite-volume methods can be observed: first, both surface grid points and surface quadrature points are deformed simultaneously based on the bump functions: the terms relevant to the design variables in this formulation include both $(\partial \mathbf{x}_s/\partial \mathbf{D})^T$ and $(\partial \mathbf{x}_q/\partial \mathbf{D})^T$. Second, the sensitivities of the coordinate mapping, $(\partial \tilde{\mathbf{x}}/\partial \mathbf{x})^T$ and $(\partial \tilde{\mathbf{x}}/\partial \mathbf{x}_q)^T$, must be evaluated. As discussed previously, these terms represent the effect of the entire set of mesh points and the surface quadrature points on the geometric mapping modal coefficients. Third, all sensitivities denoted in the second bracket of the right-hand-side of the equation, such as $[\partial \mathbf{R}/\partial \tilde{\mathbf{x}}]^T$ or $(\partial L/\partial \tilde{\mathbf{x}})^T$ must be evaluated with respect to the corresponding modal coefficients of geometric element mapping in the DG discretizations, as opposed to the direct use of grid coordinates in finite-volume methods. This is because geometric information appears directly in the residual and objective formulations only through the Jacobian from the reference to physical mapping as well as its metrics $\partial(x, y)/\partial(\xi, \eta)$.

We note that the term $[\partial \mathbf{R}/\partial \tilde{\mathbf{u}}]$ in Eq. (3.11) refers to the Jacobian matrix of the spatial residual, which has the same formulation as that used in the steady-state flow solver (in Eq. (2.16)). Since a direct solve for the inverse of the full Jacobian matrix or its transpose can be very expensive, the flow adjoint variables, $\boldsymbol{\lambda}$, are introduced by replacing the last two terms, satisfying,

$$\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right]^{-T} \frac{\partial L}{\partial \tilde{\mathbf{u}}}^{T} = \boldsymbol{\lambda} \quad \text{or} \quad \left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right]^{T} \boldsymbol{\lambda} = \frac{\partial L}{\partial \tilde{\mathbf{u}}}^{T} \tag{3.12}$$

Thus the transpose of the Jacobian of the discrete flow equations is used in the definition of the flow adjoint variables, evaluated using the converged steady-state flow state, $\tilde{\mathbf{u}}$, therefore, the flow adjoint problem corresponds to a linear problem. In the present work, this linear system is solved using the same efficient *hp*-Multigrid algorithm used in the flow solver as described in Section 2.2.3, thus producing similar convergence rates, due to the fact that both problems contain the same eigenvalues. However, the linear nature of the adjoint problem often results in lower cost per iteration compared to the flow or primal problem. Substituting the flow adjoint variables $\boldsymbol{\lambda}$ into Eq. (3.11) yields:

$$\frac{dL}{d\mathbf{D}}^{T} = \frac{\partial \mathbf{x}_s}{\partial \mathbf{D}}^{T} [K]^{-T} \frac{\bar{\partial L}}{\partial \mathbf{x}}^{T} + \frac{\partial \mathbf{x}_q}{\partial \mathbf{D}}^{T} \frac{\bar{\partial L}}{\partial \mathbf{x}_q}^{T} \tag{3.13}$$

where

$$\frac{\bar{\partial L}}{\partial \mathbf{x}}^{T} = \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}}^{T} \frac{\bar{\partial L}}{\partial \tilde{\mathbf{x}}}^{T} \tag{3.14}$$

$$\frac{\bar{\partial L}}{\partial \mathbf{x}_q}^{T} = \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}_q}^{T} \frac{\bar{\partial L}}{\partial \tilde{\mathbf{x}}}^{T} \tag{3.15}$$

$$\frac{\bar{\partial L}}{\partial \tilde{\mathbf{x}}}^{T} = \frac{\partial L}{\partial \tilde{\mathbf{x}}}^{T} - \frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}}^{T} \boldsymbol{\lambda} \tag{3.16}$$

where $(\bar{\partial L}/\partial \mathbf{x})^{\mathbf{T}}$ denotes objective sensitivities with respect to the mesh deformation and $(\bar{\partial L}/\partial \mathbf{x}_q)^{T}$ denotes objective sensitivities with respect to the surface quadrature point deformation.

Returning to Eq. (3.13), a similar approach is taken to avoid a direct solve for the inverse of the transposed mesh stiffness matrix $[K]^{-T}$ in Eq. (3.13) by introducing the mesh adjoint variables, $\boldsymbol{\lambda}_x$, satisfying:

$$[K]^{-T} \frac{\bar{\partial L}}{\partial \mathbf{x}}^{T} = \boldsymbol{\lambda}_x \quad \text{or} \quad [K]^{T} \boldsymbol{\lambda}_x = \frac{\bar{\partial L}}{\partial \mathbf{x}}^{T} \tag{3.17}$$

To solve this equation, we use several hundred sweeps of a Gauss-Seidel scheme since these equations are relatively simple and inexpensive to solve due to the relatively coarse meshes which are generally employed with high-order discretizations. Substituting Eq. (3.17) to Eq. (3.13), the final sensitivity is then computed as follows:

$$\frac{dL}{d\mathbf{D}}^T = \frac{\partial \mathbf{x}_s}{\partial \mathbf{D}}^T \boldsymbol{\lambda}_x + \frac{\partial \mathbf{x}_q}{\partial \mathbf{D}}^T \frac{\partial \bar{L}}{\partial \mathbf{x}_q}^T \quad \text{or} \tag{3.18}$$

$$\frac{dL}{d\mathbf{D}} = \boldsymbol{\lambda}_x^T \frac{\partial \mathbf{x}_s}{\partial \mathbf{D}} + \frac{\partial \bar{L}}{\partial \mathbf{x}_q} \frac{\partial \mathbf{x}_q}{\partial \mathbf{D}} \tag{3.19}$$

Since the terms relevant to the design variables (i.e. $\frac{\partial \mathbf{x}_s}{\partial \mathbf{D}}^T$ and $\frac{\partial \mathbf{x}_q}{\partial \mathbf{D}}^T$) are evaluated at the last step in this formulation, the evaluation of the adjoint-based sensitivity derivatives is essentially independent of the number of design variables, which makes the adjoint method well-suited for cases with large numbers of design variables.

3.3 Shape Optimization Procedure

Once the objective sensitivities have been evaluated, they are used to drive the optimization problem. In particular, the perturbation of the design variables is computed based on the steepest-descent method in the present work as:

$$\Delta \mathbf{D} = -\beta \frac{dL}{d\mathbf{D}} \tag{3.20}$$

where β refers to the optimization step size, chosen to be small enough to ensure convergence of the design optimization process and the negative sign implies the direction for minimizing the objective functional.

The complete shape optimization procedure is shown in Fig. 3.4. To summarize, five sequential steps are required for every design cycle:

1. Solve the steady-state flow equations denoted in Eq. (2.15).

2. Solve the flow adjoint and mesh adjoint variables denoted in Eq. (3.12) and Eq. (3.17) respectively.

3. Evaluate gradients or objective functional sensitivities $dL/d\mathbf{D}$ described in the previous section (c.f. Eq. (3.18) or (3.19)).

4. Compute the perturbation of the design variables based on Eq. (3.20) and change $\mathbf{D}$ in the steepest descent direction, as $\mathbf{D}^{\text{new}} = \mathbf{D}^{\text{old}} + \Delta\mathbf{D}$.

5. Deform the geometry shape based on the new design variables and deform the interior mesh based on the mesh motion equations.

This procedure is repeated until the objective functional is sufficiently minimized. The step size β in Eq. (3.20) is chosen by trial and error. Although a line search [93] procedure could be used for this purpose, and would likely result in superior performance of the optimization algorithm, the current approach is simple to implement and sufficient for demonstration purpose.

3.4 Design Results

In this section, a typical two-dimensional design problem is used to demonstrate the design procedure. The design problem consists of minimizing the root mean square (RMS) difference between pressure distributions on a current geometry configuration against those on the target NACA0012 airfoil configuration. The objective functional used for this purpose is given as,

$$L = \sqrt{\frac{\sum_{i=1}^{N_s} \left(p_i - (p_i)_{target}\right)^2}{N_s}} \tag{3.21}$$

where $(p_i)_{target}$ represents the target pressure value at an arbitrary surface node number i and N_s denotes the total number of the surface points. In this test case, the design variables are set to be the magnitudes of the bump functions placed at surface grid points, ranging from 10% to 90% of the chord locations of the upper and lower airfoil surface. Once the objective functional and design variables are given, the sensitivity gradients can be evaluated using the discrete adjoint sensitivity technique described previously.

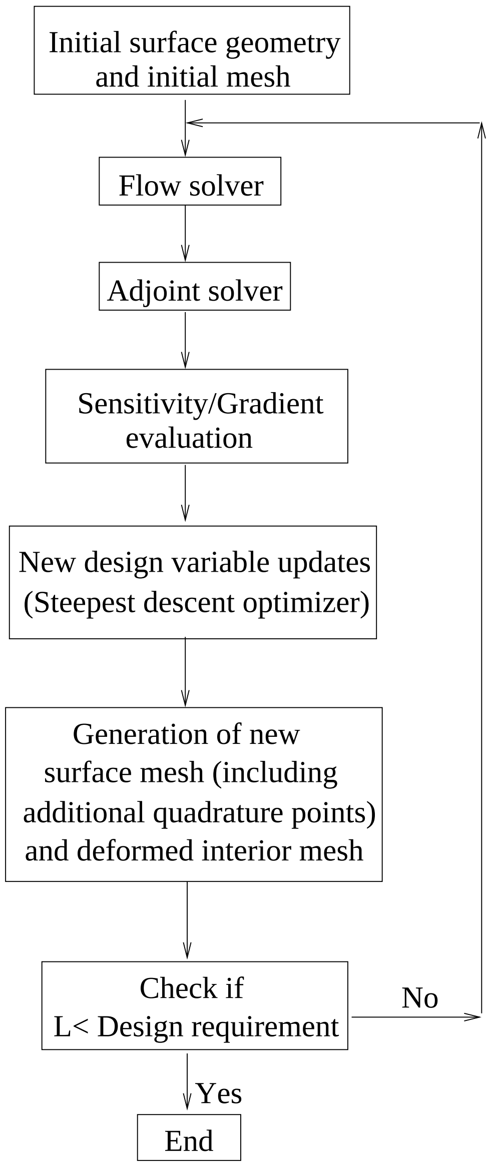

Initial surface geometry and initial mesh
Flow solver
Adjoint solver
Sensitivity/Gradient evaluation
New design variable updates (Steepest descent optimizer)
Generation of new surface mesh (including additional quadrature points) and deformed interior mesh
Check if L< Design requirement
No
Yes
End

Figure 3.4: Illustration of the shape optimization algorithm.

The initial geometry and computational mesh, which contains 2128 unstructured triangular elements and 145 surface node points, are illustrated in Fig. 3.5. The initial airfoil configuration is obtained by deforming the surface nodes of the target airfoil (NACA0012), which ensures that the designed airfoil should match the target airfoil as the objective functional is minimized. A free-stream Mach number of 0.5 and a zero degree of angle of attack are specified for this design problem. A fourth-order ($p = 3$) spatial discontinuous Galerkin discretization is employed and the p-Multigrid

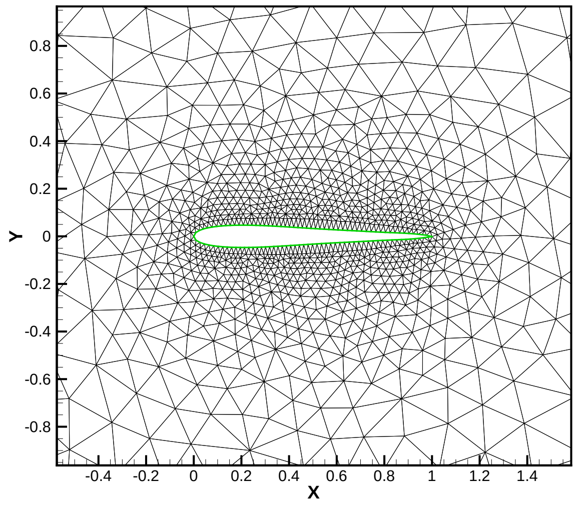


Figure 3.5: Initial wing-body geometry and unstructured computational mesh for the airfoil design optimization problem.

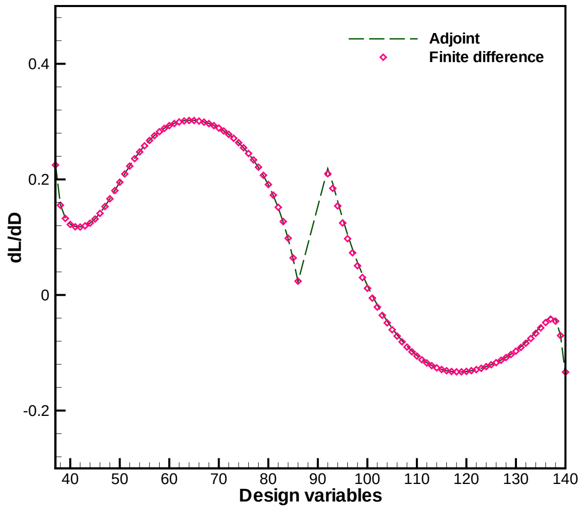


Figure 3.6: Comparison of the computed adjoint-based sensitivities and finite-differenced sensitivities for the $p = 3$ discretization on the initial mesh.

approach with 5 smoothing passes at each multigrid level is employed for the flow and adjoint problems. In addition, the steady-state flow solution from a previous design cycle is used as an initial condition for the next run, in order to provide a good initial solution for the next design step and to reduce computational cost.

Before proceeding to the design optimization procedure, the sensitivity vector computed using the proposed discrete adjoint procedure is verified by comparing with

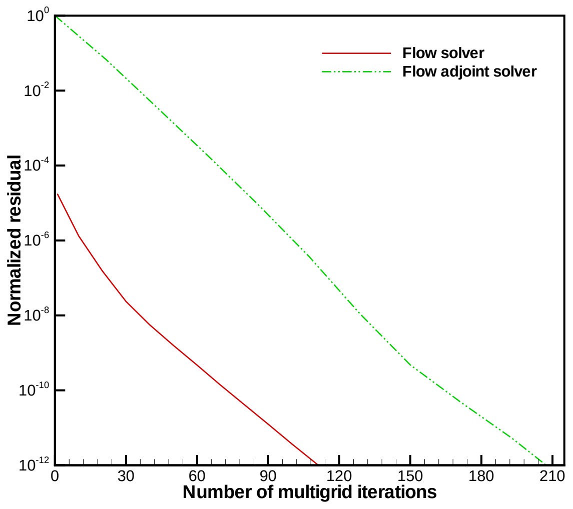
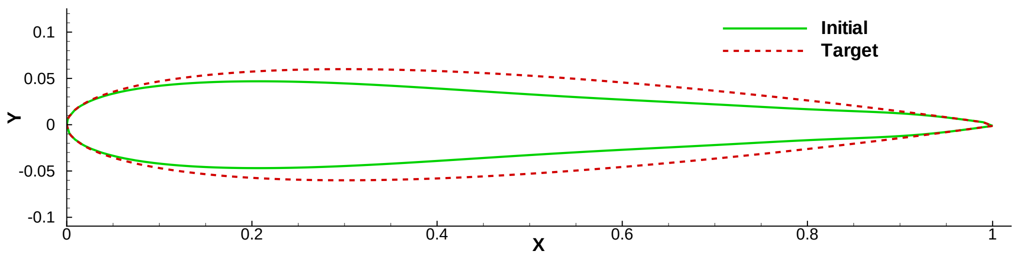


Figure 3.7: Convergence histories of the flow and adjoint solvers at the first design cycle of the optimization problem.

the finite-differenced values. In the finite difference scheme, the i^{th} component of the sensitivity vector (i.e. dL/dD_i) is obtained by perturbing the bump function placed at the airfoil surface point i, obtaining a new surface geometry and re-computing the steady-state flow solution to obtain the change of the objective due to the perturbation D_i. This procedure is invoked for all design variables to obtain the entire sensitivity vector. Fig. 3.6 illustrates the sensitivity values computed using the discrete adjoint approach for the $p = 3$ discretization scheme and the finite-difference scheme. It is shown that the computed adjoint sensitivities provide good agreement with the finite differenced values, thereby verifying the linearization terms determined in Eq. (3.11).

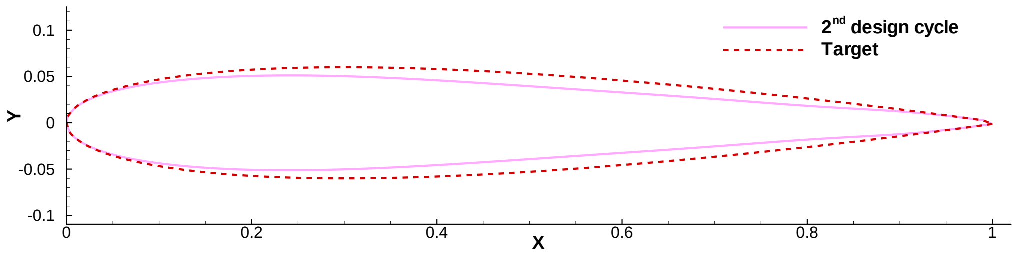
Fig. 3.7 illustrates the p-Multigrid convergence of the flow and adjoint solvers at the first design cycle of the optimization procedure for the $p = 3$ discretization scheme. As can be seen, the discrete adjoint solver delivers a convergence rate which is asymptotically equivalent to that of the flow solver. Fig. 3.8 shows the airfoil surface shapes at various stages of the optimization procedure obtained using the $p = 3$ discretization scheme, compared with the target NACA0012 airfoil geometry. Although the initial airfoil shape differs considerably from the target airfoil, as shown in Fig. 3.8(a), the target airfoil shape is recovered in a few design cycles. For example, the airfoil shape produced at the 4^{th} design stage matches the target very closely.

Fig. 3.9 plots the computed objective against the number of shape optimization cycles. It is clearly shown that the objective functional decreases significantly for the initial steps (e.g. step 1 to 4), although convergence slows down in the later design steps. Nevertheless, a one order of magnitude reduction in the objective is achieved in approximately 40 design cycles and the shape of the target airfoil is reproduced closely. It is noted that the steepest-descent optimization approach used in this example constitutes a relatively simple and crude optimization approach (particularly when using a fixed step size β) and superior performance should be achievable with more sophisticated optimizers such as LBFGS [93, 94].

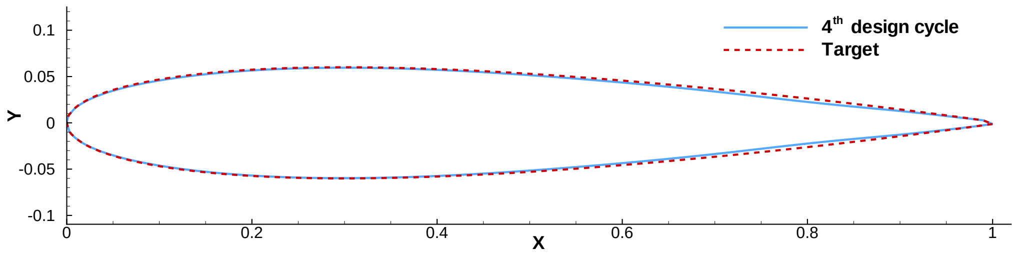


q

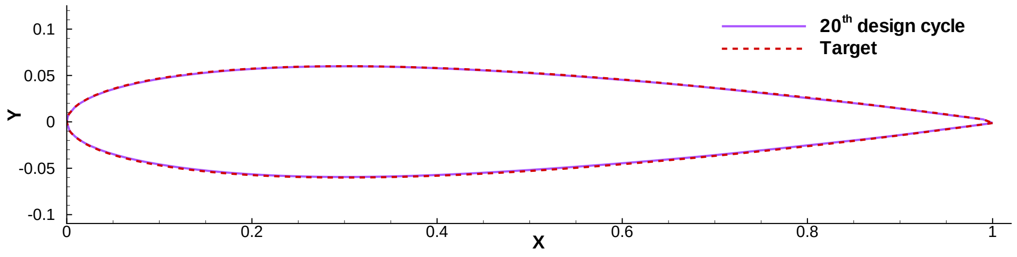
(a) Design cycle 1



(b) Design cycle 2



(c) Design cycle 4



(d) Design cycle 20

Figure 3.8: Results of airfoil surface shapes against the target NACA0012 airfoil geometry at various stages of the optimization procedure.

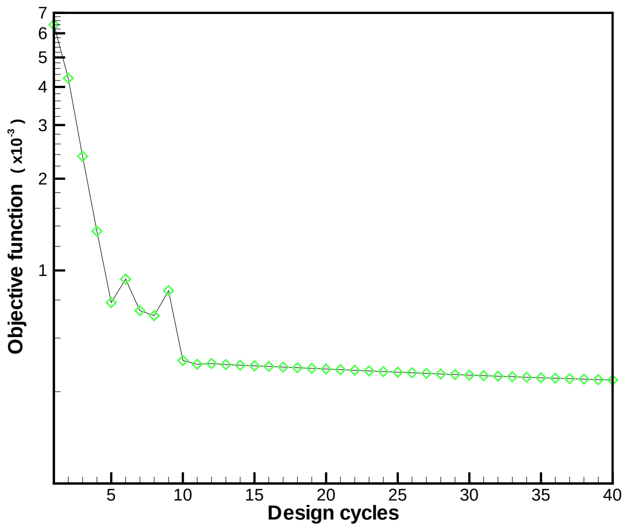


Figure 3.9: Convergence of the objective functional in terms of number of design cycles for the optimization procedure.

Chapter 4

Adjoint-based Spatial Error Estimation and Adaptation

In this chapter, an adaptive discontinuous Galerkin algorithm driven by an adjoint-based spatial error estimation technique for the inviscid compressible Euler equations is investigated and developed. The main procedure to implement this approach consists of solving the flow (i.e. primal) solution and the adjoint (i.e. dual) solution corresponding to a particular simulation objective output of interest. The benefit of the adaptive algorithm is that it provides a well-founded technique for estimating the error in specific simulation outputs and resulting in a spatial error distribution which can be used to guide adaptive refinement strategies for automatically producing accurate simulations at optimal cost in the objectives of interest.

4.1 Objective

Practical formulations for goal-oriented a posteriori error estimation based on adjoint techniques have been developed and demonstrated successfully, initially within the context of a variational framework [27, 49, 69–72, 95], and more recently for finite-volume discretizations [24–26, 73, 96, 97]. Fidkowski and Darmofal [98] also presented a side-by-side comparison of output error estimation using the finite element method as well as finite volume methods. In this work, high-order discontinuous Galerkin discretizations are combined with adjoint error estimation techniques and adaptive refinement strategies, with the goal of developing simulation strategies capable of delivering a prescribed level of accuracy in specific engineering objectives of interest. The discrete adjoint formulation developed in Section 3.2.2 is employed exclusively and the objective error estimates are derived in a non-variational framework, although the resulting equations are identical to those obtained using a variational framework. One notable difference with some previous formulations is that we estimate only the change in the functional value between two successively refined or enriched mesh levels, as opposed to attempting to estimate the total error defined as the change between the current functional value and the value which would be obtained in the continuous limit [24, 69]. The current approach, which has been demonstrated successfully for complex three-dimensional problems [73], involves fewer approximations and is sufficient for adaptive mesh refinement purposes. In order to be competitive, the gains afforded by the reduced number of degrees of freedom for a given accuracy level achieved by the adaptive process, compared to a global refinement strategy, must outweigh the additional cost of the error estimation technique, as well as the cost of solving the flow and adjoint problem on multiple intermediate adaptively refined levels. To this end, the cost of the error estimation technique is designed to be equivalent to, or in most cases lower than, the cost of solving the analysis problem on the current refinement level. This is achieved by solving the adjoint problem on the same mesh as the analysis problem at each refinement level, and using reconstruction techniques to obtain a fine level adjoint field, as required in the error estimation formulation.

In the context of discontinuous Galerkin discretizations, accuracy enhancements can be achieved either by refining the mesh (h), or by raising the order of accuracy of the discretization (p). In this work, purely h and purely p refinement approaches are examined, as well as a combined hp-adaptive approach, which has been shown to be optimal, in the sense that it has the potential to enable exponential error convergence [50]. The key to choosing between h and p lies in the ability to assess the local smoothness of the solution, and thereby two previously described smoothness indicators [99–102] are employed for this purpose. When properly implemented, the

hp-adaptive scheme has the potential to capture discontinuous phenomena such as strong shocks without the need for limiting, since the smoothness indicator acts to preserve low-order discretizations in such regions.

The standard p- or hp-Multigrid approach begins with an approximation level represented by the highest discretization order p, and coarser approximation levels consist of lower-order discretizations, such as $p-1$, $p-2$, down to $p = 0$, while the same grid is used by all p-levels. This approach applies naturally to cases with a uniform discretization order (p) of the domain. However, a discretization generated using either p- or hp-refinement results in the presence of different orders of discretizations in different regions of the mesh. This complicates the implementation of the standard p-Multigrid approach, which assumes the presence of a uniform discretization order p on each multigrid level. In this context, a zonal multigrid strategy is investigated as a modification to the traditional multigrid approach described previously. The goal of this investigation is to further improve overall efficiency of the multigrid approach by avoiding appearance of certain elements of the mesh on multiple multigrid levels with unchanged approximation orders.

The outline of this chapter is as follows. In Section 4.2 key techniques for estimating the error in the functional with respect to its values on a globally refined or p-enriched fine mesh are described. This procedure avoids expensive computations of the flow and adjoint solutions on the fine mesh, while requiring a linear adjoint problem to be solved on the original coarse mesh. Section 4.3 demonstrates various adaptive strategies to locally refine the coarse mesh in areas which most adversely influence the functional accuracy, as predicted by the aforementioned error estimation technique. In Section 4.4, several numerical test cases including subsonic compressible flow over an idealized four-element airfoil geometry and hypersonic compressible flow over a half-circular cylinder are used to exhibit the performance of the adaptive solution strategy, as well as the ability of the hp-adaptive scheme to capture strong shocks or discontinuities while improving functional accuracy. Finally in Section 4.5, the basic idea of the zonal multigrid approach in the context of variable p-order discretizations is described and computational results are compared with the prescribed global multigrid approach with respect to both convergence and computational cost.

4.2 Adjoint-based Spatial Error Estimation

In this chapter, the model problems involve the two-dimensional steady-state compressible Euler equations. The discretization and solution strategies used for these equations have been described in the preceding chapter. Here we address and analyze adjoint-based error estimates for specific functional outputs. The goal of this procedure is to obtain a spatial distribution of the functional error which can be used either to correct the current functional value or to drive an adaptive meshing procedure for improved functional accuracy [103].

4.2.1 Functional Output

In many engineering and aerospace applications, some key objective functional outputs are of particular interest in computational simulations, such as the objective functional of lift, drag or integrated wall temperature, as chosen in this work. Once the flow solution is obtained, it is relatively simple to evaluate the functional outputs. For example, the Cartesian force vector $\mathbf{F}$=$(F_x, F_y)^t$ acting on the boundaries of interest ($\partial\Omega_w$) for the model problem is given as,

$$F_x = \int_{\partial\Omega_w} p\ n_x dS \qquad F_y = \int_{\partial\Omega_w} p\ n_y dS \tag{4.1}$$

Then the lift and drag which are defined as the components of $\mathbf{F}$ perpendicular and tangential to the free-stream velocity, respectively, are derived as,

$$L = -F_x\ sin\alpha + F_y\ cos\alpha \qquad D = \ F_x\ cos\alpha + F_y\ sin\alpha \tag{4.2}$$

where α is the angle of attack. The objective functional of integrated temperature on a solid surface is computed as:

$$T_I = \int_{\partial\Omega_w} T dS = \int_{\partial\Omega_w} \frac{p}{\rho} dS \tag{4.3}$$

4.2.2 Error Estimation

Consider a coarse mesh $\mathcal{T}_H$, where the parameter H represents an average element size as well as a low discretization order p for the current finite-element mesh. Let $L(\tilde{\mathbf{u}})$ denote an objective functional of interest, which is computed based on the flow-field variables in terms of the expansion coefficients, $\tilde{\mathbf{u}} = \{\tilde{\mathbf{u}}_{i,k}, i = 1, \ldots, M; k = 1, \ldots, N\}$ in Eq. (2.7). The objective functional can be evaluated on the coarse mesh, represented as $L_H(\tilde{\mathbf{u}}_H)$ by using the steady-state flow solution $\tilde{\mathbf{u}}_H$ that satisfies Eq. (2.5) on the coarse mesh. We seek an approach to approximate the objective functional on a globally refined mesh $\mathcal{T}_h$, as illustrated in Fig. 4.1, by either subdividing each element into four congruent elements (subdivided mesh), or by increasing the order of discretizations from p to $p+1$ (enriched mesh). However, in practice we wish to avoid computing the flow solution on $\mathcal{T}_h$ directly due to the possibly high expense of solving the fine level flow problem. Thus the output functional $L_h(\tilde{\mathbf{u}}_h)$ on the fine mesh is expanded with respect to the projected coarse mesh flow solution, $\tilde{\mathbf{u}}_H^h = I_H^h \tilde{\mathbf{u}}_H$, based on a Taylor series expansion, expressed as:

$$L_h(\tilde{\mathbf{u}}_h) = L_h(\tilde{\mathbf{u}}_H^h) + \left(\frac{\partial L_h}{\partial \tilde{\mathbf{u}}_h}\right)_{\tilde{\mathbf{u}}_H^h} (\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h) + \cdots \tag{4.4}$$

where $L_h(\tilde{\mathbf{u}}_H^h)$ is the output functional on the fine mesh evaluated with the projected coarse mesh flow solution. The vector $\left(\frac{\partial L_h}{\partial \tilde{\mathbf{u}}_h}\right)_{\tilde{\mathbf{u}}_H^h}$ denotes the sensitivities of the fine level objective functional with respect to the fine level flow variables evaluated at the state $\tilde{\mathbf{u}}_H^h$. If the output functional consists of a surface integral, the sensitivities of the objective with respect to the flow solution are non-zero only on boundary elements which intersect the surface under consideration. Similarly, the fine level residual vector $\mathbf{R}_h = \{\mathbf{R}_{hj,k}, j = 1, \ldots, M; k = 1, \ldots, N\}$ can be expanded about the projected coarse level flow solution:

$$\mathbf{R}_h(\tilde{\mathbf{u}}_h) = \mathbf{R}_h(\tilde{\mathbf{u}}_H^h) + \left[\frac{\partial \mathbf{R}_h}{\partial \tilde{\mathbf{u}}_h}\right]_{\tilde{\mathbf{u}}_H^h} (\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h) + \cdots \tag{4.5}$$

where $\left[\frac{\partial \mathbf{R}_h}{\partial \tilde{\mathbf{u}}_h}\right]_{\tilde{\mathbf{u}}_H^h}$ is the full Jacobian matrix of the fine level flow problem evaluated by the projected coarse mesh flow solution state. We proceed to derive an approximation of the solution error, $\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h$, by re-arranging Eq. (4.5), given as,

$$\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h \approx - \left[\frac{\partial \mathbf{R}_h}{\partial \tilde{\mathbf{u}}_h} \right]_{\tilde{\mathbf{u}}_H^h}^{-1} \mathbf{R}_h(\tilde{\mathbf{u}}_H^h) \tag{4.6}$$

where the fact that $\mathbf{R}_h(\tilde{\mathbf{u}}_h) = 0$ has been used, since $\tilde{\mathbf{u}}_h$ represents the finite-element flow solution on the fine mesh. Substituting Eq. (4.6) into Eq. (4.4), we obtain an expression to approximate the fine mesh functional as,

$$L_h(\tilde{\mathbf{u}}_h) \approx L_h(\tilde{\mathbf{u}}_H^h) - \left(\frac{\partial L_h}{\partial \tilde{\mathbf{u}}_h} \right)_{\tilde{\mathbf{u}}_H^h} \left[\frac{\partial \mathbf{R}_h}{\partial \tilde{\mathbf{u}}_h} \right]_{\tilde{\mathbf{u}}_H^h}^{-1} \mathbf{R}_h(\tilde{\mathbf{u}}_H^h) \tag{4.7}$$

where the flow residuals $\mathbf{R}_h(\tilde{\mathbf{u}}_H^h)$ are non-zero since the coarse level flow solution projected onto the refined mesh $\tilde{\mathbf{u}}_H^h$ does not satisfy the discretized flow equations in the fine space. In order to relate the functional error to the local residuals of the flow solution and to prevent computing the inverse of the Jacobian matrix directly, the fine adjoint variable, $\boldsymbol{\lambda}_h$, is introduced, satisfying:

$$\left[\frac{\partial \mathbf{R}_h}{\partial \tilde{\mathbf{u}}_h} \right]_{\tilde{\mathbf{u}}_H^h}^{T} (\boldsymbol{\lambda}_h)_{\tilde{\mathbf{u}}_H^h} = \left(\frac{\partial L_h}{\partial \tilde{\mathbf{u}}_h} \right)_{\tilde{\mathbf{u}}_H^h}^{T} \tag{4.8}$$

In actual fact, Eq. (4.8) is the approach corresponding to the discrete adjoint formulation for the flow adjoint variables discussed in the previous chapter. Recalling this linear system, the coefficient matrix constructed for the adjoint problem corresponds to the transpose of the full-Jacobian matrix of the flow equations and thus the adjoint solution scheme delivers similar convergence characteristics as the flow problem. Substituting Eq. (4.8) into Eq. (4.7) and rearranging, we may write the functional on the fine mesh as:

$$L_h(\tilde{\mathbf{u}}_h) \approx L_h(\tilde{\mathbf{u}}_H^h) \underbrace{- (\boldsymbol{\lambda}_h)_{\tilde{\mathbf{u}}_H^h}^T \mathbf{R}_h(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a^*} \tag{4.9}$$

The ε_a^* term corresponds to an estimate of the error in the fine mesh functional $L_h(\tilde{\mathbf{u}}_h)$, and involves the fine mesh adjoint solution, $\boldsymbol{\lambda}_h$. In order to avoid solving the fine mesh adjoint problem, we first solve the discrete adjoint problem on the coarse mesh:

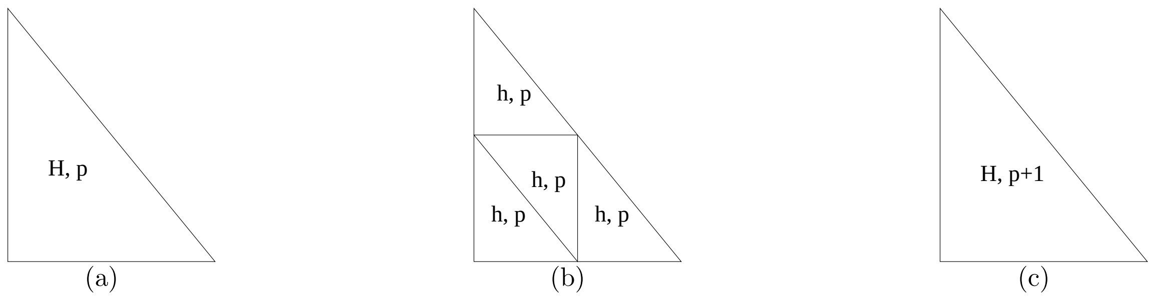


Figure 4.1: Illustration of refined finite-element meshes; (a) coarse mesh, (b) subdivided mesh, (c) enriched mesh.

$$\left[\frac{\partial \mathbf{R}_H}{\partial \tilde{\mathbf{u}}_H}\right]^T \boldsymbol{\lambda}_H = \left(\frac{\partial L_H}{\partial \tilde{\mathbf{u}}_H}\right)^T \tag{4.10}$$

using the *hp*-Multigrid approach driven by the linearized element Gauss-Seidel smoother [16,20] described in sections 2.2.2 and 2.2.3 to accelerate convergence. An approximation to the fine level adjoint solution is then obtained by performing a reconstruction postprocessing procedure to the coarse level adjoint solution, $\boldsymbol{\lambda}_H$, onto the refined mesh via a patch-wise least square method [76], where the reconstructed adjoint solution at an element k ($k \in \mathcal{T}_h$) is solved by minimizing the following equations:

$$I\Big((\boldsymbol{\lambda}_H^h)_{i_k}\Big) = \sum_{l \in \mathcal{P}_k} \Big\| \sum_{j=1}^{M^*} (\boldsymbol{\lambda}_H^h)_{j_k} \phi_j|_l - \sum_{j=1}^{M} (\boldsymbol{\lambda}_H)_{j_k} \phi_j|_l \Big\|_{L_2}^2, \quad i = 1, \cdots, M^* \tag{4.11}$$

with respect to each variable as:

$$\frac{\partial I\Big((\boldsymbol{\lambda}_H^h)_{i_k}\Big)}{\partial (\boldsymbol{\lambda}_H^h)_{j_k}} = 0 \quad i, j = 1, \cdots, M^* \tag{4.12}$$

where $\mathcal{P}_k$ represents the patch of the element k consisting of all of its neighboring elements, and the subscript $(\cdot)_{i_k}$ denotes the i th expansion coefficient of the adjoint solution in element k. M^* the is number of modes required for the order of discretization on the fine mesh. Using the reconstructed adjoint expression in Eq. (4.9), we obtain the error expression:

$$L_h(\tilde{\mathbf{u}}_h) \approx L_h(\tilde{\mathbf{u}}_H^h) \underbrace{-(\boldsymbol{\lambda}_H^h)^T \mathbf{R}_h(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a} \underbrace{-((\boldsymbol{\lambda}_h)_{\tilde{\mathbf{u}}_H^h} - (\boldsymbol{\lambda}_H^h))^T \mathbf{R}_h(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_r} \tag{4.13}$$

where ε_a and ε_r denote the computable error correction and the remaining error, respectively. The computable error correction ε_a is expressed as the inner product of the local residuals with the reconstructed adjoint solution, and results in a spatial distribution of the functional error. The remaining error term is typically an order of magnitude smaller than the computable error correction, and thus the remaining error can be safely absorbed into the adjoint correction without compromising the reliability of the adaptive algorithm. Therefore the functional approximation on a globally refined mesh becomes:

$$L_h(\tilde{\mathbf{u}}_h) \approx L_h(\tilde{\mathbf{u}}_H^h) - (\boldsymbol{\lambda}_H^h)^T \mathbf{R}_h(\tilde{\mathbf{u}}_H^h) \tag{4.14}$$

Next the variation of the discrete functional between coarse and fine levels, $L_h(\mathbf{u}_h) - L_H(\mathbf{u}_H)$, is derived by subtracting from both sides the coarse functional, $L_H(\mathbf{u}_H)$, written as:

$$\underbrace{L_h(\mathbf{u}_h) - L_H(\mathbf{u}_H)}_{\varepsilon_c} \approx \underbrace{L_h(\mathbf{u}_H^h) - L_H(\mathbf{u}_H)}_{\varepsilon_d} \underbrace{-(\boldsymbol{\lambda}_H^h)^T \mathbf{R}_h(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a} \tag{4.15}$$

In this form, the additional term ε_d is the error incurred between the evaluation of the functional on the fine level using the projected coarse mesh flow solution, and the functional evaluated on the coarse mesh level with the coarse mesh flow solution. Referring to Eq. (4.15), the major computational work at each adaptation cycle involves solving the flow and adjoint problems on the coarse mesh and the proper reconstruction procedure to the fine mesh level.

4.2.3 Refinement Criteria

The adjoint correction ε_a provides a spatial functional error estimator on each element k in the current computational domain, given as,

$$\varepsilon_{a,k} = -(\boldsymbol{\lambda}_H^h)_k^T \mathbf{R}_{h,k}(\tilde{\mathbf{u}}_H^h) \tag{4.16}$$

where the local element-wise error indicator is expressed as the inner product of the local residual vector with the approximated adjoint variables within the children elements of a coarse element k. In order to adaptively reduce the error in the objective functional, an error-balancing refinement criterion [49] is employed in the current work where elements are flagged for local refinement if the inequality

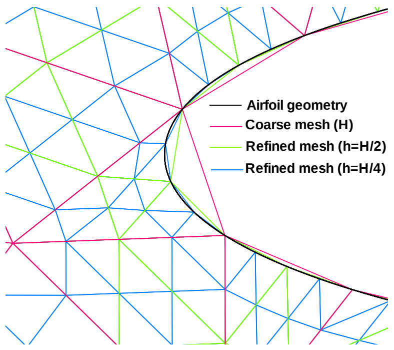
$$|\varepsilon_{a,k}| > \frac{E_{tol}}{N} \tag{4.17}$$

holds, where E_{tol} is a positive user-desired global tolerance and N denotes the number of elements in $\mathcal{T}_H$. Eq. (4.17) provides a stopping criterion for the simulation when the local error indicator on the current mesh is within the maximum allowable equidistributed error level $\bar{\eta} = \frac{E_{tol}}{N}$ in each element.

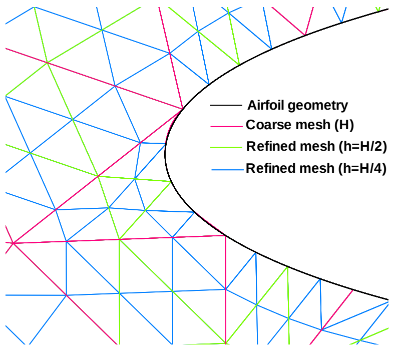
We note that an alternate approach [25, 27] in the adaptive criteria is to estimate ε_r, as denoted in Eq. (4.13) using the combination of both primal-based and dual-based error estimate expressions and to reduce this quantity by adaptive mesh refinement. On the other hand, the adaptive criteria in this work is based on the correction term, ε_a, obtained by using the reconstructed adjoint solution, which is seen to be sufficient for adaptive mesh refinement purposes [72, 103]. If the termination criterion is not reached, we store a list of indices of the flagged elements and then proceed to the local mesh refinement operation presented in the following section to improve the quality of the finite-element mesh.

4.3 Adaptive Mesh Refinement

In the context of discontinuous Galerkin discretizations, accuracy enhancements can be achieved either by refining the mesh (h), or by raising the order of accuracy (p) of the discretization. Therefore, the mesh refinement operation can be classified as h-refinement, p-enrichment and combined hp-refinement.



(a)



(b)

Figure 4.2: An example of h-refinement for an original coarse mesh and its consecutively refined meshes near an airfoil boundary. (a): refinement on a fixed discretization order $p = 1$. (b): refinement on a fixed discretization order $p = 4$.

4.3.1 *h*-refinement

h-refinement involves locally refining the mesh by adding nodes to midpoints of each of the edges of flagged elements on the current coarse mesh. Thus an h-refinement procedure subdivides one element into four self-similar children elements (1:4 refinement), while keeping the original discretization order p fixed. New nodes on interior edges are simply added at the edge midpoints since all interior elements have straight-sided edges. However, for wall or surface boundary elements, in order to ensure that the newly generated nodes and and additional quadrature points for curvilinear boundary conditions [16] conform to the original geometry boundary, a quintic spline interpolation is used based on the original surface (data) grid points to obtain the required new surface information. Fig. 4.2 illustrates an example of the meshes obtained by an h-refinement process on an airfoil using fixed discretization orders $p = 1$ and $p = 4$, respectively.

Upon subdividing one flagged element into four congruent children elements, hanging nodes may be generated on shared edges where only a single side of two neighboring elements has been marked for refinement. Several algorithms are employed in this work in order to eliminate the presence of hanging nodes, as depicted in Fig. 4.3: A given element with a single hanging node is subdivided into two chil-

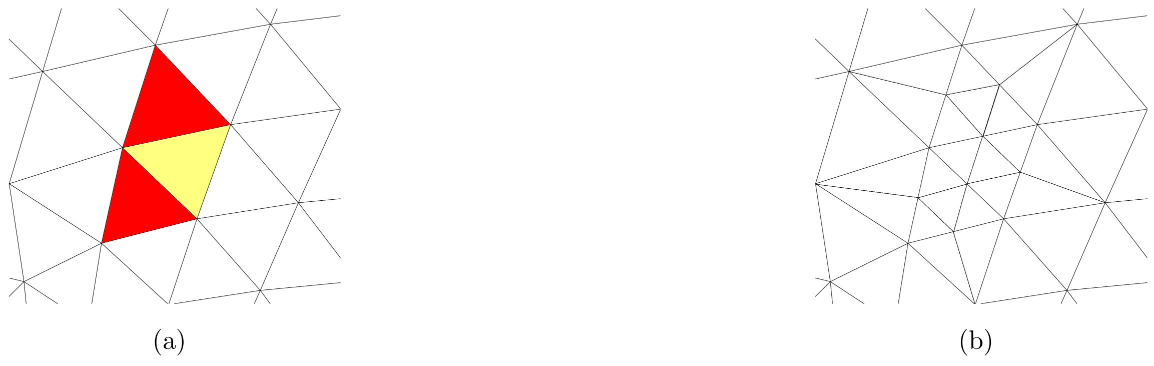


Figure 4.3: Example of h-refinement to eliminate the presence of hanging nodes. (a): Original mesh where the red elements represent elements flagged for regular 1:4 refinement and the yellow element represents a forced 1:4 refinement in this element due to the generation of two additional nodes. (b): The subsequent h-refined mesh.

dren elements (1:2 refinement); if two hanging nodes are generated in one element, a regular 1:4 subdivision is implemented. Since repeated implementation of the 1:2 refinement due to hanging nodes on a given element may result in poorly shaped elements, only a single level of 1:2 refinement is permitted, and 1:4 refinement is employed if subsequent refinement of these elements is required. Additional mesh optimization techniques [104] are also employed including an edge-swapping procedure, where the local topology of the mesh is changed based on a criterion which maximizes the minimum interior angle of the elements (a MaxMin triangulation), and a mesh smoothing procedure where the positions of the mesh points are relocated to the centroid of the surrounding mesh points which are connected to the current point, while the mesh topology remains unchanged.

4.3.2 *p*-enrichment

An alternate approach to mesh h-refinement is p-enrichment, where the local discretization order p is increased to $p + 1$, while the underlying triangulation remains fixed. Note that during the process of the adaptive p-enrichment, it is very common to have two neighboring elements with different discretization orders, as illustrated in Fig. 4.4. In order to obtain the design solution accuracy, the number of quadrature points must increase with higher discretization orders [36]. Thus, in this context, the

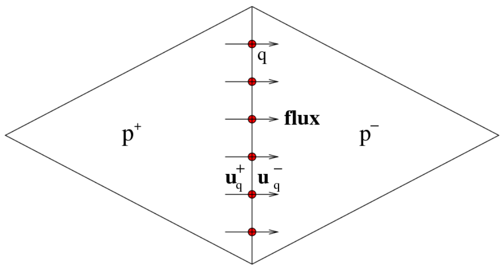


Figure 4.4: Illustration of an interior boundary between two neighboring elements with different interpolation orders.

number of quadrature points required to calculate the surface integral at a shared edge of two neighboring elements of different discretization orders is taken as the number required to satisfy the element with the higher discretization orders. However, the inner and outer traces of the conservative flow variables at each quadrature point need to be approximated by their own expansion coefficients and basis functions, and the optimal number of quadrature points required to evaluate the volume integral in Eq. (2.15) is based on the discretization order of the computed element itself.

p-enrichment is known to be capable of achieving exponential convergence in terms of number of unknowns for smooth solutions. However, this approach can not be employed directly for non-smooth solutions or for cases with shocks or singularities, since the higher-order discretizations ($p > 0$) produce unbounded oscillations in such regions and may result in loss of stability.

4.3.3 *hp*-refinement

In order to avoid higher-order discretizations in non-smooth solution regions in the context of an adaptive scheme, the combined *hp*-refinement approach is invoked. In this case, a choice between *h*-refinement and *p*-enrichment must be made individually for each one of the elements flagged as large error contributors on the current coarse mesh. The goal of the combined *hp*-adaptive refinement scheme is to make use of *h*-refinement in regions with discontinuities or large flow gradients where high-order discretizations may not perform suitably and to utilize *p*-enrichment in areas with smooth solution behavior to deliver high accuracy. This is accomplished by employing a smoothness indicator based on an element-wise or inter-element indicator.

Smoothness Indicator

Since the coefficients in the solution expansions of the DG methodology are assumed to have a similar decay rate ($1/n^2$) as the Fourier coefficients for the case of smooth solutions, we utilize a local smoothness indicator which involves element-wise integrals defined as:

$$S_k = \frac{(q - \breve{q}, q - \breve{q})_k}{(q, q)_k} \tag{4.18}$$

and yields a scalar sensor as a measure of smoothness. In this expression, $(\cdot, \cdot)$ denotes the standard L_2 inner product within element k. q and $\breve{q}$ represent one representative quantity of the flow variables such as density or Mach number with a full (for order p) or truncated expansion (for order $p-1$), respectively, written as:

$$q = \sum_{i=1}^{M} \tilde{q}_i \ \phi_i \tag{4.19}$$

$$\breve{q} = \sum_{i=1}^{\bar{M}} \tilde{q}_i \ \phi_i \tag{4.20}$$

where M and $\bar{M}$ denote the total number of terms in the solution expansions of discretization orders p and $p-1$, respectively. Note that the same expansion coefficients $\tilde{q}_i$ of this quantity are used in Eq. (4.19) and Eq. (4.20). An automatic criteria for choosing between h-refinement and p-enrichment within flagged elements is given as [99–101]:

$$\begin{cases} s_k \geq s_0 - \kappa, & h-\text{refinement} \\ s_k < s_0 - \kappa, & p-\text{enrichment} \end{cases} \tag{4.21}$$

where, $s_k = log_{10}(S_k)$ and the parameter $s_0 \sim \frac{1}{p^4}$, and κ is an empirical parameter and set equal to 6.0 in our experience so as to capture non-smooth regions. Moreover, we use density in practice as the quantity q to determine the decay rate of expansion coefficients.

The other smoothness indicator used in this work particularly for the cases of high speed flows in the next section is a jump discontinuity indicator [100, 102] which measures the integral of the discontinuous jump of the flow-field variables over each edge of an element, given as,

$$S_k = \frac{1}{|\partial\Omega_k|} \int_{\partial\Omega_k} \left| \frac{q^+ - q^-}{\frac{1}{2}(q^+ + q^-)} \right| dS \tag{4.22}$$

where q^+ and q^- denote the inner and outer traces of the selected quantity. The discontinuity detection scheme is implemented as:

$$\begin{cases} S_k > \frac{1}{\mathcal{K}}, & h-\text{refinement} \\ S_k \leq \frac{1}{\mathcal{K}}, & p-\text{enrichment} \end{cases} \tag{4.23}$$

The parameter $\mathcal{K}$ is required to be sufficiently large to capture any strength of shocks and in our experience it is set equal to 25. The jump indicator performs better than the element-wise smoothness indicator for high speed flows and pressure has been found to be a reliable quantity to calculate inter-element jumps.

Shock Capturing

For high-speed flows, it is well known that limiters or added artificial dissipation are required in order to robustly capture strong shocks with higher-order discretizations. On the other hand, the amount of dissipation added by a first-order spatial discretization scheme is sufficiently large to handle any shock using an appropriate Riemann solver. However, the dissipation added by a first-order accurate scheme is proportional to the element size, h, and therefore large numbers of small elements are required for high accuracy. Thus, the role of the *hp*-refinement approach for problems with strong shocks is to provide enhanced accuracy through mesh refinement in shock regions identified by a smoothness indicator, and simultaneously to increase discretization orders in smooth areas to improve functional accuracy.

4.4 Results

The adaptive mesh strategies are first applied to a subsonic flow over an idealized four-element airfoil. Comparisons of the adapted meshes and the error convergence histories are demonstrated for the h-refinement alone as well as for the p-enrichment alone, and for the combined hp-adaptive approach. The functional error estimates are examined for each adaptation process of the various mesh adaptation strategies. A second test case involves strong shocks or discontinuities produced by a hypersonic flow over a half-circular cylinder in order to demonstrate the shock-capturing properties of the hp-adaptive approach.

The input for the simulations consists of an initial coarse mesh $\mathcal{T}_H$ associated with a uniformly lower discretization order p, an objective functional of interest as well as a user-desired functional error tolerance. The output includes the refined mesh $\mathcal{T}_{\widehat{h}}$ together with a polynomial degree distribution $\widehat{p}_k, k \in \mathcal{T}_{\widehat{h}}$, and the final flow solution as well as final functional value. The performance of the adaptive process is measured in terms of the reduction of the error in the functional output versus the number of unknowns (i.e. degrees of freedom) or the computational cost in CPU time. Degrees of freedom (DOF) are computed as the total number of unknowns, excluding the number of subcomponents for the system of equations (i.e. 4 for the two-dimensional Euler equations). The computational time required for one adaptation cycle includes the accumulated expense of computing both the flow and adjoint problems for all previous adaptation cycles plus an additional flow solution for the current adapted mesh. On the other hand, the computational cost for the uniform refinement approach is only considered as the expense of solving a single flow problem on the uniformly refined mesh.

4.4.1 Subsonic Flow over a Four-element Airfoil

The computational domain for this test case consists of an idealized four-element airfoil with a far-field outer boundary placed at a distance of 50 chord lengths away from the airfoil. The initial mesh illustrated in Fig. 4.5 contains 1,508 unstructured

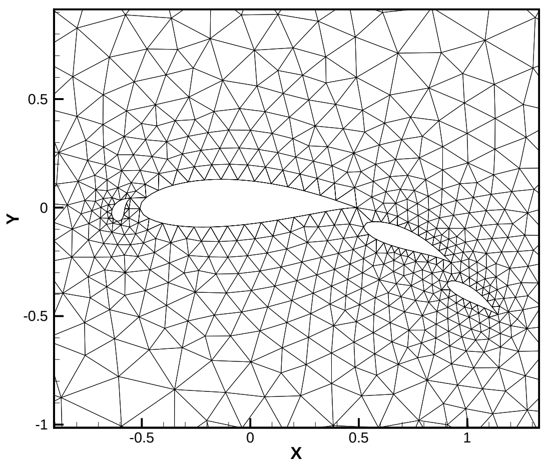


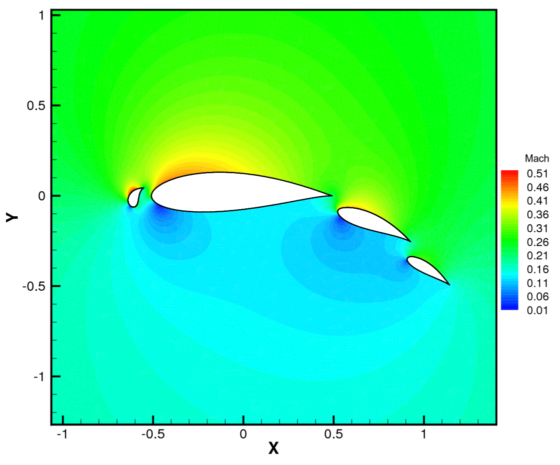
Figure 4.5: Close-up view of the initial unstructured mesh for a four-element airfoil (1,508 elements).

triangular elements. A low free-stream Mach number of 0.2 as well as a zero incidence are prescribed for the flow field. The HLLC approximate Riemann solver [45, 46] is used for the flux function at all interior edges/boundaries to resolve the discontinuity in the flow variables at each elemental interface, and the particular boundary treatment described in Section 2.2 is employed at all physical boundaries. The exact functional value is taken from a solution on a h-adapted mesh with a uniformly $p = 3$ discretization order, where the relative difference in the functional between the current and the previous adapted functional is within 10^{-4}.

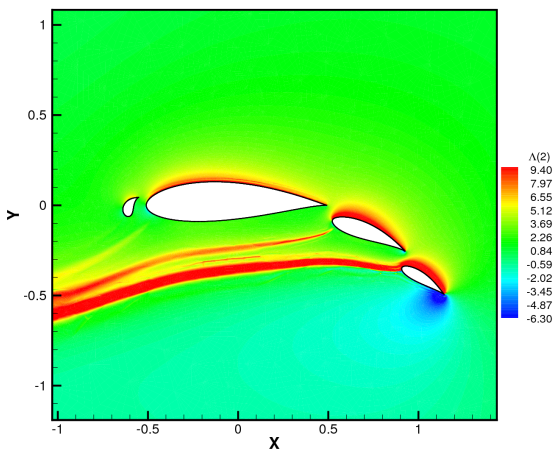
hp-Multigrid Solution

As mentioned previously, both the flow and the discrete adjoint problems must be solved at each mesh adaptation cycle, which consumes most of the computational cost (over 95%) throughout the adaptive simulation. Thus, in order to make the overall scheme competitive, we make use of efficient solution techniques including the hp-multigrid approach driven by the element Gauss-Seidel smoother described in sections 2.2.2 and 2.2.3 to accelerate convergence for both the flow and adjoint problems.

The computed steady-state solution in terms of Mach number contours and the



(a) Mach number contours



(b) x-momentum component of the discrete adjoint solution

Figure 4.6: Flow and discrete adjoint solutions for subsonic flow over a four-element airfoil with a free-stream Mach number of 0.2; Lift is set to be the objective functional of interest.

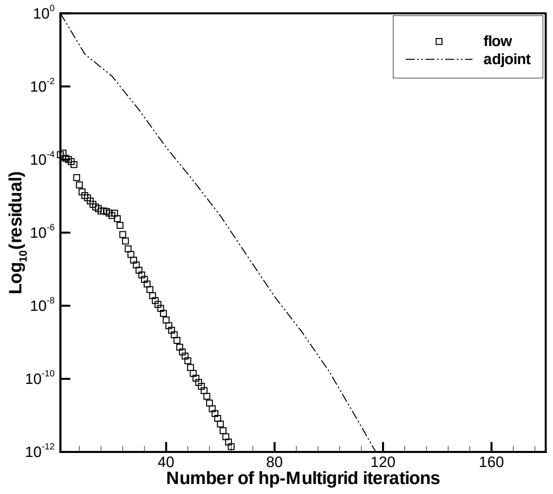


Figure 4.7: Comparison of convergence histories of both flow and adjoint solutions in terms of number of *hp*-Multigrid cycles.

x-momentum component of the adjoint solution corresponding to the objective functional of lift on the initial mesh with a uniform $p = 4$ order of discretization are illustrated in Figures 4.6(a) and 4.6(b), respectively. The x-momentum component of the adjoint solution corresponds to the sensitivity of the lift value to point sources of x-momentum in the flow field. The fact that regions upstream of the airfoil denoted by the red contours in Fig. 4.6(b) have a significant effect on the lift output provides an illustration of the non-local effect due to the hyperbolic property of the governing equations. Fig. 4.7 provides a comparison of convergence histories for both the flow and the adjoint solvers in terms of the number of *hp*-multigrid cycles. The discrete adjoint solver delivers a convergence rate which is asymptotically equivalent to that of the flow solver, since the adjoint problem is constructed as an exact dual to the primal problem.

Error Prediction for Two Mesh Levels

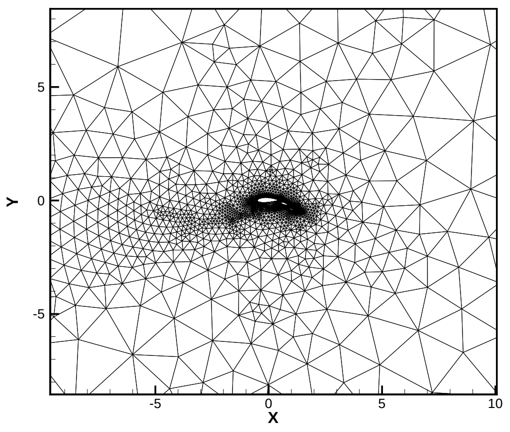
Before proceeding to the mesh adaptation test cases, it is important to verify that the computable adjoint error correction denoted as ε_a in Eq. (4.13) is capable of providing an acceptable approximation for the fine adjoint error correction, denoted as ε_a^* in Eq. (4.9), as well as providing an accurate prediction of the corresponding globally refined mesh value. Table 4.1 and Table 4.2 consider the comparisons of the computed drag change between two mesh levels, where the fine level is obtained by increasing the grid resolution using a uniform 1:4 h-refinement (Case A) or by raising the discretization order p to $p+1$ globally (Case B), with the computable adjoint error correction ε_a and the corresponding term ε_a^* evaluated using the fine level adjoint solution. It can be observed that the reconstructed adjoint error correction provides good agreement with the fine adjoint error estimates. For example, the correction term evaluated using the reconstructed adjoint solution agrees to within 88.0% of the corresponding error evaluated using the fine level adjoint solution for Mesh 1 in Case A. However, this agreement increases to 99.1 % for Mesh 3. In Case B, the reconstructed adjoint solution also provides acceptable agreement with the corresponding error relevant to the fine level adjoint solution: a 61.0 % agreement is provided for Mesh 1 and a 91.4 % for Mesh 4. This implies that the reconstructive procedure for the coarse level adjoint sufficiently captures the behavior of the fine level adjoint. Moreover, the accuracy of the resulting estimates based on the computable adjoint error is satisfactory compared with the corresponding two-level functional error. For example, the computed ε_a value corresponds to 98.7% of the actual two-level error for Mesh 3 of Case A and 99.7% for Mesh 4 of Case B.

Table 4.1: Case A: comparisons of the adjoint error correction for drag output computed using fine and reconstructed coarse adjoint solutions (ε_a^* and ε_a, respectively) with the true functional error between two mesh levels where grid resolution is increased by a uniform 1:4 refinement and the discretization order $p = 1$ is fixed for all runs. Mesh 1, 2 and 3 contain 1508, 6032 and 24128 elements respectively.

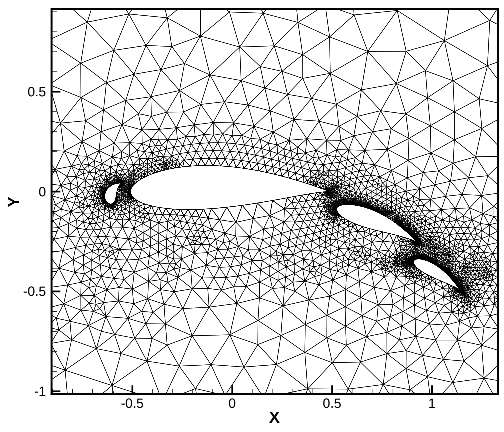
Mesh	error	ε_a^* (based on $\boldsymbol{\lambda}_h$)	ε_a (based on reconstructed $\boldsymbol{\lambda}_H^h$)
1	-2.76903217E-3	-2.71897249E-3	-2.39419266E-3
2	-7.10679246E-4	-7.41825861E-4	-7.19017106E-4
3	-1.14173193E-4	-1.16541513E-4	-1.15579326E-4

Table 4.2: Case B: comparisons of the adjoint error correction for drag output computed using fine and reconstructed coarse adjoint solutions (ε_a^* and ε_a, respectively) with the true functional errors between two mesh levels where the discretization order is uniformly increased from p to $p+1$ and the underlying grids (1508 elements) are fixed for all runs. Mesh 1, 2, 3 and 4 are uniformly discretized by $p = 0$, 1, 2, and 3, respectively.

Mesh	error	ε_a^* (based on $\boldsymbol{\lambda}_h$)	ε_a (based on reconstructed $\boldsymbol{\lambda}_H^h$)
1	-3.81409957E-2	-3.32547210E-2	-2.03270239E-2
2	-2.82996939E-3	-3.03437520E-3	-1.33580368E-3
3	-5.72914101E-4	-5.41733478E-4	-5.85074179E-4
4	-1.72394870E-4	-1.82852793E-4	-1.71881095E-4



(a) Final h-adapted mesh (8,387 elements)



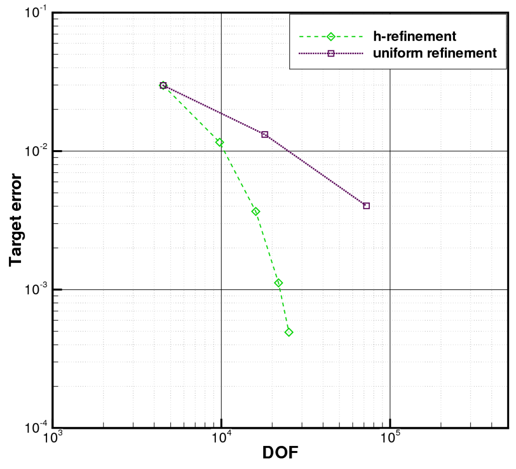
(b) Close-up view of the final adapted mesh

Figure 4.8: Final h-adapted mesh for the objective functional of lift and a fixed discretization order of $p = 1$ (i.e. second-order accurate) in the subsonic flow test case.

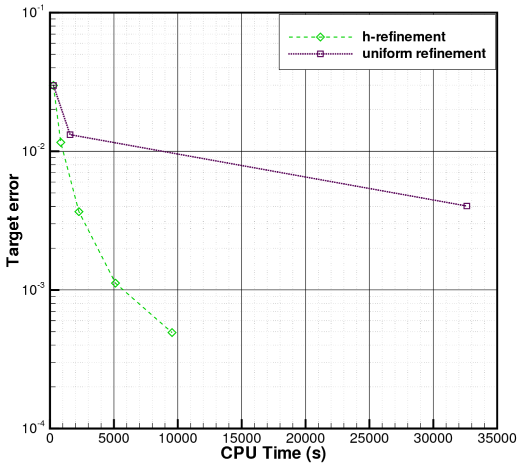
h-adaptation for Lift Output

Here the performance of the h-adaptive mesh refinement algorithm driven by the adjoint-based error estimation is demonstrated for the case of inviscid subsonic flow over a four-element airfoil. The objective functional of interest is specified as lift, which is calculated from the static pressure distribution on the airfoil surface as described in Section 4.2.1. A uniform discretization order of $p = 1$ (i.e. second-order accurate) is fixed for all h-adaptive refinement levels and the error tolerance is set to be 10^{-3}. Figures 4.8(a) and 4.8(b) depict the final h-adapted mesh with 8,387 elements after four adaptation cycles. As expected, most of the refinement occurs around the airfoil surface, particularly near the leading and trailing edges. However, some h-refinement occurs upstream of the airfoil as shown in Fig. 4.8(a) due to the hyperbolic nature of the problem.

The convergence behavior of the functional error for the adaptive approach is compared with that achieved using uniform mesh refinement approach, where each element in the computational domain is refined by the aforementioned 1:4 ratio, regardless of the objective functional error contribution. Thus, the objective functional value obtained from the uniformly refined mesh represents the optimal functional er-



(a) Lift error vs. degrees of freedom



(b) Lift error vs. CPU time

Figure 4.9: Comparison of error convergence histories for the target functional of lift between the h-refinement and uniform mesh refinement approaches, using a fixed discretization order of $p = 1$.

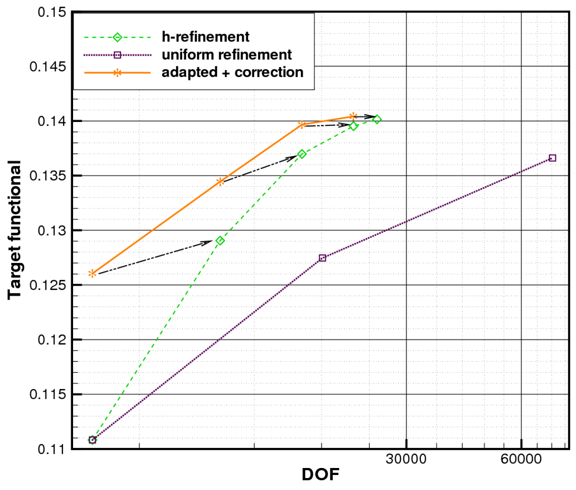


Figure 4.10: Comparison of lift functional convergence as a function of degrees of freedom for the h-refinement and uniform mesh refinement approaches using a fixed discretization order of $p = 1$ for the subsonic test case.

ror reduction achievable on the corresponding h-adaptively refined mesh. Fig. 4.9 summarizes the results of the convergence of the functional error for both approaches. In terms of degrees of freedom as depicted in Fig. 4.9(a), the h-refinement approach achieves equivalent error levels as the uniform refinement approach at each adaptation cycle, while using fewer degrees of freedom. In other words, the uniform mesh refinement approach contains excessive resolution in areas of little influence on the functional accuracy. For example, at the second adaptation cycle, the number of unknowns required by the h-refinement approach is only 22% of the number employed by the uniform refinement method, and this advantage is even more evident for the two following h-adaptation iterations. Moreover, in terms of the required CPU time shown in Fig. 4.9(b), the adaptive approach incurs significantly lower computational cost resulting from the use of fewer degrees of freedom at equivalent accuracy, even though the computational cost of the adaptive approach includes all previous adaptive flow and adjoint solutions, while the uniform refinement case only includes the cost of the fine level flow solution alone.

Next we examine the effectiveness of the error estimate or correction produced by the adjoint technique for predicting the functional value on the next finer level. Fig. 4.10 depicts the lift functional convergence histories on the h-adapted meshes, the functional values on the adapted meshes including the corresponding correction, and the functional values computed on the uniformly refined meshes, in terms of degrees of freedom for this subsonic test problem. The correction produced by the adjoint-based error estimation becomes more accurate with increasing refinement levels, and the functional value on the two finest levels is predicted very accurately on the previous coarser level, as indicated by arrows shown in this figure. The increasing effectiveness of the correction term on finer levels is explained by the fact that this is based on a linearization about the coarser level solution. The correction term can be used to predict the value of the functional on a finer mesh level.

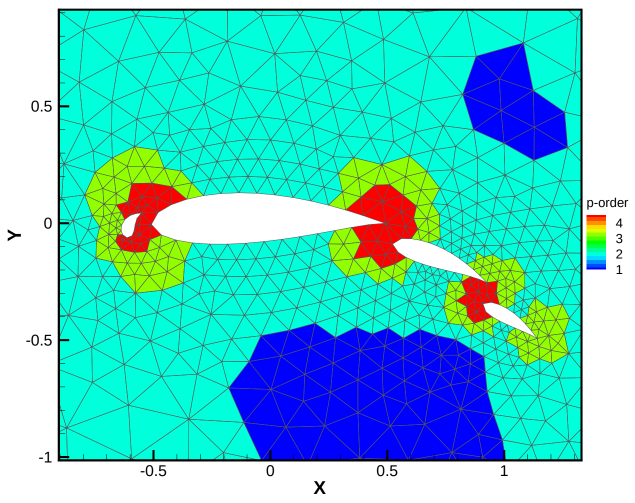
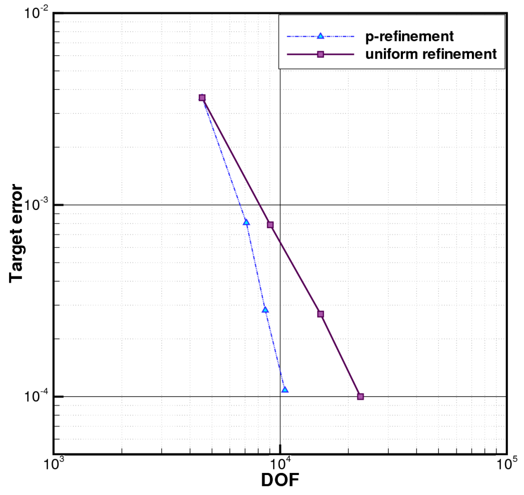


Figure 4.11: Final p-adapted mesh (1,508 elements; $p_{\min} = 1, p_{\max} = 4$) for the objective functional of drag in the subsonic flow test case.

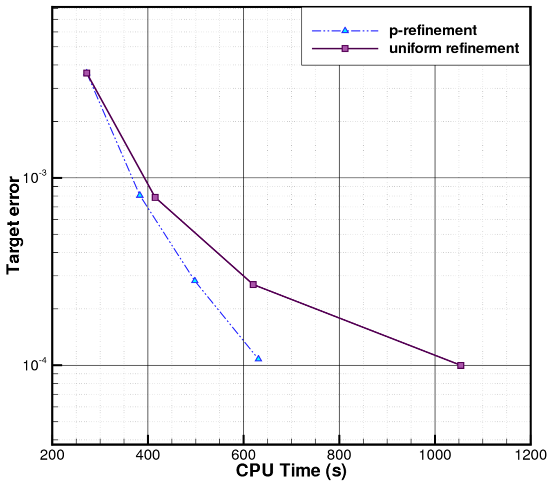
p-adaptation for Drag Output

In this case, the p-enrichment algorithm is performed using drag as the output. The same coarse mesh illustrated in Fig. 4.5 together with a uniform $p = 1$ discretization and an error tolerance of 5×10^{-4} are set as the initial inputs. The final p-adapted mesh depicted in Fig. 4.11 illustrates the variation of discretization orders ranging from $p = 1$ to $p = 4$ while operating on the same initial mesh. Areas targeted for order refinement are mainly concentrated on the airfoil surface particularly around the leading and trailing edges, which demonstrates similar results in the lift-h-adaptation case.

The error convergence history of the p-adaptive enrichment algorithm for this subsonic test case is also compared with that of the uniform order refinement approach, where the refined mesh is obtained by globally increasing the discretization order from p to $p + 1$ with the underlying mesh fixed. The functional error obtained from the uniform order refinement method is used to determine the optimal achievable error reduction possible for one p-adaptation cycle. In terms of degrees of freedom, as illustrated in Fig. 4.12(a), the p-enrichment scheme is also capable of delivering equivalent error levels as the uniform refinement approach at each adaptation cycle,



(a) Drag error vs. degrees of freedom



(b) Drag error vs. CPU time

Figure 4.12: Comparison of error convergence histories for the target functional of drag between the p-enrichment and uniform order refinement approaches.

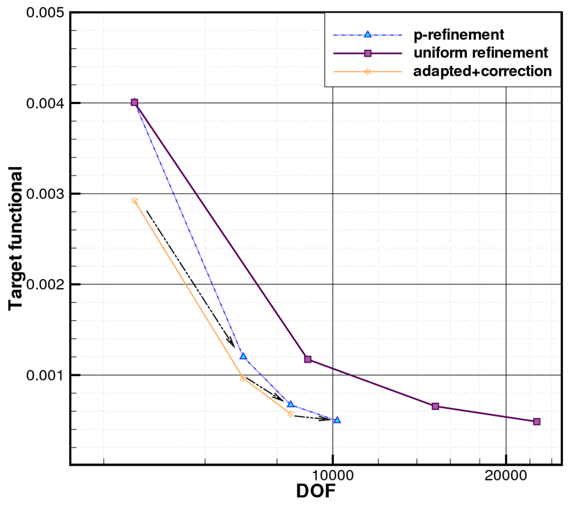


Figure 4.13: Comparison of drag functional convergence for the adaptive p-enrichment and the uniform order refinement approaches in terms of degrees of freedom for the subsonic test case.

while requiring fewer degrees of freedom. In terms of CPU cost, as shown in Fig. 4.12(b), the p-enrichment scheme demonstrates superior performance compared to the uniform order-refinement scheme particularly on the last two adaptation cycles. For example, to achieve a 10^{-4} drag error, the uniform order refinement method incurs roughly twice the cost of the adaptive scheme in CPU time. The examination of the effectiveness of the error estimates or the correction term provided by the adjoint-based error estimation for the p-enrichment test case is demonstrated in Fig. 4.13. The functional value obtained from the purely p-adapted mesh at each refinement level is close to the functional obtained by the uniform order refinement. Moreover, the approximated functional for the next finer level provided by the functional with the added correction term on the previous coarse adaptation level (as represented by the orange line) is seen to become increasingly more accurate with additional refinement levels, and the finest objective functional is predicted very accurately for the final adaptation level using the computable error correction term.

***hp*-adaptation for Drag Output**

The following test case considers the effective combination of both h- and p-refinement algorithms (i.e. the approach of hp-adaptation) for subsonic flow over the four-element airfoil, using drag as the objective functional. The element-wise smoothness indicator described by Eq. (4.18) in Section 4.3.3 is employed for this subsonic test case.

The hp-adaption test case starts with the same initial mesh (Fig. 4.5) together with a uniform discretization order of $p = 1$ and the desired error tolerance is specified as 10^{-5}. The final adapted mesh is shown in Fig. 4.14 with a total of 7,105 elements and a variation of discretization orders ranging from $p = 1$ to $p = 4$. Areas of high h-refinement are concentrated near the airfoil leading edges, due to the presence of large gradients in these regions. On the other hand, areas of high order p-enrichment mainly occur near the surface of the last two airfoil-elements.

Fig. 4.15(a) compares the drag error convergence of hp-refinement with pure h-refinement and pure p-enrichment implemented using the same initial meshes for this test case. In terms of degrees of freedom, the h-refinement approach represented

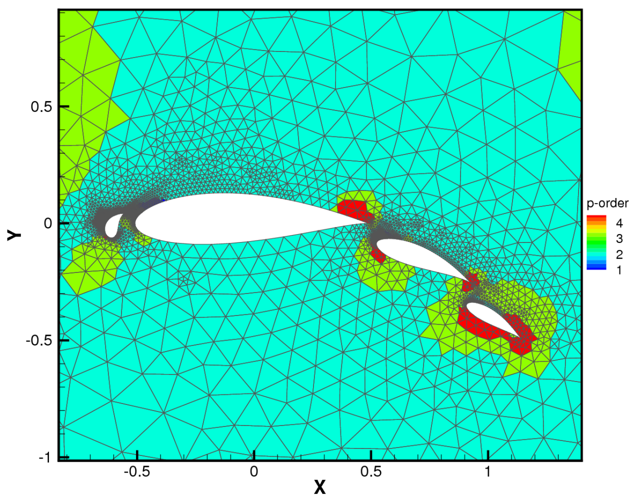
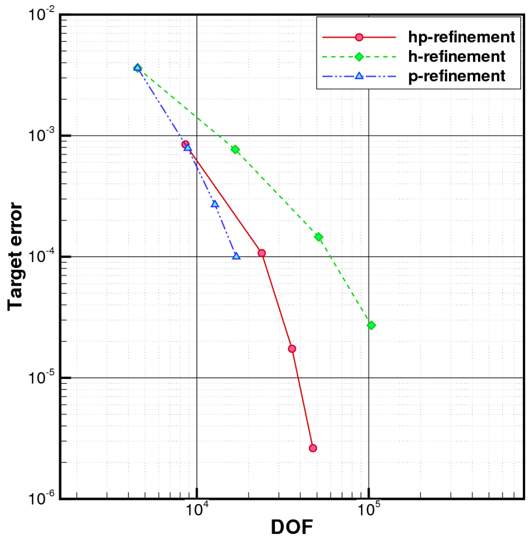


Figure 4.14: Final hp-adapted mesh (7,105 elements; $p_{\min} = 1, p_{\max} = 4$.) for the objective functional of drag in the subsonic flow test case.

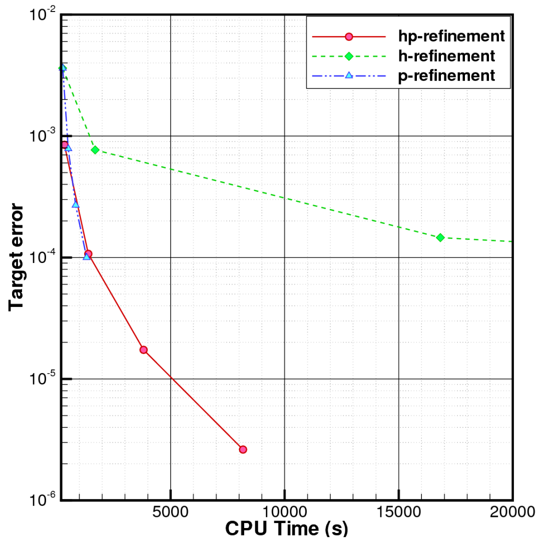
by the green line requires more degrees of freedom for the same error reduction as the p- and hp-refinement approaches. Both hp-refinement and p-enrichment algorithms demonstrate exponential error convergence rates but the slope of the hp-enrichment (i.e slope = 5.3) exceeds the slope of 2.9 achieved using p-enrichment alone. In addition, for a fixed range of discretization orders ($p_{\min} = 1, p_{\max} = 4$), the hp-refinement scheme is capable of achieving roughly two orders of magnitude higher error reduction than the p-enrichment scheme. In terms of CPU cost, as illustrated in Fig. 4.15(b), the performance of the hp-refinement scheme demonstrates superior efficiency over the purely h-refinement scheme, while displaying similar cost compared to the p-enrichment scheme over the initial refinement levels. Fig. 4.16 exhibits the convergence of the functional for the hp-adaptive case in terms of degrees of freedom. It can be observed that the final output functional is accurately predicted by the corrected functional on the second and third adaptation cycles.

4.4.2 Hypersonic Flow over a Half-circular Cylinder

The next test case involves the computation of hypersonic flow over a half-circular cylinder with a free-stream Mach number of 6 using the hp-adaptive scheme. The far-



(a) drag error vs. degrees of freedom



(b) drag error vs. CPU time

Figure 4.15: Comparisons of error convergence histories for the functional of drag between the *hp*-refinement, pure *h*-refinement and pure *p*-enrichment schemes.

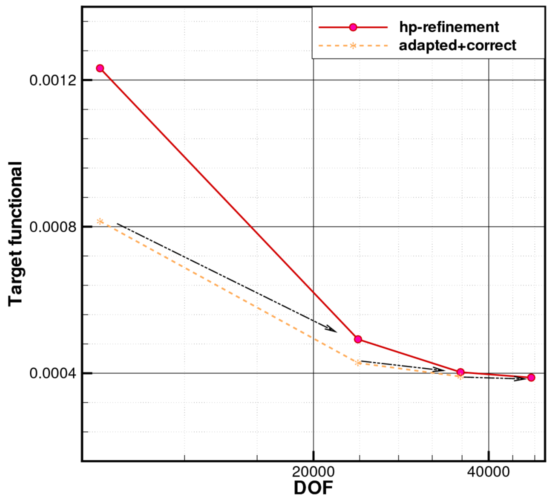


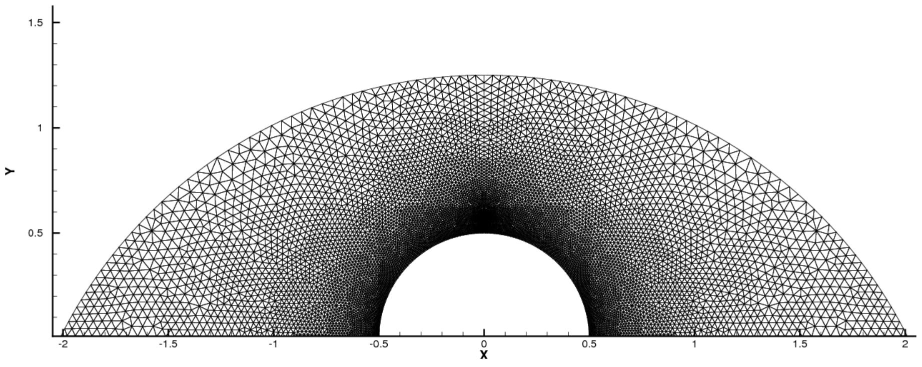
Figure 4.16: Comparisons of drag functional convergence for the *hp*-refinement approach in the subsonic test case.

field inlet boundary of the mesh is treated using a fully prescribed supersonic inflow condition. The inviscid fluxes of the flow solver are calculated using the Lax-Friedrichs flux functional [47] and the flow solution obtained at the previous adaptation cycle is used as an initial condition for the next finer refinement cycle. A first-order accurate scheme ($p = 0$) is used initially to ensure a stable solution, and the hp-adaptive refinement strategy is implemented in areas where the spatial functional error has relatively large contributions. The switch between mesh subdivision (i.e. h-refinement) and variation of discretization order (i.e. p-enrichment) is based on the inter-element smoothness indicator evaluated by Eq. (4.22).

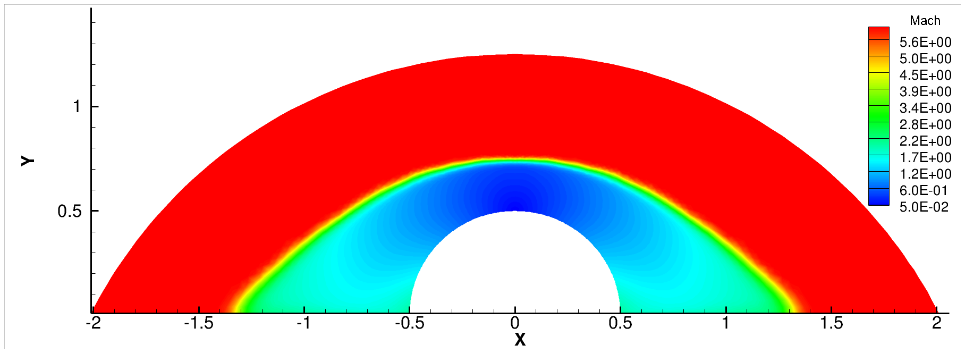
hp-adaptation for Integrated Surface Temperature

The objective functional in this test case is set to the integrated temperature on the cylinder surface (c.f. Eq. (4.3)) since surface heating is of significant interest for hypersonic problems. Here we examine the shock resolution as well as the functional accuracy convergence throughout the adaptation process. The adaptation starts with an initial mesh shown in Fig. 4.17(a) and a uniform $p = 0$ order of discretization. Since the mesh in the shock region is relatively coarse, discontinuities are not well resolved (spreading over $4 \sim 5$ elements), as observed from the Mach number contours illustrated in Fig. 4.17(b).

The sequentially adapted meshes together with the distributions of discretization orders are illustrated in Fig. 4.18. It can be observed that no refinement takes place ahead of the shock due to the fact that the uniform flow ahead of the shock wave is exactly represented by the first-order ($p = 0$) discretization in this region. Substantial h-refinement occurs around/in the shock regions where a first-order accurate scheme, represented by the white color on the plots, is maintained throughout all the hp-adaptation cycles, and higher-order discretizations are prescribed behind the shock and ahead of the cylinder. The final adapted mesh illustrated in Fig. 4.18(c) contains a dense distribution of h-refinement in shock areas and a range of discretization orders from $p = 0$ to $p = 3$ in the region between the shock and cylinder. It is also noted that the entire shock region is not refined, since lateral regions of the shock wave



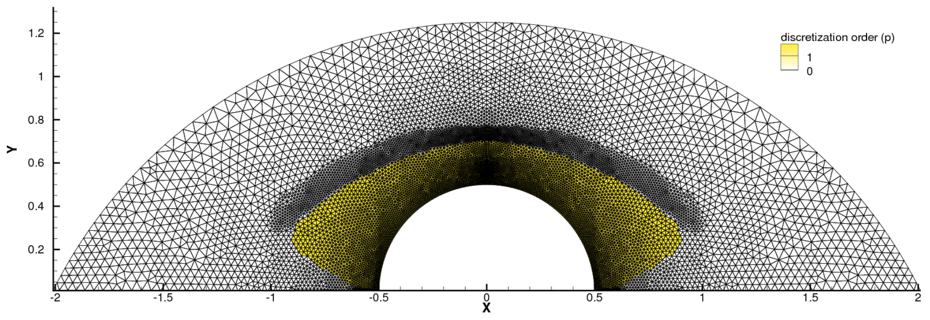
(a) Initial mesh, 17072 elements



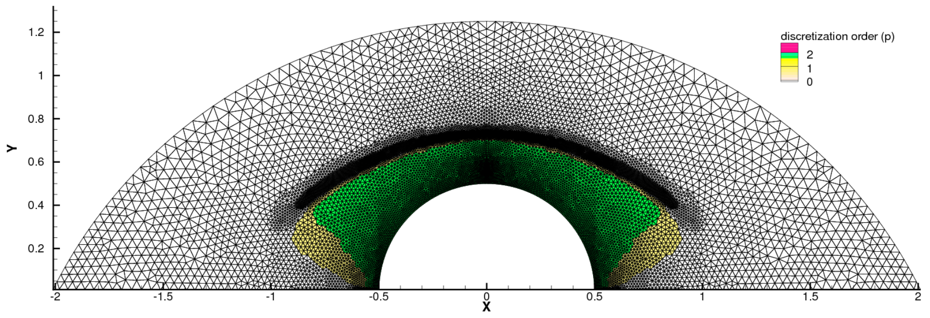
(b) Mach number contours

Figure 4.17: Initial mesh and computed Mach number contours using a first-order accurate discretization ($p = 0$) for hypersonic flow passing a half-circular cylinder with a free-stream Mach number of 6.

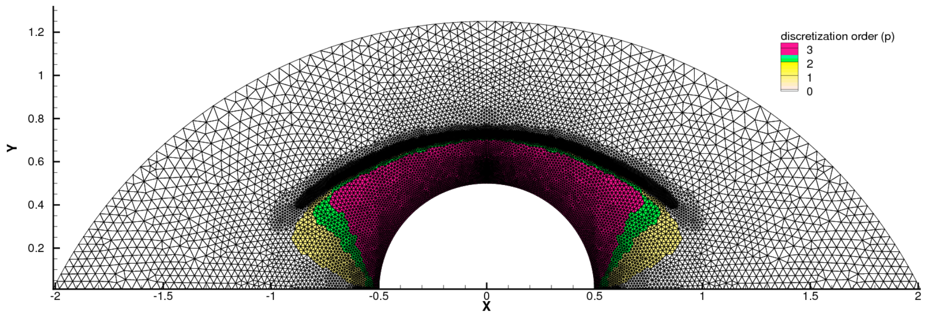
which are not close to the cylinder have little influence on the functional of interest. In addition, a thin shock profile upstream of the cylinder is properly captured and resolved on the final *hp*-adapted mesh as illustrated by the final pressure and Mach number contours in Fig. 4.19. Fig. 4.20 compares the shock profiles on the initial mesh and each of the three adapted meshes as a plot of pressure along the centerline $x = 0$. The original shock profile is resolved over a relatively wide distance, while the shock computed on the final adapted mesh is around 10 times thinner than the one resolved on the initial mesh. Note that in order to obtain a similar scale of the shock profile, the original mesh would require three successive full refinements, which produces a mesh with over 1 million degrees of freedom, whereas the number of unknowns in terms of degrees of freedom on the final adapted mesh is only about 6% of this triple-uniformly refined mesh.



(a) 1st adapted mesh, 21372 elements, discretization orders ($p_{\min} = 0, p_{\max} = 1$)



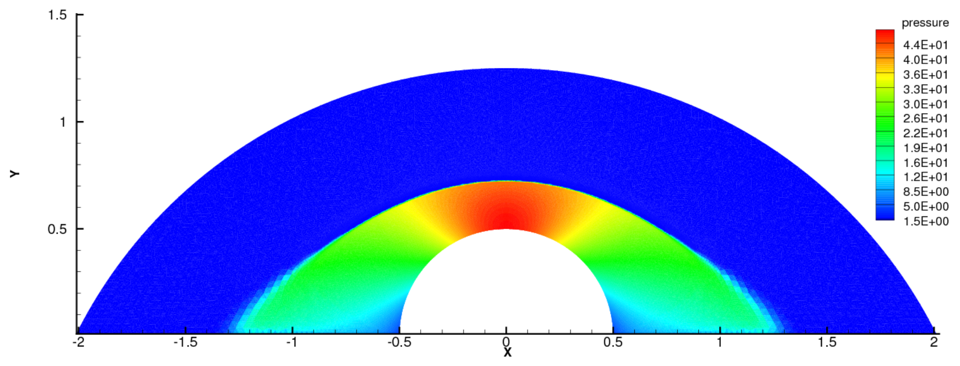
(b) 2nd adapted mesh, 29100 elements, discretization orders ($p_{\min} = 0, p_{\max} = 2$)



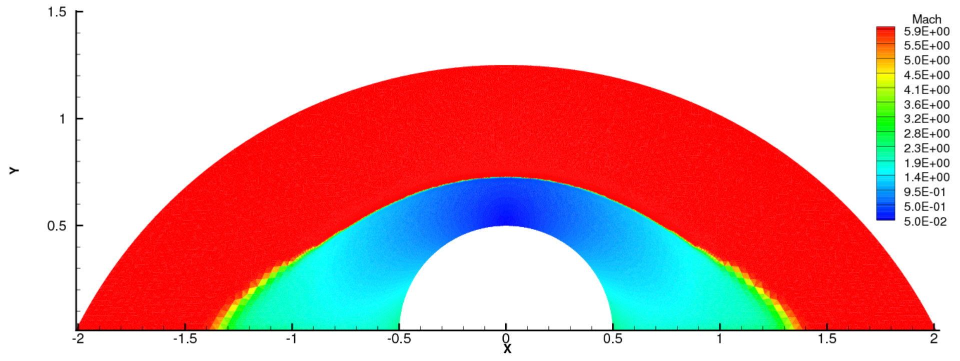
(c) 3rd adapted mesh, 42234 elements, discretization orders ($p_{\min} = 0, p_{\max} = 3$)

Figure 4.18: *hp*-adapted meshes with the distributions of orders of discretization for hypersonic flow (Mach 6) passing a half-circular cylinder.

Fig. 4.21 depicts the functional convergence for this test case. Although on the initial mesh ($p = 0$) the corrected functional represented by the first point (star symbol) of the green line does not predict the finer level functional very well, it does provide a notable improvement in the correct direction. Furthermore, the difference between the adapted functional value and corrected functional value significantly decreases with increasing adaptation cycles, and the last (third) corrected functional



(a) Pressure solution



(b) Much number solution

Figure 4.19: Pressure and Mach number contours on the final *hp*-adapted mesh for hypersonic flow (Mach 6) passing a half-circular cylinder.

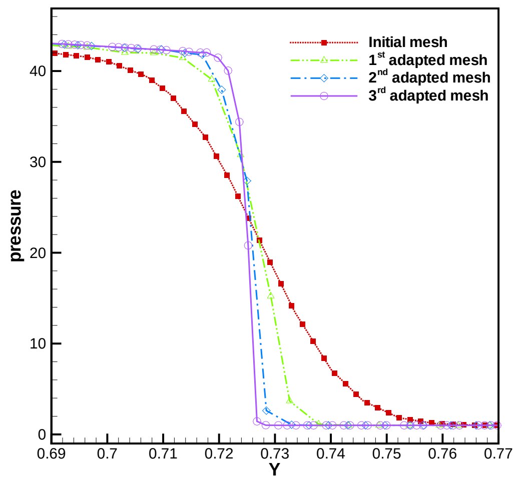


Figure 4.20: Comparison of pressure profiles at the centerline $x = 0$ between the initial mesh and *hp*-adapted meshes for hypersonic flow over half-circular cylinder (Mach 6).

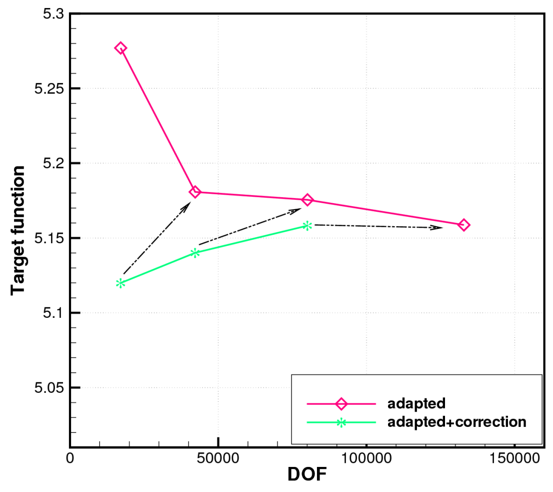


Figure 4.21: Comparison of functional convergence in terms of degrees of freedom (DOF) for hypersonic flow passing half-circular cylinder (Mach 6).

value predicts the final functional value very accurately.

4.5 Modified p-Multigrid Scheme

The solutions of the flow and adjoint problems in the previous test cases are obtained using the standard hp-Multigrid approach discussed in Section 2.2.3. As mentioned previously, the p- or hp-Multigrid approach begins with an approximation level represented by the highest discretization order p, and coarser approximation levels consist of lower-order discretizations, such as $p-1$, $p-2$, down to $p=0$. This approach applies naturally to cases with a uniform discretization order (p) of the domain. However, a discretization generated using either p- or hp-refinement results in the presence of various orders of discretization on the same mesh, such as the adapted meshes shown in Figures 4.11, 4.14 and 4.18. This complicates the implementation of the p-Multigrid approach, which assumes the presence of a uniform discretization order p on each multigrid level.

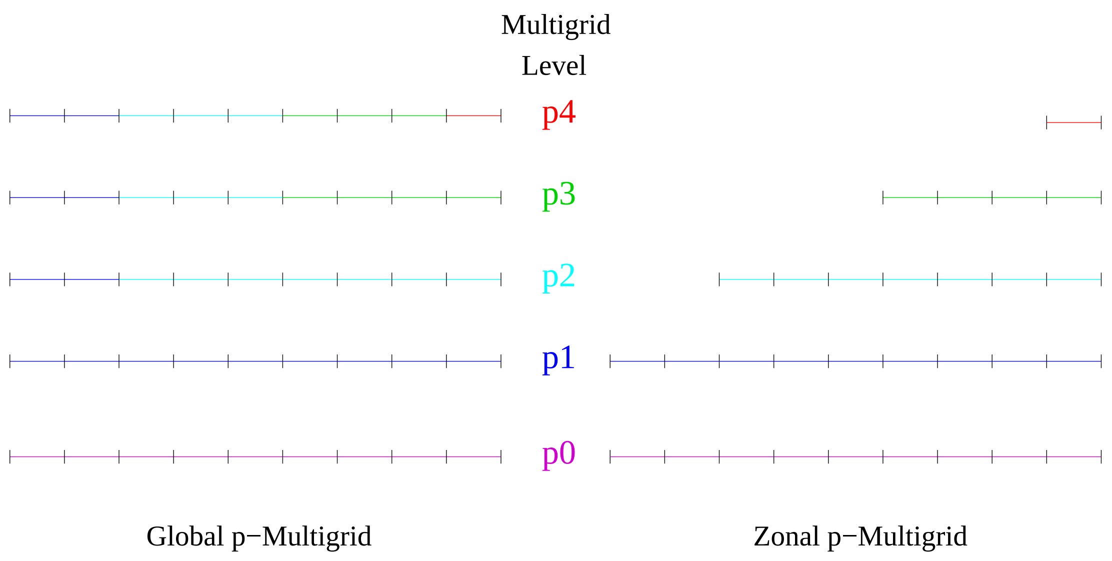


Figure 4.22: Illustration of global and zonal multigrid strategies using a one-dimensional mesh. Various colors indicate various orders of discretization: red: $p = 4$, green: $p = 3$, light blue: $p = 2$, blue: $p = 1$ and purple: $p = 0$.

4.5.1 Global versus Zonal *p*-Multigrid

A straight-forward implementation of the p-Multigrid solution scheme for cases with variable p-order discretization can be found in the "Global p-Multigrid" strategy. In this approach, the finest multigrid level consists of the entire variable p-order discretizations of the problem to be solved (i.e. Fig. 4.11 in our example test case). The next coarser level is obtained by reducing the order of the highest p-order discretization elements to $p - 1$, while retaining the same discretization orders of all other elements. This procedure is applied recursively until a uniform $p = 0$ level is obtained. The "Global-p-Multigrid" scheme is illustrated for the one-dimensional case in Fig. 4.22, where the finite element mesh contains elements of various discretization orders ranging from $p = 4$ to $p = 1$. The drawback of this approach lies in the fact that certain elements of the mesh appear on multiple coarser levels with unchanged discretization orders. Because the efficiency of the multigrid scheme is predicated on the elimination of specific error frequencies on each level, this can be expected to result in additional unnecessary computations, thus reducing overall efficiency.

To overcome these drawbacks, we propose an alternate "Zonal p-Multigrid" strategy. This approach is also illustrated in Fig. 4.22 for the one-dimensional case. In the "Zonal p-Multigrid" strategy, each finest multigrid level contains only the mesh elements of the highest discretization order p. The next coarser level contains only the mesh elements of order $p-1$, in addition to the elements from the preceding fine level p, which are now reduced to discretization order $p-1$. This process is applied recursively down to a uniform $p=0$ level. In the "Zonal p-Multigrid" approach, each level contains a uniform discretization order. Furthermore, mesh elements of a given discretization order may appear only once on a given level, and are not recalculated at the same order on other coarser levels as in the "Global-p-Multigrid" approach. However, individual finer levels in the "Zonal p-Multigrid" approach no longer contain the entire computational domain, leading to the description of this scheme as a "zonal" approach. This zonal multigrid scheme has previously been demonstrated for low-order finite-difference [105] and finite volume schemes [82], although their extension to p-Multigrid schemes has not been previously investigated. As in previous zonal multigrid work, in order to ensure that the final discretization on the union of all zonal grid levels reproduces exactly the finest level global multigrid discretization, additional fringe elements for each zonal level must be retained. In particular, if one element, in which the discretization order is lower than p (e.g. $p-1$), contains a direct neighboring element of discretization order p, this element is included in the solution approximation at multigrid level of p.

4.5.2 Computational Comparisons

As pointed out previously, the zonal multigrid method can be implemented by storing only qualified elements for each multigrid level in order to save computational time. The previous case of p-adaptation for subsonic flow over a four-element airfoil is revisited as an example for comparing the performance of the zonal multigrid method against the global multigrid method. Here, the test case starts with the final p-adapted finite-element mesh shown in Fig. 4.11, and implements both solution strategies. A uniform initial condition and the same flow conditions as described earlier in the p-adaptation case are used. Since the highest discretization order is

Table 4.3: Number of elements and edges required at each level of the zonal p-Multigrid for the p-adapted mesh illustrated in Fig. 4.11.

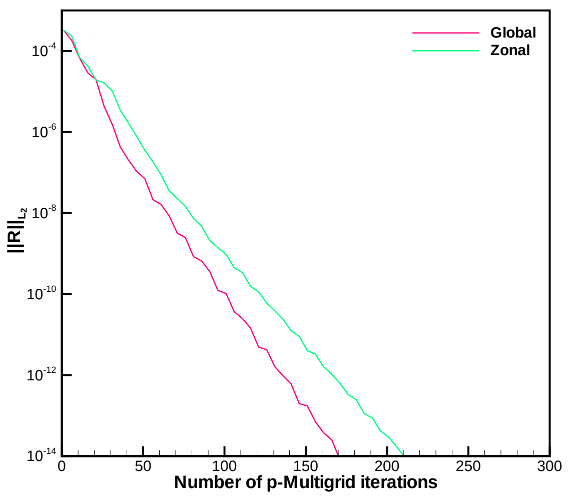
	p_4	p_3	p_2	p_1 (entire mesh)	p_0 (entire mesh)
Elements	137	402	1428	1508	1508
Edges	251	686	2214	2314	2314

Table 4.4: Computational cost (sec) for one zonal multigrid cycle and one global multigrid cycle.

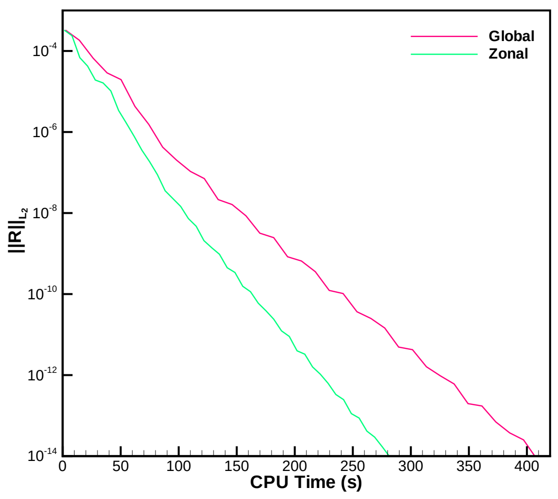
	Zonal	Global	Ratio
One multigrid cycle	1.3803	2.4256	0.5960

$p = 4$ in this case, both zonal and global multigrid methods begin with the finest level, $p = 4$, and employ coarser levels with $p = 3, 2, 1$ and $p = 0$ sequentially.

Table 4.3 provides the numbers of elements and edges or faces involved in each level of the zonal p-Multigrid strategy for this mesh. Note that the levels of $p = 4$ and $p = 3$ contain only 9% and 27% of the total number of elements, respectively, thereby saving significant computational operations. Fig. 4.23(a) plots the residual convergence histories of both methods with respect to the number of multigrid cycles. It can be observed that the zonal p-Multigrid scheme delivers a slightly slower convergence rate than the global p-Multigrid scheme due to omission of repeated smoothing passes on lower order elements appearing on multiple levels in the global multigrid approach. However, in terms of computational cost as depicted in Fig. 4.23(b), the zonal scheme is seen to be more efficient, demonstrating 37% of savings in CPU time, compared with the global multigrid approach. This is achieved by a decreased number of elements, and edges (as listed in Table 4.3) on the finer multigrid levels and thus lower required number of operations per multigrid cycle. Table 4.4 shows that the computational cost for one zonal multigrid cycle is approximately 56.9% of the cost of one global multigrid cycle. In summary, the zonal multigrid method provides an avenue to improve the multigrid solver performance for problems with variable p-order discretizations. However, this benefit relies on the specific construction of the multigrid mesh levels, such as the proportion of high-order discretized elements against lower-order elements in the computational domain, and requires the inclusion of extra fringe elements with lower-order discretizations. Therefore, in general, the



(a) Residual convergence vs. multigrid iterations



(b) Residual convergence vs. computational cost

Figure 4.23: Comparison of convergence histories of the global and zonal multigrid methods for the case of subsonic flow over a four-element airfoil.

performance benefit of the zonal p-Multigrid approach will be highly dependent on the distribution of the discretization orders for each given problem.

Chapter 5

High-order Implicit Temporal Schemes for Unsteady Flows

The emphases of the last three chapters are given to steady-state problems and this chapter concentrates on extending the techniques carried out for steady-state problems to unsteady compressible flow problems. Specifically, efficient solution techniques for high-order accurate temporal schemes are developed and incorporated with high-order accurate spatial discontinuous Galerkin discretizations. Implicit time-integration techniques are considered exclusively in order to avoid the stability restrictions of explicit methods. Standard Backwards differencing methods (BDF1 and BDF2) as well as a second-order Crank-Nicholson (CN2) and a fourth-order implicit Runge-Kutta (IRK4) scheme are considered in an attempt to balance the spatial and temporal accuracy of the overall approach. The implicit system arising at each time step is solved using the p-Multigrid approach described in Section 2.2.3, which is shown to produce both h independent convergence rates, while remaining relatively insensitive to the time-step size. The Crank-Nicholson methodology, although not strictly L-stable, demonstrates superior performance compared to the BDF2 scheme for the problems chosen in this work. However, the fourth-order accurate implicit Runge-Kutta scheme is found to be the most efficient in terms of computational cost for a given accuracy level as compared to the lower order schemes, in spite of the added cost per time step, and the benefits of this scheme increase for tighter error tolerances.

5.1 Introduction

For time-dependent problems, discontinuous Galerkin methods have generally been used in conjunction with high-order accurate explicit time-integration methods, such as explicit Runge-Kutta discontinuous Galerkin (RKDG) methods [13, 18, 52]. While such methods are well suited for problems with similar spatial and temporal scales, they are notoriously inefficient for problems with disparate temporal and spatial scales, such as low reduced frequency problems, and for steady-state problems.

The CFL stability limit of explicit schemes applied to high-order spatial discretizations becomes more restrictive as the order of the spatial discretization increases. In general, the stability limit can be as severe as CFL $\sim \frac{1}{p^2}$ [106], where p represents the polynomial degree of the basis functions. As a consequence, implicit time-integration strategies, which are unconditionally stable and allow the selection of the time step based purely on temporal accuracy considerations [54–56], can be expected to provide a more effective approach for problems with disparate length and time scales or stiff problems. However, implicit methods require the solution of one or more non-linear problems at each time step, thus requiring the use of an efficient solution technique in order to make these schemes competitive. Recently, various efforts have investigated the use of high-order implicit time-integration methods [55, 57, 58, 107] and solution procedures.

This work considers implicit time-integration approaches of various orders, namely the first and second-order implicit Backwards Differencing schemes (BDF1 and BDF2), the second-order Crank-Nicholson scheme (CN2), and an implicit fourth-order Runge-Kutta scheme (IRK4), for solving the Euler equations with high-order spatial discontinuous Galerkin discretizations. In order to provide a competitive approach, the implicit system arising at each time-step in these time-integration schemes must be solved in an efficient manner. This is achieved using the spectral multigrid (p-Multigrid) method developed for steady-state problems in Section 2.2.3.

The remainder of this chapter is organized as follows. Section 5.2 describes the temporal discretizations used in this work. Section 5.3 briefly describes the implicit solution techniques developed in this work. In Section 5.4, two-dimensional numerical results are shown for an isentropic convecting vortex case, and for a periodic vortex-shedding problem using an unstructured triangular mesh to examine the performance of the solution strategy with regards to mesh size and time-step size, and to examine the accuracy and efficiency of higher-order temporal discretizations, with particular emphasis on the suitability of these schemes in terms of accuracy and efficiency for spatial discretizations of various orders.

5.2 Temporal Discretizations

The time-dependent formulation incorporated with the spatial discontinuous Galerkin discretizations for the two-dimensional inviscid compressible Euler equations is shown in Eq. (2.5). By substituting the Galerkin solution expansion formulation denoted in Eq. (2.7), Eq. (2.5) becomes:

$$M\frac{d\tilde{\mathbf{u}}}{dt} + \mathbf{R}(\tilde{\mathbf{u}}) = 0 \tag{5.1}$$

where $\mathbf{R}$ represents the spatial residual described in Eq. (2.15), and M denotes the mass matrix which has identical diagonal blocks, M_d, for each of the four modal variables of $(\rho, \rho u, \rho v, \rho e)_i$, and which can be written as:

$$M_d = \begin{pmatrix} \int_{\Omega_k}\phi_1\phi_1 dV & \int_{\Omega_k}\phi_1\phi_2 dV & \dots & \int_{\Omega_k}\phi_1\phi_N dV \\ \int_{\Omega_k}\phi_2\phi_1 dV & \int_{\Omega_k}\phi_2\phi_2 dV & \dots & \int_{\Omega_k}\phi_2\phi_N dV \\ \dots & \dots & \dots & \dots \\ \int_{\Omega_k}\phi_N\phi_1 dV & \int_{\Omega_k}\phi_N\phi_2 dV & \dots & \int_{\Omega_k}\phi_N\phi_N dV \end{pmatrix} \tag{5.2}$$

The implicit time-integration schemes currently employed in this work range from first to fourth-order accurate in time, including both first and second-order accurate multistep backwards difference formulations (BDF1 and BDF2), the second-order accurate Crank-Nicolson or trapezoidal scheme, and a fourth-order accurate implicit multistage Runge-Kutta scheme (IRK4). The BDF1 and BDF2 schemes are both unconditionally stable, while the CN2 scheme is A-stable, but not L-stable [108]. For these reasons, CN2 has often been shunned in favor of BDF2 in many computational fluid dynamics problems. However, the lack of L-stability may be acceptable particularly for problems with smooth solutions, and the scheme is therefore included in the present study. Because higher-order multistep backwards difference schemes beyond second-order are not A-stable, we choose to investigate the use of implicit Runge Kutta schemes for achieving higher temporal accuracy. A six-stage diagonally implicit IRK scheme is chosen which is fourth-order accurate in time. While this scheme may not necessarily represent the optimal fourth-order temporal scheme for all problems, it has been designed with stiff stability and accuracy considerations in mind [109], and has been used successfully on lower-order finite-volume schemes by various authors [55, 110]. One of the drawbacks of IRK methods is their expense, since these require the solution of multiple implicit problems at each time step (one per stage), as opposed to BDF and CN schemes which only require the solution of a single implicit problem per time step. Therefore, one of our objectives is to determine if IRK schemes can be competitive or superior to lower-order schemes when used in conjunction with efficient solvers, particularly when high accuracy is required.

Starting from the set of ordinary differential equations given by Eq. (5.1), the formulations for BDF1, BDF2 and CN2 schemes are given respectively as:

$$\text{BDF1}: \quad \frac{M}{\Delta t}\left(\tilde{\mathbf{u}}^{n+1} - \tilde{\mathbf{u}}^{n}\right) + \mathbf{R}\left(\tilde{\mathbf{u}}^{n+1}\right) = 0 \tag{5.3}$$

$$\text{BDF2}: \quad \frac{M}{\Delta t}\left(\frac{3}{2}\tilde{\mathbf{u}}^{n+1} - 2\tilde{\mathbf{u}}^{n} + \frac{1}{2}\tilde{\mathbf{u}}^{n-1}\right) + \mathbf{R}\left(\tilde{\mathbf{u}}^{n+1}\right) = 0 \tag{5.4}$$

$$\text{CN2}: \quad \frac{M}{\Delta t}\left(\tilde{\mathbf{u}}^{n+1} - \tilde{\mathbf{u}}^{n}\right) + \frac{1}{2}\mathbf{R}\left(\tilde{\mathbf{u}}^{n+1}\right) + \frac{1}{2}\mathbf{R}\left(\tilde{\mathbf{u}}^{n}\right) = 0 \tag{5.5}$$

where Δt represents the integration time step, and $\tilde{\mathbf{u}}^{n}$ and $\tilde{\mathbf{u}}^{n+1}$ denote numerical solutions for the current and the next (unknown) time step, respectively. By defining a nonlinear unsteady residual, $\mathbf{R}_e$, for the corresponding BDF1, BDF2 and CN2 schemes as:

$$\text{BDF1}: \mathbf{R}_e(\tilde{\mathbf{u}}^{n+1}) = \frac{M}{\Delta t}\tilde{\mathbf{u}}^{n+1} + \mathbf{R}(\tilde{\mathbf{u}}^{n+1}) - \frac{M}{\Delta t}\tilde{\mathbf{u}}^{n} = 0 \tag{5.6}$$

$$\text{BDF2}: \mathbf{R}_e(\tilde{\mathbf{u}}^{n+1}) = \frac{M}{\Delta t}(\frac{3}{2}\tilde{\mathbf{u}}^{n+1}) + \mathbf{R}(\tilde{\mathbf{u}}^{n+1}) - \frac{M}{\Delta t}(2\tilde{\mathbf{u}}^{n} - \frac{1}{2}\tilde{\mathbf{u}}^{n-1}) = 0 \tag{5.7}$$

$$\text{CN2}: \mathbf{R}_e(\tilde{\mathbf{u}}^{n+1}) = \frac{M}{\Delta t}\tilde{\mathbf{u}}^{n+1} + \frac{1}{2}\mathbf{R}(\tilde{\mathbf{u}}^{n+1}) - (\frac{M}{\Delta t}\tilde{\mathbf{u}}^{n} - \frac{1}{2}\mathbf{R}(\tilde{\mathbf{u}}^{n})) = 0 \tag{5.8}$$

the solution of these schemes at the time step $n+1$ can be achieved by solving the nonlinear problems $\mathbf{R}_e(\tilde{\mathbf{u}}^{n+1}) = 0$. These schemes are relatively efficient because they solve only one implicit set of equations per time step. In the case of multistage implicit Runge-Kutta schemes, however, multiple implicit problems are required per time step (each stage requires one implicit system solution), but these schemes are easily implemented in the presence of variable time steps and can be constructed to be A- and L-stable for any temporal order. In the current work a particular class of Explicit first stage, Single Diagonal coefficient, diagonally Implicit Runge-Kutta (ESDIRK scheme), is considered. The formula for a $\mathcal{S}$-stage ESDIRK scheme can be written as:

$$\begin{array}{ll} (i) & \tilde{\mathbf{u}}^{(0)} = \tilde{\mathbf{u}}^{n} \\ (ii) & For\ s = 1, \cdots, \mathcal{S} \\ & \tilde{\mathbf{u}}^{(s)} = \tilde{\mathbf{u}}^{n} - \Delta t \sum_{j=1}^{s} a_{sj} M^{-1}\mathbf{R}(\tilde{\mathbf{u}}^{(j)}) \\ (iii) & \tilde{\mathbf{u}}^{n+1} = \tilde{\mathbf{u}}^{n} - \Delta t \sum_{j=1}^{\mathcal{S}} b_j M^{-1}\mathbf{R}(\tilde{\mathbf{u}}^{(j)}) \end{array} \tag{5.9}$$

where a_{sj} are the Butcher coefficients of the scheme. The Butcher table for the six-stage ESDIRK scheme (i.e. $\mathcal{S} = 6$, fourth-order accurate) employed presently is shown in Table 5.1 and the values [55,110] are given in Table 5.2. The set of coefficients, a_{sj} in Eq. (5.9), defines the implicit RK schemes. The first stage is explicit due to $a_{11} = 0$ and a single implicit scheme is solved at each additional individual stage since the set of a_{kj} has the form of a lower triangular matrix. The last stage coefficients take on the form $a_{\mathcal{S}j} = b_j$, and thus the solution for the next time step is equal to the solution at the last stage, i.e. $\tilde{\mathbf{u}}^{n+1} = \tilde{\mathbf{u}}^{(\mathcal{S})}$. c_k represents the point in the time interval, $[t, t+\Delta t]$ and satisfies:

Table 5.1: Butcher Tableau for ESDIRK class of six-stage RK schemes.

$c_1 = 0$	0	0	0	0	0	0
c_2	a_{21}	$a_{22} = a_{66}$	0	0	0	0
c_3	a_{31}	a_{32}	$a_{33} = a_{66}$	0	0	0
c_4	a_{41}	a_{42}	a_{43}	$a_{44} = a_{66}$	0	0
c_5	a_{51}	a_{52}	a_{53}	a_{54}	$a_{55} = a_{66}$	0
$c_6 = 1$	$a_{61} = b_1$	$a_{62} = b_2$	$a_{63} = b_3$	$a_{64} = b_4$	$a_{65} = b_5$	a_{66}
$\tilde{\mathbf{u}}^{n+1}$	b_1	b_2	b_3	b_4	b_5	b_6

Table 5.2: The Butcher coefficients a_{sj} and c_s for the ESDIRK4 scheme (with $a_{6j} = b_j$.

a_{21}	a_{22}	a_{31}	a_{32}
a_{33}	a_{41}	a_{42}	a_{43}
a_{44}	a_{51}	a_{52}	a_{53}
a_{54}	a_{55}	a_{61}	a_{62}
a_{63}	a_{64}	a_{65}	a_{66}
c_1	c_2	c_3	c_4
c_5	c_6		

$\frac{1}{4}$	$\frac{1}{4}$	$\frac{8611}{62500}$	$\frac{-1743}{31250}$
$\frac{1}{4}$	$\frac{5012029}{34652500}$	$\frac{-654441}{2922500}$	$\frac{174375}{388108}$
$\frac{1}{4}$	$\frac{15267082809}{155376265600}$	$\frac{-71443401}{120774400}$	$\frac{730878875}{902184768}$
$\frac{2285395}{8070912}$	$\frac{1}{4}$	$\frac{82889}{524892}$	0
$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
0	$\frac{1}{2}$	$\frac{83}{250}$	$\frac{31}{50}$
$\frac{17}{20}$	1		

$$c_k = \sum_{j=1}^{k} a_{kj} \qquad (k = 1, 2, \dots, 6) \tag{5.10}$$

Similarly, the unsteady residual corresponding to the non-linear implicit system at each stage of the IRK4 scheme can be written as:

$$\begin{aligned}\mathbf{R}_e(\tilde{\mathbf{u}}^{(1)}, \cdots, \tilde{\mathbf{u}}^{(s)}) &= \frac{M}{\Delta t}\tilde{\mathbf{u}}^{(s)} + a_{ss}\mathbf{R}(\tilde{\mathbf{u}}^{(s)}) \\ &- \left[\frac{M}{\Delta t}\tilde{\mathbf{u}}^{n} - \sum_{j=1}^{s-1} a_{sj}\mathbf{R}(\tilde{\mathbf{u}}^{(j)})\right] = 0 \quad (s = 1, \cdots, 6)\end{aligned} \tag{5.11}$$

5.3 Solution Approach

As pointed out previously, implicit time-integration methods require the solution of one or more implicit problems per time step. Efficient solvers are required for this task in order to achieve an overall competitive approach. Our approach consists of using the *p*-Multigrid approach alone described in Section 2.2.3, in conjunction with an element Jacobi smoother on each multigrid level, for solving the implicit system at each time step. The use of *p*-Multigrid alone rather than the *hp*-Multigrid approach is due to the fact that the addition of *h*-Multigrid can be expected to be less important for time-dependent problems, since the resulting implicit systems are more diagonally dominant and are somewhat more local in nature than the corresponding steady state problem. The extension to *hp*-Multigrid can be employed naturally as described in Section 2.2.3.

Returning to the unsteady residual formulation, $\mathbf{R}_e(\tilde{\mathbf{u}}^{n+1})$, for solving the flow solution at time step $n+1$ in the implicit schemes, a Newton scheme can be expressed in a similar way to that used for the steady-state problem as:

$$\begin{aligned}&(i) \quad \left[\frac{\partial \mathbf{R}_e(\tilde{\mathbf{u}})}{\partial \tilde{\mathbf{u}}}\right]^k \Delta\tilde{\mathbf{u}}^{k+1} = -\mathbf{R}_e(\tilde{\mathbf{u}}^k) \\ &(ii) \quad \tilde{\mathbf{u}}^{k+1} = \tilde{\mathbf{u}}^k + \alpha\Delta\tilde{\mathbf{u}}^{k+1}, \quad k = 1, 2, \cdots, \mathcal{V} \\ &(iii) \quad \tilde{\mathbf{u}}^{n+1} = \tilde{\mathbf{u}}^{\mathcal{V}} \quad \text{when} \quad \mathbf{R}_e(\tilde{\mathbf{u}}^{\mathcal{V}}) = 0\end{aligned} \tag{5.12}$$

where k refers to the subiteration index in the Newton iterations. $[\partial \mathbf{R}_e(\tilde{\mathbf{u}})/\partial \tilde{\mathbf{u}}]$ is the Jacobian matrix for the unsteady flow problem and its formulation can be derived based on Equations (5.6)-(5.8) and Eq. (5.11) for the respective BDF1, BDF2, CN2 and IRK4 schemes considered in this work. α is a relaxation parameter similarly employed in the steady-state solver, which is designed to keep $\|\alpha\Delta\tilde{\mathbf{u}}^{k+1}\|_{L_\infty}/\|\tilde{\mathbf{u}}^{k+1}\|_{L_\infty} \leq 10\%$. The current solution method considers the use of the element Jacobi smoothers investigated for the steady state problem (in Section 2.2.2), including the non-linear element Jacobi (NEJ), the quasi-non-linear element Jacobi (qNEJ) and the linearized element Jacobi (LEJ) solvers. As described previously, the NEJ and qNEJ smoothers store only the Jacobian entries corresponding to the modal coupling between all modes within an element, resulting in a block diagonal matrix, $[D_T]$, which is easily inverted using Gaussian elimination at the block level, and all other entries, $[O_T]$ are discarded. In the LEJ solver, the full Jacobian entries are decomposed into both diagonal, $[D_T]$, and off-diagonal, $[O_T]$, block components, i.e. $[\partial \mathbf{R}_e(\mathbf{u})/\partial \mathbf{u}]^k = [D_T]^k + [O_T]^k$, where the $[D_T]$ blocks are treated implicitly, and the $[O_T]$ blocks are treated explicitly. Note that the diagonal block matrix $[D_T]^k$, the off-diagonal blocks $[O_T]^k$ and the unsteady residual $\mathbf{R}_e$ will have different forms for the various time-integration schemes. To make this point clear, we specify in Table 5.3 these terms for the corresponding BDF1, BDF2 and CN2 schemes, and for intermediate stage values of the IRK4 scheme, respectively. $\mathbf{R}(\tilde{\mathbf{u}}^k)$ in this table refers to the spatial residual vector corresponding to Eq. (2.15). The respective notations of $[D]^k$ and $[O]^k$ denote diagonal and off-diagonal block components of the steady-state Jacobian matrix, evaluated using $\tilde{\mathbf{u}}^k$.

The procedure used for implementing the p-Multigrid approach for the steady-state problem in Section 2.2.3 can also be extended to the time-dependent problem at each implicit time step (e.g. $n+1$), described as:

- **Step 1:** Start with $\tilde{\mathbf{u}}^l = \tilde{\mathbf{u}}^n$.

- **Step 2:** Perform $\mathcal{M}$ subiterations ($k = 1, \cdots, \mathcal{M}$) on the high-order approximation level (p) to solve the problem: $\mathbf{R}_{\mathbf{e}_p}(\tilde{\mathbf{u}}_p^k) = \mathbf{S}_p$, using an element Jacobi solver mentioned previously; Get residual: $\mathbf{r}_p = \mathbf{S}_p - \mathbf{R}_{e_p}(\tilde{\mathbf{u}}_p^{\mathcal{M}})$.

- **Step 3:** Restrict both solution and residual to the low-order approximation level $(p-1)$: $\tilde{\mathbf{u}}_{p-1}^0 = I_p^{p-1}\tilde{\mathbf{u}}_p^{\mathcal{M}}$; $\mathbf{S}_{p-1} = \mathbf{R}_{e_{p-1}}(\tilde{\mathbf{u}}_{p-1}^0) + I_p^{p-1}\mathbf{r}_p$.

- **Step 4:** Solve the low-order approximation level problem by using the same element Jacobi solver with $\mathcal{M}$ subiterations $(k = 1, \cdots, \mathcal{M})$: $\mathbf{R}_{e_{p-1}}(\tilde{\mathbf{u}}_{p-1}^k) = \mathbf{S}_{p-1}$; Obtain the low-order level error: $e_{p-1} = \tilde{\mathbf{u}}_{p-1}^{\mathcal{M}} - I_p^{p-1}\tilde{\mathbf{u}}_p^{\mathcal{M}}$.

- **Step 5:** Prolongate this low-order level error to correct the high-order approximation level: $\tilde{\mathbf{u}}_p^{l+1} = \tilde{\mathbf{u}}_p^{\mathcal{M}} + I_{p-1}^p e_{p-1}$.

- **Step 6:** Repeat Steps 2 through 5 for each level of the V-cycle p-Multigrid method until $\mathbf{R}_{e_p}(\tilde{\mathbf{u}}_p^{l+1}) = 0$ (machine zero or a suitably determined tolerance), then the solution for the $n+1$ time step is obtained as: $\tilde{\mathbf{u}}^{n+1} = \tilde{\mathbf{u}}_p^{l+1}$.

Similarly, the use of the notation $(\cdot)_p$ is to specify the approximation level of the p-Multigrid approach. The source term $\mathbf{S}_p$, which represents the residual restriction term from the finer multigrid levels, vanishes on the finest level (highest-order approximation level) in the multigrid formulation, but is retained on all levels in the description for generality. The two-level multigrid scheme described above is used recursively to solve the coarse level problem, resulting in the full multilevel algorithm. Furthermore, five subiterations are utilized in both the qNEJ and LEJ schemes between non-linear Jacobian (and residual for the LEJ method) updates.

5.4 Results

In this section, two test cases are used to illustrate the performance of the solution techniques for the time-dependent compressible Euler equations. The first test case consists of an isentropic convecting vortex on a uniform mesh, while the second test case consists of the time-dependent vortex shedding from a triangular wedge on a highly graded unstructured mesh.

Table 5.3: Diagonal blocks, off-diagonal blocks and unsteady residuals for the BDF1, BDF2, CN2 and IRK4 schemes.

Schemes	$[D_T]^k$	$[O_T]^k$	$\mathbf{R}_e(\tilde{\mathbf{u}}^k)$
BDF1	$[D_T]^k = \frac{M}{\Delta t} + [D]^k$	$[O]^k$	$\frac{M}{\Delta t}\tilde{\mathbf{u}}^k + \mathbf{R}(\tilde{\mathbf{u}}^k) - \mathbf{F}$ $\mathbf{F} = \frac{M}{\Delta t}\mathbf{u}_h^n$
BDF2	$[D_T]^k = \frac{3}{2}\frac{M}{\Delta t} + [D]^k$	$[O]^k$	$\frac{3}{2}\frac{M}{\Delta t}\tilde{\mathbf{u}}^k + \mathbf{R}(\tilde{\mathbf{u}}^k) - \mathbf{F}$ $\mathbf{F} = \frac{M}{\Delta t}(2\mathbf{u}_h^n - \frac{1}{2}\mathbf{u}_h^{n-1})$
CN2	$[D_T]^k = \frac{M}{\Delta t} + \frac{1}{2}[D]^k$	$\frac{1}{2}[O]^k$	$\frac{M}{\Delta t}\tilde{\mathbf{u}}^k + \frac{1}{2}\mathbf{R}(\tilde{\mathbf{u}}^k) - \mathbf{F}$ $\mathbf{F} = \frac{M}{\Delta t}\mathbf{u}_h^n - \frac{1}{2}\mathbf{R}(\tilde{\mathbf{u}}^n)$
IRK4 $(s = 1, \cdots, 6)$	$[D_T]^k = \frac{M}{\Delta t} + a_{ss}[D]^k$	$a_{ss}[O]^k$	$\frac{M}{\Delta t}\tilde{\mathbf{u}}^k + a_{ss}\mathbf{R}(\tilde{\mathbf{u}}^k) - \mathbf{F}$ $\mathbf{F} = \frac{M}{\Delta t}\mathbf{u}_h^n - \sum_{j=1}^{s-1} a_{sj}\mathbf{R}(\tilde{\mathbf{u}}^{(j)})$

For the first test case, we concentrate on assessing the accuracy and efficiency of the various time-integration schemes in the presence of high spatial accuracy (p=4, fifth-order spatial accuracy). Accuracy is assessed on the one hand by comparing solutions using the same time step for the various schemes with each other and with an exact solution, in order to demonstrate that accuracy is within (or close to) the asymptotic region of convergence for the higher-order time-integration methods for such problems. A more quantitative time step refinement study is then performed to demonstrate the asymptotic error reduction properties of the respective temporal schemes. In order to compare the efficiency of the various schemes, the performance of the p-Multigrid solution strategy is first evaluated, including the dependencies on grid resolution and time step size, which are ultimately the key determining factors for any efficiency comparison of time-implicit schemes. The delivered accuracy of the various schemes as a function of CPU time is then examined.

The second test case represents a more realistic problem, for which an exact solution is not available. In this case, we investigate the effect of low and high-order accurate time-integration schemes combined with low and high-order spatial discretizations on highly graded meshes on overall solution accuracy. We begin by establishing the asymptotic convergence of the time-integration schemes on this more demanding problem, through a time-step refinement study, and then examine the overall solution error for various combinations of spatial and temporal accuracy. Finally, a qualitative comparison is made with a simple explicit scheme, to illustrate the advantages of time-implicit approaches for low-reduced frequency problems of this type.

5.4.1 Convection of an Isentropic Vortex

The convection of a two-dimensional inviscid isentropic vortex [111–113] is simulated to examine the performance of the implicit time-stepping schemes. The exact solution for this test case at any time t is the initial solution at $t_0 = 0$ translated over a distance $u_\infty t$ for a horizontally convecting vortex, which provides a valuable reference for measuring the accuracy of the computed solution.

The mean flow density, ρ_∞, velocity, u_∞ and v_∞, pressure, p_∞ and temperature T_∞ are taken as freestream values, which are set as $(\rho_\infty, u_\infty, v_\infty, p_\infty, T_\infty) = (1, 0.5, 0, 1, 1)$ in this test case. Freestream boundary conditions are imposed on the top and bottom boundaries, while periodic boundary conditions are applied between the inlet and outlet of the domain. These boundary conditions are applied on all levels of the multigrid sequence. At $t_0 = 0$, the flow is perturbed by an isentropic vortex $(\delta u, \delta v, \delta T)$ centered at (x_0, y_0) with the form:

$$\delta u = -\frac{\sigma}{2\pi}(y - y_0)e^{\vartheta(1-r^2)} \tag{5.13}$$

$$\delta v = \frac{\sigma}{2\pi}(x - x_0)e^{\vartheta(1-r^2)} \tag{5.14}$$

$$\delta T = -\frac{\sigma^2(\gamma - 1)}{16\vartheta\gamma\pi^2}e^{2\vartheta(1-r^2)} \tag{5.15}$$

where, ϑ and σ are parameters which determine the strength of the vortex, $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$ is the distance to the vortex center, and $\gamma = 1.4$ is the ratio of specific heats of air. In this study, we set ϑ as unity and σ as 4.0. Given the perturbation functions shown in Eq. (5.13), Eq. (5.14) and Eq. (5.15), the other resulting conservative variables can be determined based on the assumption of isentropic flow throughout the domain (i.e. $p/\rho^\gamma = 1$ and $T = p/\rho$ for a perfect gas), given as:

$$\rho = T^{1/(\gamma-1)} = (T_\infty + \delta T)^{1/(\gamma-1)} = \left[1 - \frac{\sigma^2(\gamma-1)}{16\vartheta\gamma\pi^2} e^{2\vartheta(1-r^2)}\right]^{1/(\gamma-1)} \tag{5.16}$$

$$u = u_\infty + \delta u = 0.5 - \frac{\sigma}{2\pi}(y - y_0) e^{\vartheta(1-r^2)} \tag{5.17}$$

$$v = v_\infty + \delta v = 0 + \frac{\sigma}{2\pi}(x - x_0) e^{\vartheta(1-r^2)} \tag{5.18}$$

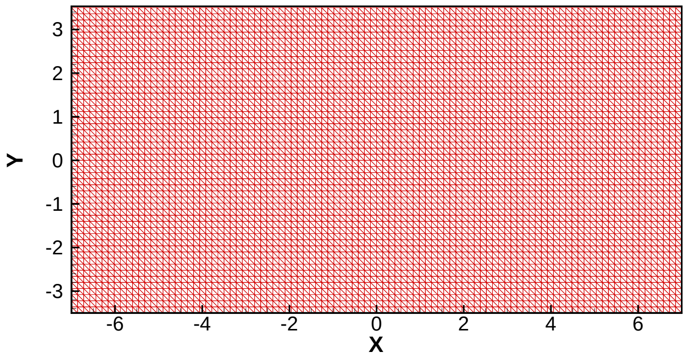
Numerical Solutions

This test case employs a uniform Cartesian triangular mesh. The initial vortex is placed at $(x_0, y_0) = (0, 0)$ on a domain of $-7 \leq x \leq 7$ and $-3.5 \leq y \leq 3.5$ with 10000 elements, as shown in Fig. 5.1. A fifth-order accurate ($p = 4$) spatial discretization is used in all cases, and the time-step (Δt) is set equal to 0.2. Since the local CFL number is defined as,

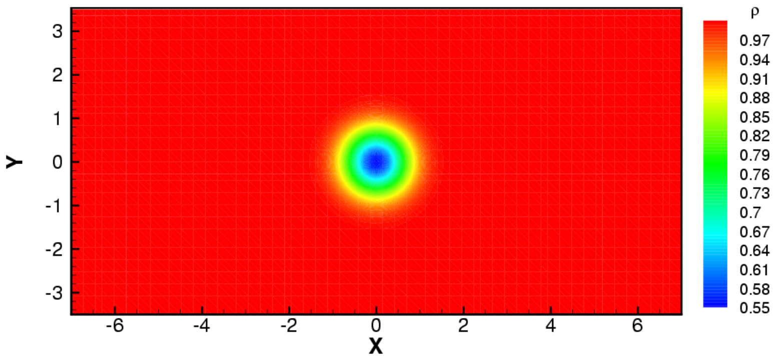
$$\text{CFL}_i = \frac{\Delta t}{vol_i} \sum_{j=1}^{3 \text{ edges}} (|\mathbf{u} \cdot \mathbf{n}| + c)_j \quad 1 \leq i \leq N \tag{5.19}$$

where vol denotes the area of the element in the 2D case, c denotes the local speed of sound and N represents the number of elements, then the fixed time-step $\Delta t = 0.2$ corresponds to a maximum CFL number of 11. Note that for the considered $p = 4$ spatial discretization, the explicit stability limit could be as much as 176 times smaller (i.e. $\sim \frac{1}{p^2}$) than our chosen time step.

Fig. 5.1 illustrates the computational mesh and the initial density contours in the domain. The length of the domain is 14, and the horizontal velocity is $u_\infty = 0.5$, thus the vortex requires $T = 28$ to complete one revolution around the periodic grid in the x-direction. Computed density solutions at times $t = 4$, $t = 10$, $t = 20$ and $t = 50$ obtained by the BDF1, BDF2, CN2 and IRK4 schemes are shown in Figures 5.2 and 5.3. The first three non-dimensional times are within the first horizontal period of the vortex motion, and the last $t = 50$ is close to the end of the second period (since it requires $t = 56$ to complete the second period). The results of the



(a) Grid

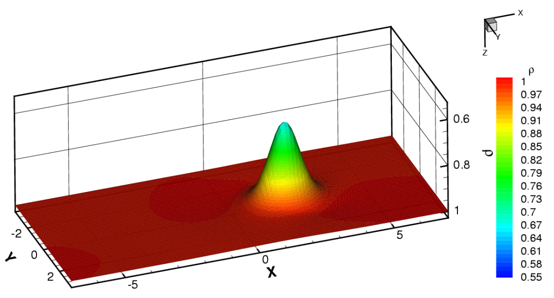


(b) Initial density contour

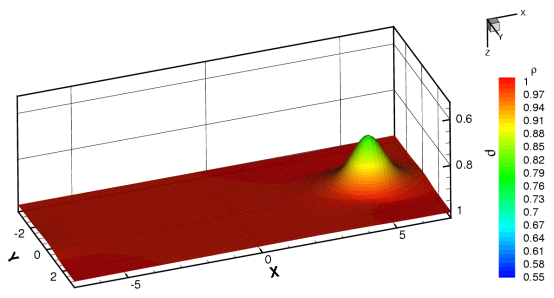
Figure 5.1: Grid and initial density contours for the isentropic vortex convection problem.

BDF1 scheme (Figures 5.2(a) – 5.2(d)) illustrate how the vortex is diffused using this first-order accurate scheme, even at small time increments. Conversely, as shown in Figures 5.2(e) – 5.2(h) and Figures 5.3(a) – 5.3(d), the BDF2 and CN2 schemes display a substantially better shape retaining property for the vortex, except at later times such as $t = 50$ where some dispersion and oscillations are evident around the brink of the vortex. On the other hand, the IRK4 scheme (Figures 5.3(e) – 5.3(h)) provides the best accuracy with the final shape of the vortex at $t = 50$ being almost indistinguishable from the initial shape.

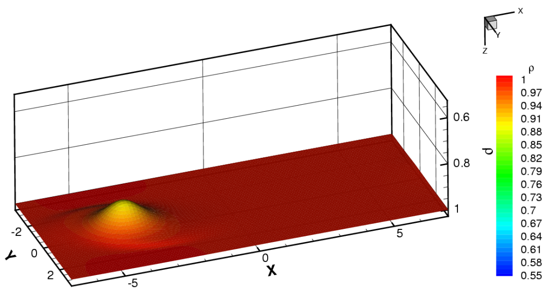
A more quantitative comparison is given in Fig. 5.4, where the density profiles along the horizontal centerline ($y = 0$) for the BDF1, BDF2, CN2 and IRK4 time-integration schemes at various times, $t = 4$, $t = 10$, $t = 20$, and $t = 50$, are compared



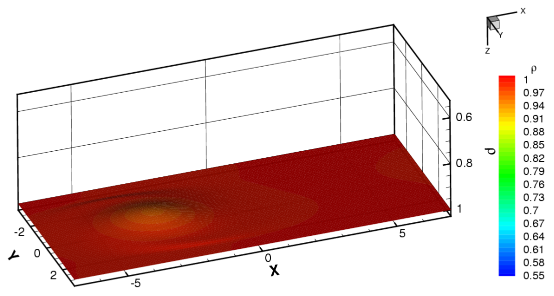
(a) t=4, BDF1



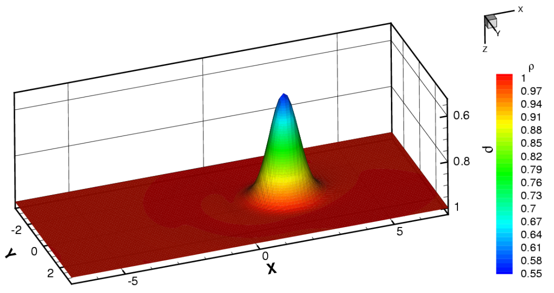
(b) t=10, BDF1



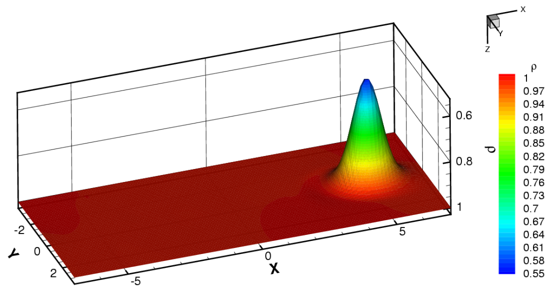
(c) t=20, BDF1



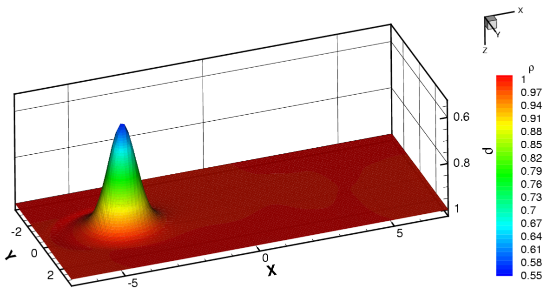
(d) t=50, BDF1



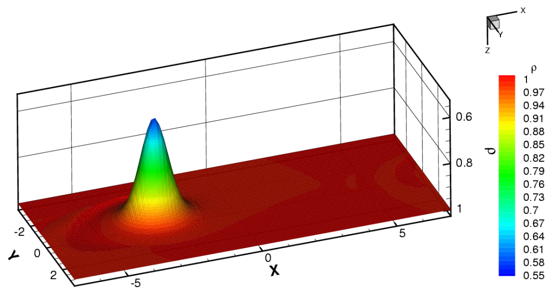
(e) t=4, BDF2



(f) t=10, BDF2

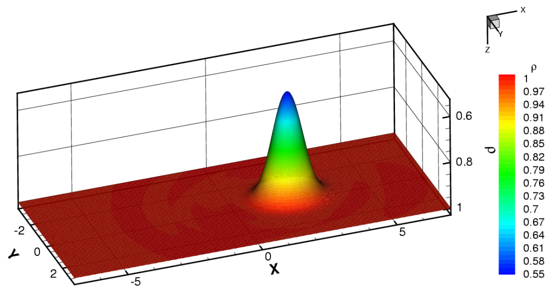


(g) t=20, BDF2

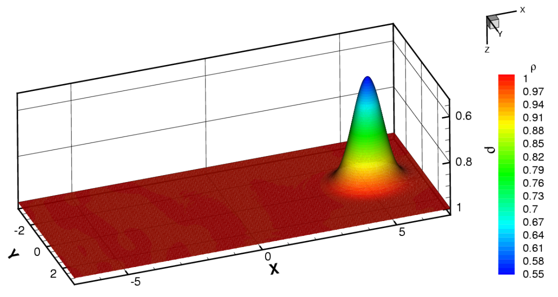


(h) t=50, BDF2

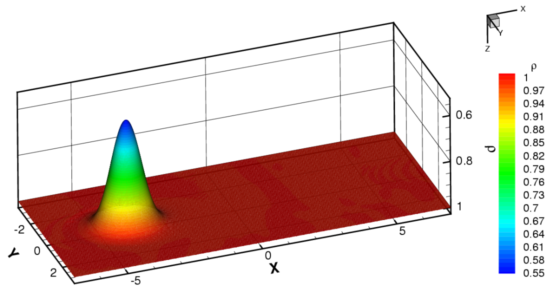
Figure 5.2: Density contours (3D) of the BDF1 and BDF2 schemes at various times, t=4, 10, 20 and 50, using a time-step of Δt=0.2 and $p = 4$ spatial discretization.



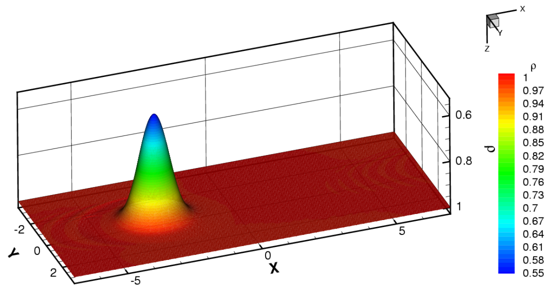
(a) t=4, CN2



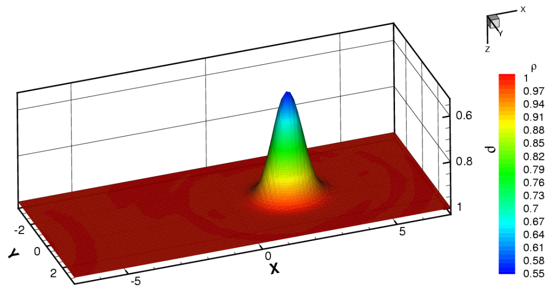
(b) t=10, CN2



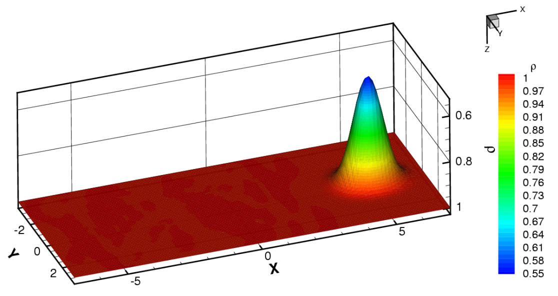
(c) t=20, CN2



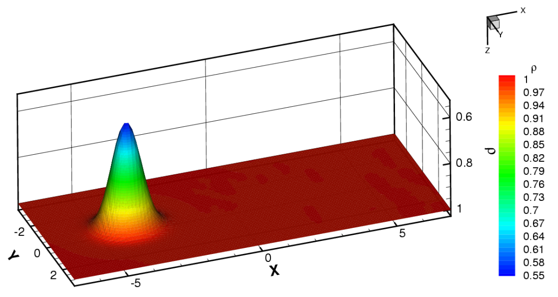
(d) t=50, CN2



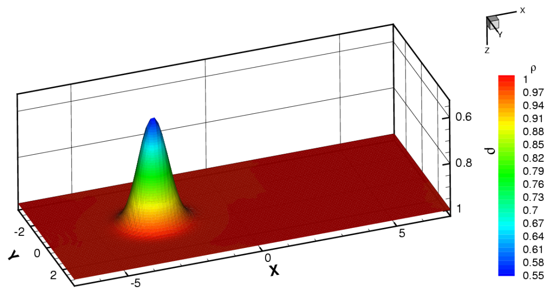
(e) t=4, IRK4



(f) t=10, IRK4

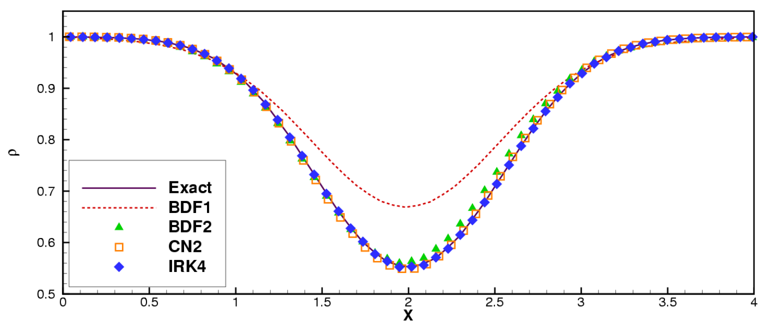


(g) t=20, IRK4

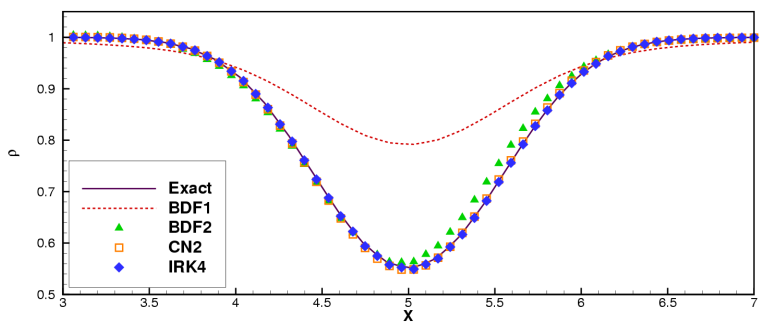


(h) t=50, IRK4

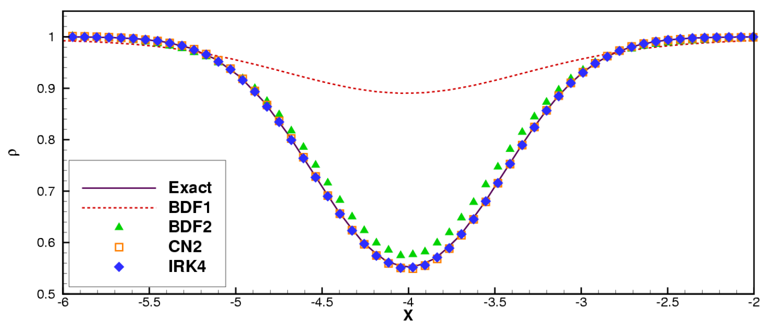
Figure 5.3: Density contours (3D) of the CN2 and IRK4 schemes at various times, t=4, 10, 20 and 50, using a time-step of Δt=0.2 and $p = 4$ spatial discretization.



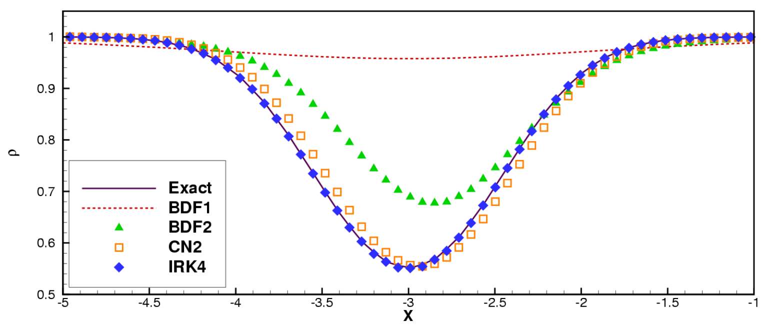
(a) t=4



(b) t=10



(c) t=20



(d) t=50

Figure 5.4: Comparison of the various temporal schemes with the exact solution, illustrated by density profiles at the centerline $y = 0$, at t=4, 10, 20 and 50, using a time-step of Δt=0.2.

with the exact solution, obtained by translating the initial centerline density profile to the appropriate spatial location for each given time. From the figure, it can be seen that the IRK4 scheme exhibits the best resolution and provides very good agreement with the exact solution, since there is no visual deviation between the computed results and exact results and the vortex core is well conserved. On the other hand, the BDF1 scheme which is only first-order accurate in time, produces rapid dissipation of the vortex core, as mentioned previously, while the BDF2 and CN2 schemes provide substantially better resolution and higher accuracy than the BDF1 scheme. Before $t = 20$, the computed results obtained by the BDF2 and CN2 schemes fall almost on top of the exact solution, but at later times, these schemes show increased deviations from the exact profile, although the CN2 scheme remains substantially more accurate than the BDF2 scheme. However, this result must be balanced by the fact that the CN2 scheme is not L-stable [108], and may perform poorly in other cases.

Temporal Accuracy

We begin by examining the error of the respective time-integration schemes at a fixed time-step size, in order to determine if these schemes are within or close to their asymptotic regions of convergence. This is important since the error properties of these schemes are asymptotic in nature, based on the presumption of smooth solutions, and there is no guarantee that higher-order methods will deliver smaller errors than lower order methods, even at equivalent time steps, when these assumptions do not hold.

In order to assess the asymptotic behavior of the time-integration schemes, a temporal refinement study is carried out using the same fixed spatial discretization ($p = 4$). Because the overall error is due to both spatial and temporal error, the "exact" numerical solution for each temporal scheme is obtained using a small time-step reference solution, in order to eliminate the effect of spatial error and to isolate the temporal error. The time-step to obtain the "exact" solution is $\Delta t = 0.01$ for all time-integration schemes. Various time-steps, consisting of $\Delta t = 2.0$, 1.0, 0.5 and 0.25, which correspond to a maximum CFL number of 110, 55, 28 and 14, respectively,

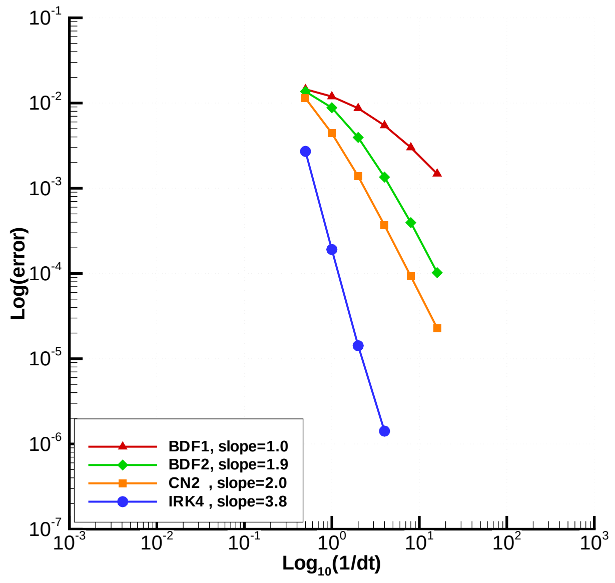


Figure 5.5: Comparisons of temporal accuracy for various implicit temporal schemes as a function of time-step size at t = 4.

have been used for all of the temporal schemes. Additionally, two smaller time-steps, $\Delta t = 0.125$ and 0.0625 are employed for the BDF1, BDF2 and CN2 schemes to extend their range of comparison. A regular mesh with a grid spacing of $\Delta x = \Delta y = 0.25$ and a total of 3136 elements is employed for this study. The temporal error is obtained by computing the RMS difference of all conserved variables at all grid points between the computed solution and the reference exact solution.

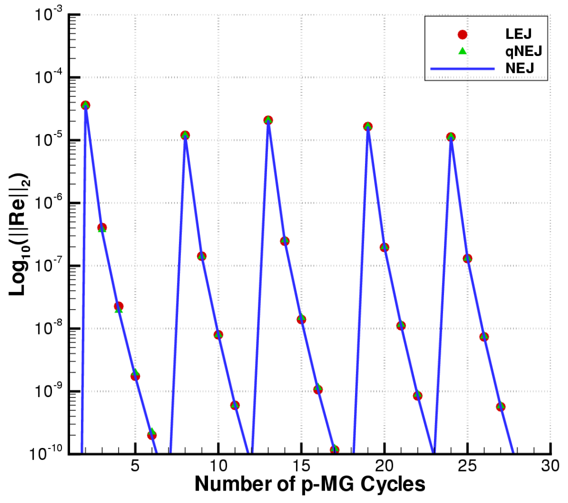
The temporal accuracy results for the BDF1, BDF2, CN2 and IRK4 schemes at $t = 4$ are illustrated in Fig. 5.5, where the computed temporal error is plotted as a function of the time-step on a log-log plot. The first-order backwards differencing scheme displays a slope of 1.0. The second-order backwards differencing scheme and the Crank-Nicholson scheme demonstrate similar slopes of 1.9 and 2.0, respectively, although the CN2 scheme is consistently more accurate in absolute terms than the BDF2 scheme. The fourth-order Runge-Kutta scheme exhibits a slope of 3.82, which is close to the design value of 4. For any given time-step size, the IRK4 scheme achieves higher accuracy than the BDF1, BDF2 and CN2 schemes, while the BDF2 and CN2 schemes provide better accuracy than BDF1. For example, using a time-step size of $\Delta t = 0.25$, the IRK4 scheme attains a temporal error of approximately 10^{-6} while all the other schemes incur temporal errors larger than 3×10^{-4}.

These results demonstrate that the chosen temporal discretization schemes achieve their design order of accuracy within the range of time steps of interest and for high-order spatial discretizations. On the other hand, since the higher-order temporal schemes (particularly the IRK scheme) are more expensive than the lower order schemes, the more practical consideration of computational efficiency of these schemes for a prescribed error tolerance must be addressed. However, since the efficiency of these schemes is inherently related to the performance of the implicit solver used at each time step, the performance of the p-Multigrid solver must first be examined and quantified.

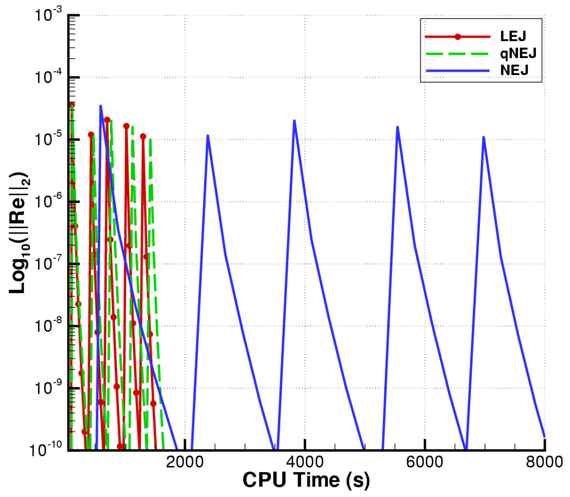
Comparison of *p*-Multigrid Smoothing Strategies

As discussed in Section 5.3, several p-Multigrid solver variants consisting of the non-linear element Jacobi (NEJ), the quasi-nonlinear element Jacobi (qNEJ) and the linearized element Jacobi (LEJ) can be used as smoothers at each implicit time step. In this section, the performance of these three variants in one implicit time step is demonstrated using the IRK4 scheme and the $p = 4$ spatial discretization for the vortex convection test case. Within the multigrid scheme, 5 smoothing passes are utilized at each mesh level. Furthermore, the qNEJ and LEJ smoothers effectively freeze the non-linear Jacobians (and residuals for the LEJ scheme) over 5 iterations, resulting in a single non-linear update on each grid level within a multigrid cycle.

Fig. 5.6(a) depicts the convergence profiles of the LEJ, qNEJ and NEJ solvers in terms of p-Multigrid cycles for a single implicit time step solution. Since the solution of five unsteady residuals (i.e. $\mathbf{R}_e(\tilde{\mathbf{u}}^{(1)}, \cdots, \tilde{\mathbf{u}}^{(s)})$, $s = 2, \cdots, 6$) is required to be solved in a single implicit time step of the IRK4 scheme, the results depicted in this figure show five stage-convergence histories. As expected, the results are similar to those obtained in the context of the equivalent steady-state solver [16, 48]: the linearized element Jacobi, quasi-nonlinear element Jacobi and non-linear element



(a) Convergence history in terms of p-Multigrid cycles



(b) Convergence history in terms of CPU time

Figure 5.6: Comparisons of convergence for solver variants (LEJ, qNEJ and NEJ) in one time step of the IRK4 scheme on a mesh of 7056 elements and discretization order $p = 4$, using fixed time-step size of $\Delta t = 0.5$.

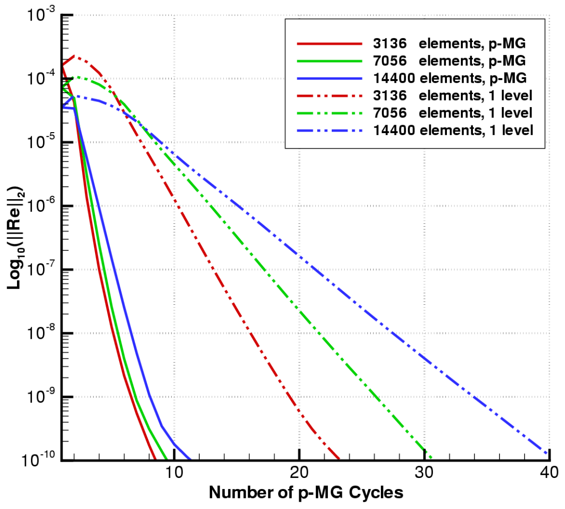
Jacobi converge with similar rates in terms of the number of p-Multigrid cycles. However, when compared in terms of CPU time, as shown in Fig. 5.6(b), the LEJ and qNEJ solvers are seen to be substantially more efficient than the NEJ solver due to the omission of the expensive block diagonal Jacobian re-computations required in the latter scheme at each smoothing pass. In this test case, the LEJ and qNEJ smoothers require only a single Jacobian evaluation and LU factorization at each mesh level within a multigrid cycle, whereas this procedure must be repeated 5 times for the NEJ scheme. The reduction of the number of non-linear residual evaluations in the LEJ scheme results in only slightly improved CPU time, moreover, since the off-diagonal blocks of the Jacobian matrix are not stored in the qNEJ scheme, the qNEJ scheme may be an appropriate compromise for cases where memory limitations are dominant.

Efficiency of p-Multigrid Approach

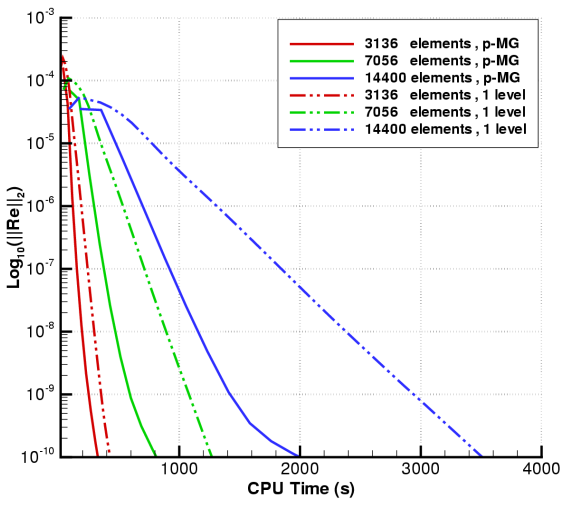
Having settled on the use of the qNEJ smoother, we must now assess the efficiency of the overall p-Multigrid solver for time-implicit problems, in order to later examine and understand the efficiency comparisons of the various time-integration schemes. The use of efficient non-linear solvers is particularly important for the performance of high-order temporal schemes, such as IRK, which entail the solution of multiple non-linear problems within a given time step. These schemes are substantially more expensive per time step than their lower-order counterparts, but enable the use of much larger time steps for equivalent accuracy, due to their superior asymptotic properties. Thus, the performance of the multigrid scheme as a function of time step size must be considered in addition to the more traditional performance metric as a function of grid size.

In this section, we examine the convergence of the non-linear p-Multigrid solver (using the qNEJ smoother) compared to that of the corresponding single level solver as a function of grid size and time-step size in one implicit time-integration. Specifically, the grid sizes vary from 3136, to 7056 to 14400 elements on a domain of $-7 \leq x \leq 7$ and $-3.5 \leq y \leq 3.5$, and time-step sizes range from $\Delta t = 0.5$, $\Delta t = 1.0$ to $\Delta t = 5.0$. The spatial discretization ($p = 4$), as well as initial and boundary conditions, are kept unchanged and we choose the BDF2 scheme as the time-integration scheme, with the understanding that the performance of the multigrid scheme will be similar for implicit systems arising from each BDF or CN2 time step, or each IRK stage.

Fig. 5.7(a) depicts the p-Multigrid convergence for various mesh sizes, compared with the single level solver for a fixed time step of the BDF2 scheme. The solid lines represent computed results using the p-Multigrid strategy and the dashed lines represent results using the corresponding single level solver, i.e. no multigrid involved. Since the time-step size has been fixed for all runs, this figure illustrates the effect of grid resolution or mesh size on convergence rate. An important aspect of these results is the relative insensitivity of the p-Multigrid convergence to the mesh size, which illustrates the h-independent property of the p-Multigrid solver. On the other hand, the convergence rate of the single level solver decreases progressively for finer mesh sizes. While a p-Multigrid cycle requires more computational time than a single-grid cycle, the p-Multigrid approach is still seen to be more efficient overall in terms of CPU time than the single level solver, as depicted in Fig. 5.7(b), and the benefit



(a) Convergence history vs. p-Multigrid cycles

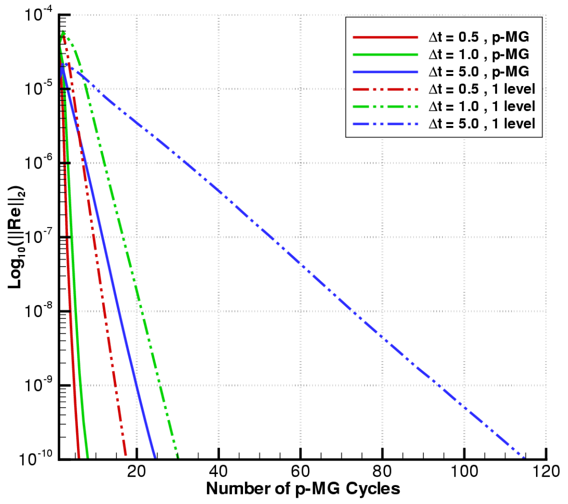


(b) Convergence history vs. CPU time

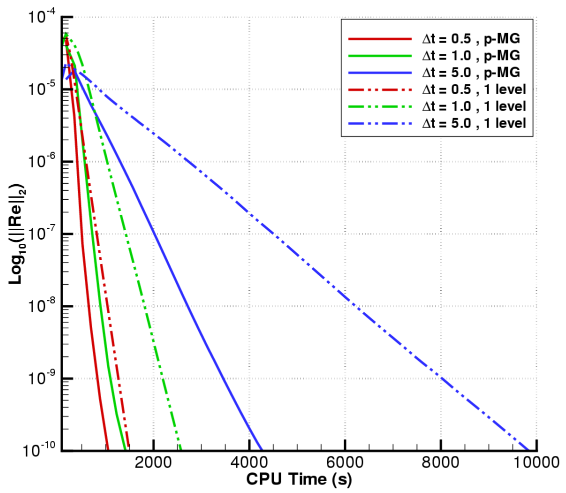
Figure 5.7: Comparisons of the convergence between p-Multigrid and single level solvers, using various grid sizes, N=3136, N=7056 and N=14400 with a fixed time-step of Δt = 1.0 for the BDF2 scheme.

of the p-Multigrid solver can be expected to increase for larger mesh sizes.

Since time-step size is another factor which affects convergence and computational time, we also discuss the characteristics of the p-Multigrid approach and corresponding single level solver with respect to time-step sizes. The obtained convergence profiles for various time-step sizes are illustrated in Fig. 5.8, for the largest mesh size of $N = 14400$. As previously, the p-Multigrid approach delivers faster convergence and requires lower overall CPU time than the corresponding single-level solver, for a given time step. As the time-step size is increased, the convergence of both solvers deteriorates, as expected, since small time-step sizes correspond to more diagonally dominant systems, which can be inferred from the diagonal block formulations in Table 5.3. In particular, at relatively small time-step values, the use of the p-Multigrid solver may not be necessary, and a few passes of the single grid solver (element Jacobi) may be sufficient to adequately solve the implicit system. However, for large time steps, the convergence of the p-Multigrid solver deteriorates less severely than that of the single level solver, and will eventually asymptote to the convergence observed for steady-state problems [16]. Thus, for moderate time-step sizes, the p-Multigrid



(a) Convergence history vs. p-MG cycles



(b) Convergence history vs. CPU time

Figure 5.8: Comparisons of the convergence between p-Multigrid and single level solvers, using various time-step sizes, $\Delta t = 0.5, \Delta t = 1.0$ and $\Delta t = 5.0$, with a fixed mesh size of N=14400 for the BDF2 scheme.

solver provides the most efficient mechanism for integrating the implicit equations in time, and the benefit of this method can be expected to increase for larger time-step sizes and for finer meshes. For example, using a time step of $\Delta t = 1$ on the mesh size of N=14400, the p-Multigrid approach achieves a speedup of 1.8 over the single grid solver, while a speedup of 2.35 is obtained for a time step of $\Delta t = 5$.

Comparison of Efficiency of Temporal Schemes

The results of the time-step refinement study discussed in the temporal accuracy section and depicted in Fig. 5.5 are revisited in Fig. 5.9, where the temporal error is plotted as a function of the required CPU time, which is obtained using the p-Multigrid algorithm (with qNEJ smoother) described in Section 5.3, the performance of which at one implicit time step was examined in the previous section. In all cases, the non-linear residuals at each time step are converged to machine precision. Although this may not be necessary for preserving the final accuracy of the computed solutions, the determination of suitable levels for implicit system convergence remains an active research area and is explored further in Chapter 6. In any case, for a given accuracy level, one may expect the required convergence levels to be the same for all

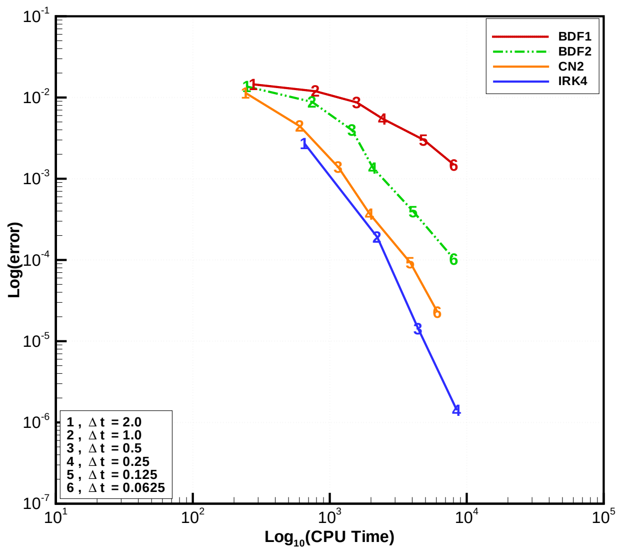
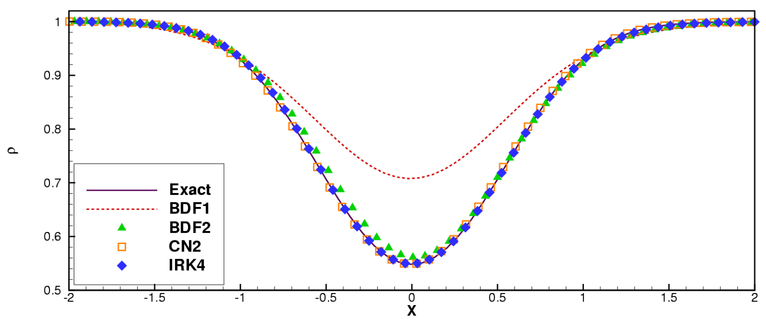
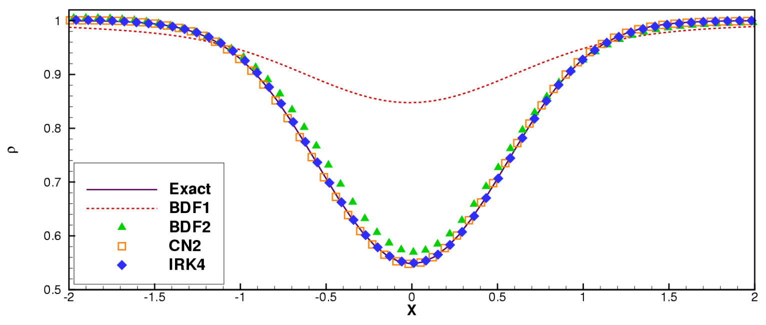


Figure 5.9: Comparisons of temporal efficiency for various implicit temporal schemes at t = 4 for vortex convection test case.

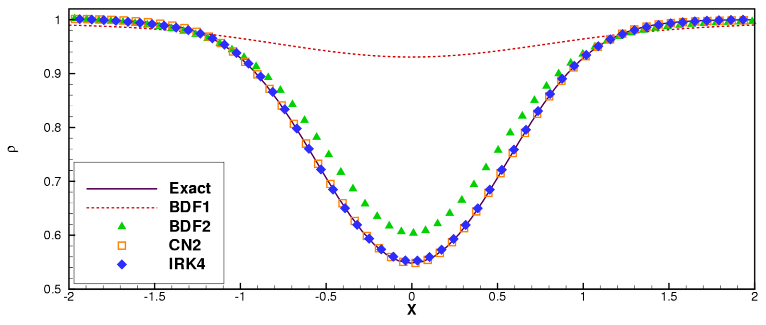
schemes, so that the relative comparisons between CPU time required for the various temporal schemes remain valid. The results of Fig. 5.9 indicate that in order to reach a specified error level, the IRK4 scheme requires the least CPU time, while the BDF1 scheme is clearly not practical in terms of efficiency, due to its low convergence rate. For example, to achieve an accuracy level of 1×10^{-4}, the IRK4 scheme requires 2615 seconds of CPU time, while the BDF2 scheme requires 8042 seconds, which is over 3 times longer. On the other hand, the CN2 scheme in this case requires 3975 seconds to achieve this accuracy level, which only is 1.52 times longer than the IRK4 scheme and twice as fast as the BDF2 scheme. While the CN2 scheme has the same asymptotic behavior as the BDF2 scheme, it consistently outperforms the BDF2 scheme and comes closest to matching the efficiency of the IRK4 scheme. Nevertheless, the lack of L-stability for the CN2 scheme remains a disadvantage which may result in poor performance of this scheme in other cases. For higher accuracy levels, the advantage of the IRK4 scheme increases, due to the asymptotic properties of these schemes. In general, the choice of the most efficient scheme will depend not only on the efficiency of the non-linear solver, but also on the desired level of accuracy, with higher accuracy levels favoring higher-order schemes.



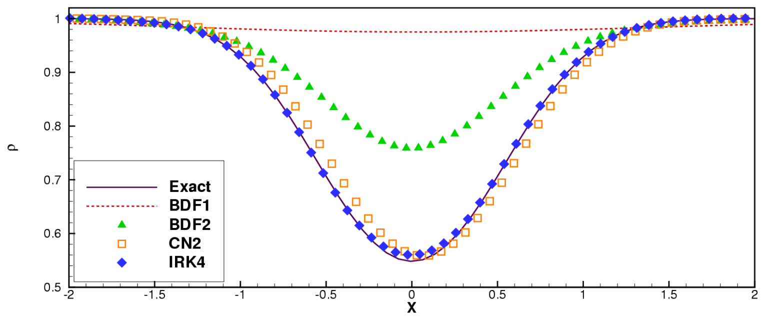
(a) t∼4



(b) t∼10



(c) t∼20



(d) t∼50

Figure 5.10: Comparison of numerical results with the exact solution, using the various temporal schemes for equal work performance, illustrated by density profiles at the centerline $y = 0$, at $t \sim 4$, 10, 20 and 50. The various required time-step sizes are indicated in Table 5.4.

Table 5.4: Time-step sizes and CPU costs for BDF1, BDF2, CN2 and IRK4 schemes used for accuracy study at equivalent CPU cost for vortex convection test case.

	BDF1	BDF2	CN2	IRK4
Δt	0.31	0.24	0.2	1.1
$t \simeq 4$	2080	1965	2180	1939
$t \simeq 10$	5056	5013	4951	4895
$t \simeq 20$	9810	10122	9895	9818
$t \simeq 50$	23908	25215	24736	24575

In another comparison of the efficiency of the various schemes, the delivered accuracy of these schemes at equivalent CPU cost is compared in Fig. 5.10. This numerical experiment is similar in nature to that displayed in Fig. 5.4, except that the comparison of the solutions is made at equivalent CPU time as opposed to the use of a constant time step. In order to achieve equivalent CPU time for the various schemes, the time step of each scheme must be adjusted accordingly to achieve a target total run time. The chosen time step and resulting computational cost for each scheme are given in Table 5.4, where the use of larger time steps for the more expensive schemes is evident. While the chosen time steps result in approximately similar total CPU costs, the final solutions do not always lie at exactly the same locations in time, due to the use of integer numbers of time steps of varying sizes. Therefore, the final solutions are compared by translating the vortex density profile back to the origin, in order to compare solution accuracy at the approximate points in time given by $t \sim 4$, $t \sim 10$, $t \sim 20$ and $t \sim 50$. The results illustrate what can be surmised from Fig. 5.9, namely that the fourth-order implicit Runge-Kutta scheme remains the most competitive approach, while the errors in the other schemes accumulate more rapidly as time proceeds, with the CN2 scheme providing the next best accuracy for a given cost. Note that the IRK scheme achieves superior accuracy in this case while using a time step which is approximately 5 times larger than that used for the lower-order schemes.

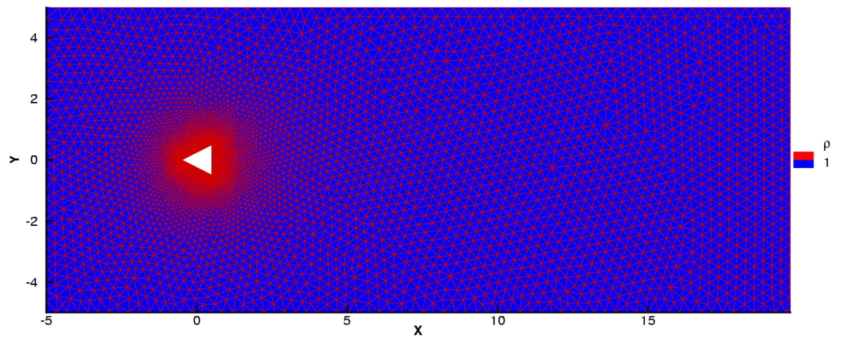
5.4.2 Shedding Flow over a Triangular Wedge

The next test case involves the problem of vortex shedding over a triangular wedge. Because of the inviscid nature of the flow simulation, a triangular geometry is chosen in order to ensure that vortices are produced due to separation at sharp corners. In addition to representing a more realistic test case and to involving a highly graded unstructured mesh, this case is also used to study the performance of the various temporal schemes in combination with low and higher-order spatial discretizations, thus focusing on the interplay between spatial and temporal errors.

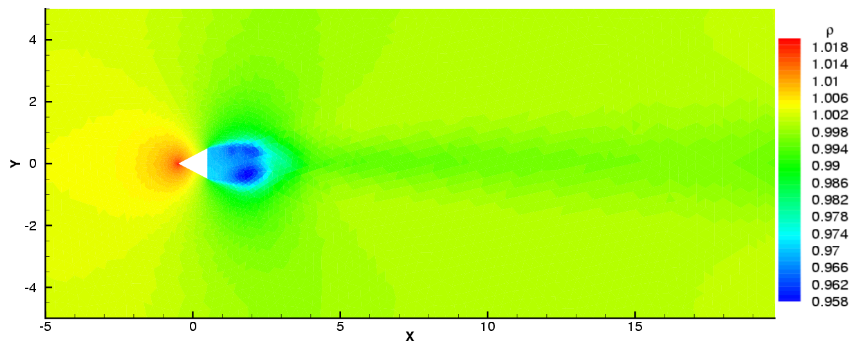
Numerical Solutions

The geometry consists of a triangular wedge placed on the centerline $y = 0$ of the computational domain, which contains 10,836 unstructured triangular elements, with the ratio of the smallest to largest cell area being 1:1425 (which corresponds to an explicit CFL ratio of 38:1). The flow is inviscid, and a uniform freestream Mach number of 0.2 is applied as the initial condition. Since our principal focus in these calculations is the ability of these schemes in retaining the shape of the vortices as these are convected downstream from the wedge, and in order to reduce the occurrence of initial discontinuities near the surface of the wedge which would be produced with a uniform freestream initial condition and would be detrimental for high-order spatial discretizations, we first employ a $p = 0$ spatial discretization and BDF2 temporal scheme, utilizing a uniform-flow initial condition: $u(\mathbf{x}, t = 0) = u_\infty = 0.5$, $v(\mathbf{x}, t = 0) = v_\infty = 0$, $\rho(\mathbf{x}, t = 0) = \rho_\infty = 1.0$ (shown in Fig. 5.11(a)), to obtain an intermediate solution (shown in Fig. 5.11(b)), in which the formed vortices have not yet separated. Then, this intermediate solution is applied as the initial condition to all other high-order $p = 1$ and $p = 3$ spatial-discretization schemes, using either the BDF2 or IRK4 time-integration schemes.

Starting with the computed intermediate solution as the initial condition, Figures 5.12 and 5.13 depict the numerical results at $t = 100$ for $p = 1$ and $p = 3$ spatial discretizations using the BDF2 and IRK4 time-integration schemes with the same



(a) mesh and uniform flow



(b) intermediate solution

Figure 5.11: Mesh and density contours for uniform flow and intermediate $p = 0$ solution used as the initial condition for high-order schemes in the shedding vortex test case.

fixed time-step size of $\Delta t = 0.05$, respectively. It can be observed that for a fixed temporal scheme (either the BDF2 or IRK4 scheme), the higher-order accurate $p = 3$ spatial scheme provides the best shape-retaining convection capability: the vortices produced around the corners of the triangle wedge keep their shapes far downstream. On the other hand, the $p = 1$ scheme is relatively dissipative as seen by the diffusion of the core of the vortices as they are convected downstream. While the improvement in the solution due to the increase in spatial discretization order is evident, any solution changes due to the use of higher-order temporal schemes are less evident in these qualitative illustrations.

Returning to the results of Fig. 5.12 and 5.13, we take a more quantitative look at the incurred errors, by plotting the the density distributions along cuts taken through the centerlines of last three downstream vortices in Fig. 5.14. For the lower-order spatial discretization of $p = 1$ (Fig. 5.14(a)), there is no evident improvement in overall solution accuracy using the IRK4 temporal discretization scheme over the BDF2 scheme, since the spatial error dominates the temporal error in this case. On the other hand, for the higher-order spatial discretization of $p = 3$ (Fig. 5.14(b)), the IRK4 scheme is seen to be more accurate than the BDF2 scheme, producing notable variations in both diffusion and dispersion rates. Therefore, one may conclude that the use of higher temporal accuracy is not always beneficial, and can even be wasteful in cases where the dominant error is due to other sources. Generally, the most efficient overall solution procedures for a given error tolerance will be strategies which attempt to balance temporal and spatial error sources. This may not be possible in the case of explicit schemes, where the maximum permissible time step is determined by stability considerations rather than accuracy considerations.

Temporal Accuracy

In order to assess the accuracy of our temporal discretizations, we perform a temporal refinement study, similar to that described for the case of the periodic convecting vortex. The temporal error is isolated by producing a reference"exact" numerical solution, using a small time-step size of $\Delta t = 2 \times 10^{-4}$ for each temporal and spatial discretization. Using the same initial and boundary conditions as described above, computations are performed using time-step sizes from 0.05 to 5×10^{-4} and the error is measured at $t = 0.1$. The results shown in Fig. 5.15 indicate that the design accuracy of the BDF2 and IRK4 schemes is approached in both cases, yielding curve slopes of 1.92 and 3.96 in the case of the second-order accurate ($p = 1$) spatial discretization, and 1.72 and 3.96 in the case of the fourth-order accurate ($p = 3$) spatial discretization, for the BDF2 and IRK4 temporal schemes, respectively. However, it is notable that the absolute temporal errors of both implicit time-integration schemes are consistently smaller for the second-order accurate ($p = 1$) spatial discretization as compared to the higher-order spatial discretization case ($p = 3$).

Implicit versus Explicit schemes

In this section, a qualitative comparison is performed between a second-order accurate explicit scheme and the second-order BDF2 implicit scheme for the vortex shedding problem described above using the fourth-order accurate ($p = 3$) spatial discretization. The formulation of the second-order explicit forward Euler scheme (FD2) is written as:

$$\text{FD2} : \mathbf{R}_e(\tilde{\mathbf{u}}^{n+1}) = \frac{M}{\Delta t}(\frac{3}{2}\tilde{\mathbf{u}}^{n+1} - 2\tilde{\mathbf{u}}^{n} + \frac{1}{2}\tilde{\mathbf{u}}^{n-1}) + \mathbf{R}(\tilde{\mathbf{u}}^{n}) = 0 \tag{5.20}$$

By arranging this formulation, the explicit solution at time step $n+1$ can be computed as:

$$\tilde{\mathbf{u}}^{n+1} = \frac{4}{3}\tilde{\mathbf{u}}^{n} - \frac{1}{3}\tilde{\mathbf{u}}^{n-1} - \frac{2}{3}M^{-1}\Delta t\mathbf{R}(\tilde{\mathbf{u}}^{n}) \tag{5.21}$$

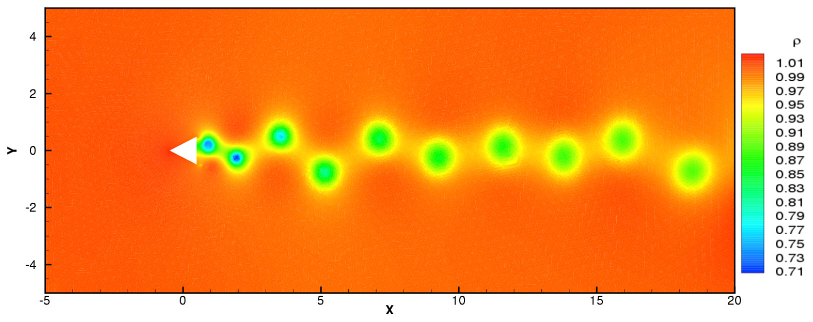
It can be observed that no Jacobian matrix is required to be computed or stored in the explicit scheme and the major computation work is to evaluate the spatial residual vector, $\mathbf{R}(\tilde{\mathbf{u}}^{n})$, as opposed to the solution of a large non-linear system of equations in the implicit schemes. Therefore the required computational cost at each time step can be drastically decreased compared to implicit schemes. However, explicit schemes are restricted to small time-step sizes due to stability considerations. In the current text case, the maximum permissible time step for stability, which is determined by the smallest cells in the mesh, is $\Delta t_{max} = 5 \times 10^{-5}$ for the FD2 scheme. This is three orders of magnitude smaller than the time-step of 0.05 used for the BDF2 scheme.

Table 5.5 compares the characteristics of the explicit and implicit approaches to this problem, using the p-Multigrid solver to drive the implicit scheme, and converging each implicit system to a relatively low tolerance of 1×10^{-7}. Since the relatively large time-step can be used in the BDF2 scheme, the total number of time steps to integrate the solution to the non-dimensional time $t = 2.5$ is 1000 times smaller than that of the explicit scheme. Thus, although the CPU time required for a single explicit time step is much lower than that required for an implicit time step, the results in Table 5.5 show that the the implicit time-integration scheme requires only 22% of the CPU

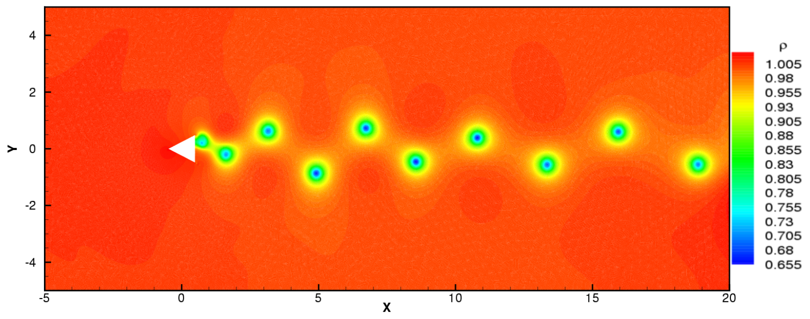
Table 5.5: Explicit-implicit comparison for the shedding flow case

for solution at $t = 2.5$	time-step size	time steps	convergence limit	CPU time (s)
implicit (BDF2)	$\Delta t = 0.05$	50	1×10^{-7}	5160
explicit (FD2)	$\Delta t = 5 \times 10^{-5}$	50000	–	22920

time of the explicit approach in this case. However, this comparison is only qualitative in nature, since the temporal accuracy of the two approaches is not matched, and the CPU time required by the implicit approach is strongly dependent on the level of convergence required of the inner iterations for the implicit solver. However the performance of the explicit scheme is determined by stability considerations, which will be detrimental when the resulting temporal accuracy imposed by the stability limit is not closely related to the desired temporal accuracy.

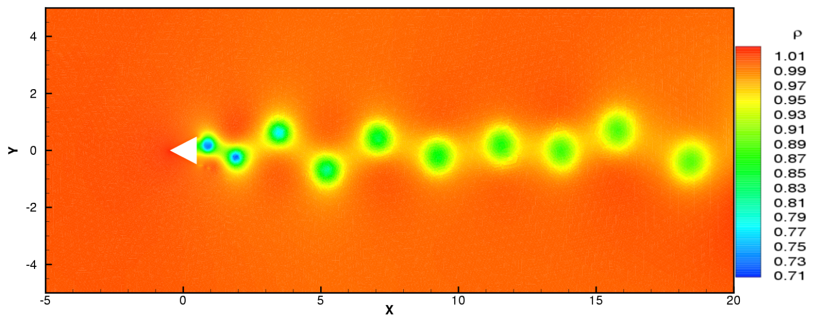


(a) $p = 1$

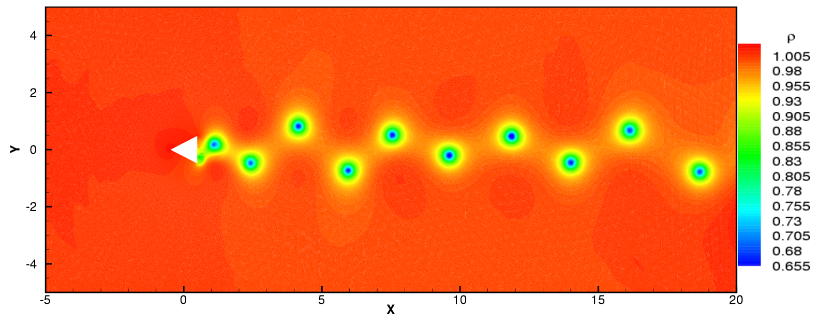


(b) $p = 3$

Figure 5.12: Density contours of the $p = 1$ and $p = 3$ spatial-discretization schemes, at $t = 100$, using the BDF2 scheme and $\Delta t = 0.05$.

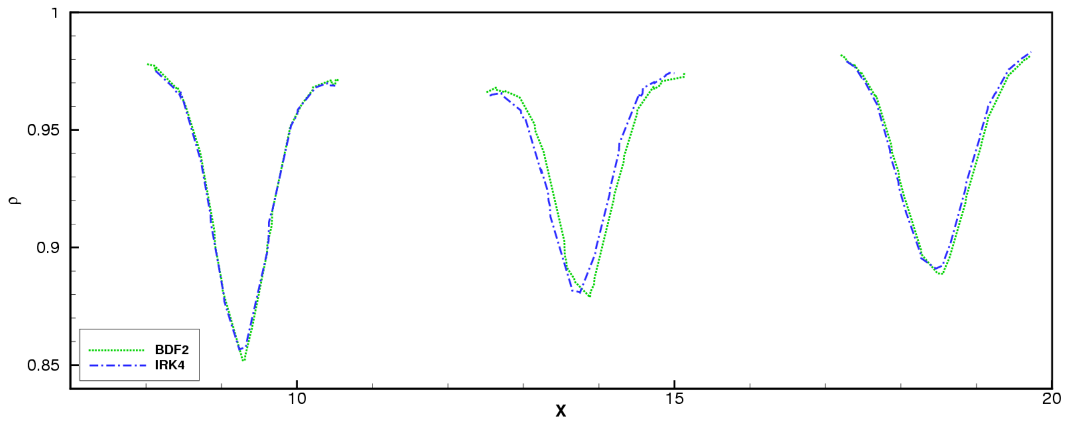


(a) $p = 1$

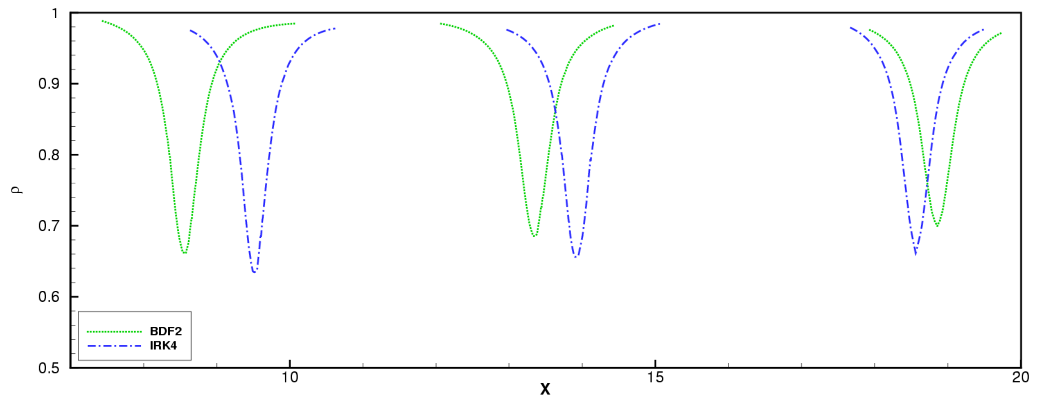


(b) $p = 3$

Figure 5.13: Density contours of the $p = 1$ and $p = 3$ spatial-discretization schemes, at $t = 100$, using the IRK4 scheme and $\Delta t = 0.05$.

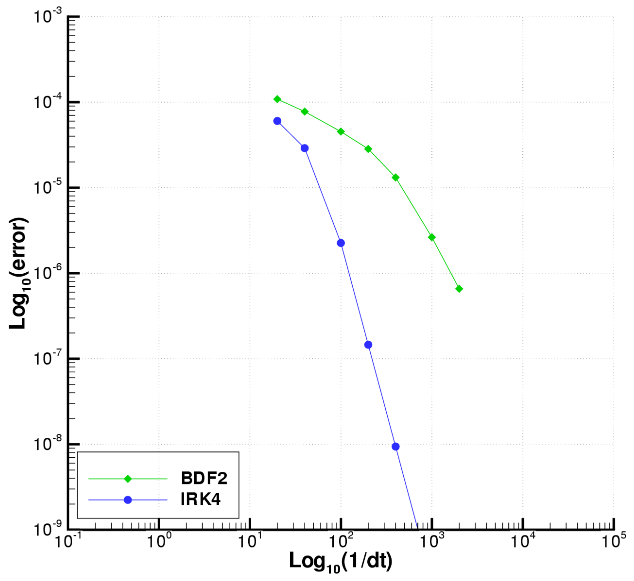


(a) $p = 1$

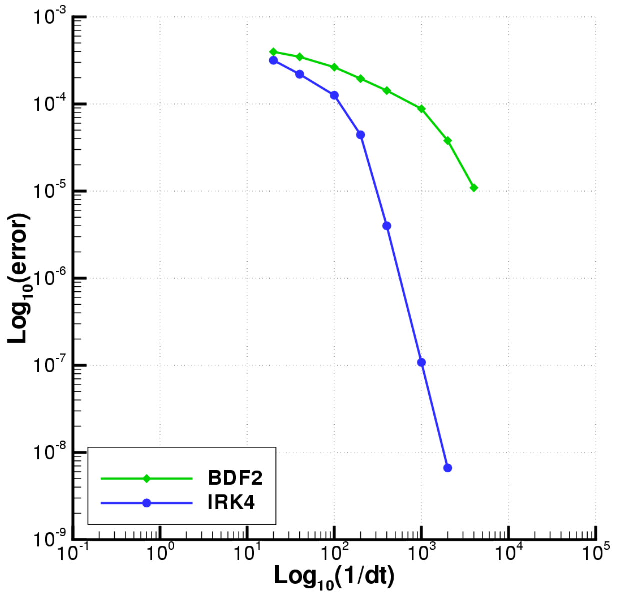


(b) $p = 3$

Figure 5.14: Density distributions of the $p = 1$ and $p = 3$ spatial-discretization schemes, at $t = 100$, using the BDF2 and IRK4 schemes and $\Delta t = 0.05$. Note the different scales used for the $p = 1$ and $p = 3$ spatial-discretizations.



(a) $p = 1$



(b) $p = 3$

Figure 5.15: Comparison of temporal accuracy as a function of time-step size for the BDF2 and IRK4 schemes at $t = 0.1$.

Chapter 6

Adjoint-based Temporal Error Estimation and Adaptation

In Chapter 4, an adjoint-based error estimation technique was developed to determine global functional error in spatial discretizations and to guide adaptive mesh refinement schemes for spatial error reduction. The main purpose of this chapter is to extend the same methodology to unsteady flow problems, namely providing a general approach for predicting temporal error relevant to a specified time-dependent objective functional, identifying temporal error distributions for discretizations in the time domain, and applying subsequently an adaptive time-step refinement procedure for temporal error reduction. In the current work, the error sources under primary consideration include the error resulting from a time discretization, which is determined by the time-step size, Δt, as well as the algebraic error resulting from incomplete convergence of the non-linear implicit system being solved at each implicit time step. The formulation derived for accomplishing these tasks reveals that a time-dependent flow problem together with an unsteady adjoint problem must be solved at each adaptation cycle. Therefore, in order to make the entire strategy perform competitively, the efficient solution strategies described in the preceding sections are used for accelerating convergence of both unsteady flow and adjoint problems.

6.1 Objective

Typically, uniform time-step sizes are utilized for unsteady flow simulations, such as the test cases described in Section 5.4. However in general, due to the non-linear nature of the governing equations, it is not simple to choose *a priori* a realistic time-step size to ensure specific simulation error tolerances. A brute-force approach to selecting suitable time-step sizes consists of repeating the simulation multiple times with progressively smaller time steps until the desired temporal accuracy is attained. However, even if a suitable time-step size can be selected, the use of a uniformly small time step can be wasteful due to excessive temporal resolution in regions where high temporal resolution is not necessary for the accuracy of outputs of interest. In this context, a global error control technique via the discrete adjoint method addressed in this chapter can be used to estimate temporal error distribution for a specified objective functional, and thus to guide temporal resolution enhancement specifically in regions which contribute strongly to the global temporal error. In addition to the temporal resolution error, other important sources of error may be present. Specifically in the present work, the error arising from incomplete convergence of the non-linear implicit system at each time step is also considered, and techniques for ensuring this "algebraic" error is significantly smaller than the temporal resolution error are developed.

The adjoint-based temporal error estimation can be derived in a similar manner as described in Chapter 4 for spatial error estimates. The resulting adjoint sensitivity formulation associated with an objective functional of interest leads to the requirement of solving an unsteady adjoint problem which must be solved backwards in time [67]. Specifically, the unsteady flow solution arising from the governing equations is solved from an initial time value to a final time value, while the unsteady adjoint solution is solved in the same discretized time domain, but in a reverse order (i.e. starting from the final time value backwards to the initial time point). This implies that the entire flow solution history must be stored since it is reused to evaluate the subsequent adjoint solution. Thus, the computer program consists of three main components: the first component performs the forward time-integration prescribed in the previous chapter for the various implicit temporal schemes in conjunction with high-order spatial discontinuous Galerkin discretizations. Since it is not practical to save the entire unsteady solution history in memory, the flow solution at each implicit time step is written as a file to the hard disk. The second component involves solving the unsteady adjoint problem at each equivalent time step and writing the adjoint solution history to the hard disk at each time step during the backwards time sweep. The third component reads all required data via input file operations, computes global as well as local estimated errors and performs the local time-step refinement strategy for the current time domain.

This chapter is organized as follows. In Section 6.2, the unsteady discrete adjoint-based error-estimation procedure is formulated for temporal error sources using various orders of temporal discretizations, namely the first-order backward difference (BDF1) scheme and the fourth-order implicit multistage Runge-Kutta (IRK4) scheme. In Section 6.3, this technique is extended to account for algebraic error sources due to incomplete convergence of the governing equations. Section 6.4 describes a temporal adaptive refinement criteria and Section 6.5 shows several time-dependent numerical results in order to demonstrate the temporal error estimation technique, providing comparisons in terms of performance with a uniform time-step refinement approach.

6.2 Adjoint-based Temporal Error Estimation

This section considers discrete adjoint formulations for the first-order BDF1 scheme and the fourth-order IRK4 scheme. Formulations for other temporal discretization schemes can be derived by applying the same strategy as proposed in this section. The model problem involves the unsteady two-dimensional compressible Euler equations as described in the previous chapter. The goal of this procedure is to obtain a distribution of the functional error in the time domain which can be used either to correct the current functional value or to drive an adaptive time-step refinement method for improving functional accuracy.

6.2.1 Time-dependent Functional Output

For time-dependent simulations of engineering interest, the objective may consist of an output functional computed at the final time, or a constructed quantity related to time-integrated flow-field variables. In general, a time-integrated objective can be expressed as:

$$L^f = \int_0^T L(t)dt \tag{6.1}$$

where $L(t)$ denotes a time-dependent functional such as drag or lift at time t. When the above time integral is discretized, the objective can be expressed as:

$$L^f(\tilde{\mathbf{u}}^1, \tilde{\mathbf{u}}^2, \cdots, \tilde{\mathbf{u}}^n) = \sum_{i=1}^{n} q^i(\Delta t^i) L^i(\tilde{\mathbf{u}}^i) \tag{6.2}$$

where the notation $(\cdot)^i$ refers to a quantity evaluated at time step i, $\tilde{\mathbf{u}}^i$ denotes the time-dependent flow variables obtained at time step i, and $L^i(\tilde{\mathbf{u}}^i)$ represents the time-dependent objective component evaluated using $\tilde{\mathbf{u}}^i$. q^i represents the quadrature weight associated with time step i. The sensitivity vector of the time-dependent objective functional with respect to the set of unsteady flow variables can be then expressed as:

$$\frac{\partial L^f}{\partial \tilde{\mathbf{u}}} = \left(\frac{\partial L^f}{\partial \tilde{\mathbf{u}}^1}, \frac{\partial L^f}{\partial \tilde{\mathbf{u}}^2}, \cdots, \frac{\partial L^f}{\partial \tilde{\mathbf{u}}^n} \right) = \left(q^1 \frac{\partial L^1}{\partial \tilde{\mathbf{u}}^1}, q^2 \frac{\partial L^2}{\partial \tilde{\mathbf{u}}^2}, \cdots, q^n \frac{\partial L^n}{\partial \tilde{\mathbf{u}}^n} \right) \tag{6.3}$$

If the objective functional of interest is only dependent on the flow quantities at the final time, the only non-zero component of the flow sensitivity vector is the last entry.

6.2.2 First-order Backwards Euler Scheme

Formulation

Consider an affordable coarse mesh in the time domain, represented by H using relatively large time-step sizes. $\tilde{\mathbf{u}}_H$ denotes the vector of time-dependent flow variables obtained using the BDF1 scheme (c.f. Eq. (5.6)) at the current temporal resolution level, i.e. $\tilde{\mathbf{u}}_H = \{\tilde{\mathbf{u}}_H^i, i = 1, 2, \cdots, n\}$, where the superscript i refers to the time step at the current temporal resolution level. By solving the time-dependent flow problem at the current temporal resolution level, the coarse time level functional, $L_H^f(\tilde{\mathbf{u}}_H)$, can be evaluated. However, $L_H^f(\tilde{\mathbf{u}}_H)$ may not be sufficiently accurate to satisfy a particular design requirement. We seek an approach to approximate the objective functional evaluated at a uniformly refined temporal resolution level (i.e. fine time level), constructed by subdividing each coarse level time step by a factor of 2 (i.e. $\Delta t_h = \Delta t_H/2$). However, it may not be practical to evaluate the unsteady flow variables and the subsequent objective output (denoted as $\tilde{\mathbf{u}}_h$ and $L_h^f(\tilde{\mathbf{u}}_h)$ respectively) directly on the fine level temporal mesh since this requires roughly twice the cost of the original time-dependent computation. Instead, we first project the current unsteady flow solution, $\tilde{\mathbf{u}}_H$, onto the fine temporal resolution level via linear interpolation to obtain the projected coarse unsteady flow solution, $\tilde{\mathbf{u}}_H^h$, i.e. $\tilde{\mathbf{u}}_H^h = I_H^h \tilde{\mathbf{u}}_H$. Then, the objective functional on the fine temporal resolution level is expanded using a Taylor series expansion with respect to the projected coarse unsteady flow variable set, expressed as:

$$L_h^f(\tilde{\mathbf{u}}_h) = L_h^f(\tilde{\mathbf{u}}_H^h) \; + \; \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h}\right)_{\tilde{\mathbf{u}}_H^h} (\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h) + \cdots \qquad (6.4)$$

where $\left(\partial L_h^f / \partial \tilde{\mathbf{u}}_h\right)_{\tilde{\mathbf{u}}_H^h}$ denotes the sensitivity vector of the fine time level objective output with respect to the fine time level flow variable set, evaluated at the projected coarse unsteady flow variable state, $\tilde{\mathbf{u}}_H^h$. The discrete solution error, $\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h$, in the above equation is not available due to the presence of the unknown fine time level flow solution, $\tilde{\mathbf{u}}_h$. Therefore, we make use of the fine level unsteady flow residual vector, $\mathbf{R}_{eh}(\tilde{\mathbf{u}}_h) = \{\mathbf{R}_{eh}^{\,i}\}$, which is expanded in a similar way with respect to the projected coarse unsteady flow variable set as:

$$\mathbf{R}_{eh}(\tilde{\mathbf{u}}_h) = \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h) \; + \; \left[\frac{\partial \mathbf{R}_{eh}}{\partial \tilde{\mathbf{u}}_h}\right]_{\tilde{\mathbf{u}}_H^h} (\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h) + \cdots \; = 0 \qquad (6.5)$$

where $[\partial \mathbf{R}_{eh} / \partial \tilde{\mathbf{u}}_h]_{\tilde{\mathbf{u}}_H^h}$ denotes the full unsteady Jacobian matrix formulated at the fine time level while evaluated at the projected coarse level flow solution state. By re-arranging Eq. (6.5) and using the relation $\mathbf{R}_{eh}(\tilde{\mathbf{u}}_h) = 0$ (since $\tilde{\mathbf{u}}_h$ represents the exact unsteady solution on the fine time level mesh) an approximation for the discrete solution error, $\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h$, can be obtained:

$$\tilde{\mathbf{u}}_h - \tilde{\mathbf{u}}_H^h = -\left[\frac{\partial \mathbf{R}_{eh}}{\partial \tilde{\mathbf{u}}_h}\right]^{-1}_{\tilde{\mathbf{u}}_H^h} \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h) \tag{6.6}$$

Substituting Eq. (6.6) into Eq. (6.4), the objective functional on the fine temporal resolution level is approximated as:

$$L_h^f(\tilde{\mathbf{u}}_h) \approx L_h^f(\tilde{\mathbf{u}}_H^h) \; - \; \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h}\right)_{\tilde{\mathbf{u}}_H^h} \left[\frac{\partial \mathbf{R}_{eh}}{\partial \tilde{\mathbf{u}}_h}\right]^{-1}_{\tilde{\mathbf{u}}_H^h} \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h) \tag{6.7}$$

In general, the unsteady Jacobian matrix $[\partial \mathbf{R}_{eh}/\partial \tilde{\mathbf{u}}_h]_{\tilde{\mathbf{u}}_H^h}$ is a large sparse lower block bi-diagonal matrix (illustrated later), and the size of this matrix depends on the total number of time steps. Therefore, it is impractical to solve the inverse of the Jacobian matrix directly. Recalling the approach invoked in the use of the adjoint variables in the steady state problem, a vector of the fine level unsteady adjoint variables, $\boldsymbol{\lambda}_h$, is introduced, satisfying the linear system of equations written as:

$$\boldsymbol{\lambda}_h^T = \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h}\right)_{\tilde{\mathbf{u}}_H^h} \left[\frac{\partial \mathbf{R}_{eh}}{\partial \tilde{\mathbf{u}}_h}\right]^{-1}_{\tilde{\mathbf{u}}_H^h} \quad \text{or} \quad \left[\frac{\partial \mathbf{R}_{eh}}{\partial \tilde{\mathbf{u}}_h}\right]^{T}_{\tilde{\mathbf{u}}_H^h} \boldsymbol{\lambda}_h = \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h}\right)^T_{\tilde{\mathbf{u}}_H^h} \tag{6.8}$$

Comparing Eq. (6.8) with Eq. (4.8), the unsteady adjoint formulation appears to be similar to the corresponding adjoint formulation for the steady-state problem, however, the unsteady adjoint problem in fact spans all discrete time steps in the BDF1 temporal discretization.

Next we describe the solution procedure for the unsteady adjoint problem. Substituting Eq. (6.8) into Eq. (6.7) yields:

$$L_h^f(\tilde{\mathbf{u}}_h) \approx L_h^f(\tilde{\mathbf{u}}_H^h) \; \underbrace{-\boldsymbol{\lambda}_h^T \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_{a*}^t} \tag{6.9}$$

or

$$L_h^f(\tilde{\mathbf{u}}_h) - L_H^f(\tilde{\mathbf{u}}_H) \approx L_h^f(\tilde{\mathbf{u}}_H^h) - L_H^f(\tilde{\mathbf{u}}_H) \underbrace{-\boldsymbol{\lambda}_h^T \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_{a*}^t} \tag{6.10}$$

where ε_{a*}^t corresponds to the error correction term for estimating the functional error arising from two successively refined temporal resolution levels. However, this term is expensive to compute since it requires the solution of the adjoint problem at the fine temporal resolution. Instead, we first solve the discrete unsteady adjoint problem at the coarse temporal resolution as:

$$\left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]_{\tilde{\mathbf{u}}_H}^T \boldsymbol{\lambda}_H = \left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H}\right)_{\tilde{\mathbf{u}}_H}^T \tag{6.11}$$

As discussed previously, the unsteady residual, $\mathbf{R}_{eH} = \{\mathbf{R}_{eH}^i\}$, and the unsteady flow solution, $\tilde{\mathbf{u}}_H = \{\tilde{\mathbf{u}}_H^i\}$, involve the state variables at all discrete locations of the BDF1 temporal discretization. Furthermore, the residual vector component, $\mathbf{R}_{eH}^i$, at the i^{th} time step is only dependent on the state variables, $\tilde{\mathbf{u}}_H^i$ and $\tilde{\mathbf{u}}_H^{i-1}$, in the BDF1 scheme (refer to Eq. (5.6)), therefore, the full unsteady Jacobian matrix has the following lower-block bi-diagonal form:

$$\left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]_{\tilde{\mathbf{u}}_H} = \begin{bmatrix} \left[\frac{\partial \mathbf{R}_{eH}^1}{\partial \tilde{\mathbf{u}}_h^1}\right] & 0 & 0 & \cdots & 0 & 0 \\ \left[\frac{\partial \mathbf{R}_{eH}^2}{\partial \tilde{\mathbf{u}}_H^1}\right] & \left[\frac{\partial \mathbf{R}_{eH}^2}{\partial \tilde{\mathbf{u}}_H^2}\right] & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & \left[\frac{\partial \mathbf{R}_{eH}^{n-1}}{\partial \tilde{\mathbf{u}}_H^{n-2}}\right] & \left[\frac{\partial \mathbf{R}_{eH}^{n-1}}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right] & 0 \\ 0 & 0 & \cdots & 0 & \left[\frac{\partial \mathbf{R}_{eH}^n}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right] & \left[\frac{\partial \mathbf{R}_{eH}^n}{\partial \tilde{\mathbf{u}}_H^n}\right] \end{bmatrix}_{\tilde{\mathbf{u}}_H} \tag{6.12}$$

Then the transpose of the full unsteady Jacobian matrix shown above yields an upper-block bi-diagonal matrix, and Eq. (6.11) yields the following matrix form:

$$\begin{bmatrix} \left[\frac{\partial \mathbf{R}_{eH}^{1}}{\partial \tilde{\mathbf{u}}_H^1}\right]^T & \left[\frac{\partial \mathbf{R}_{eH}^{2}}{\partial \tilde{\mathbf{u}}_H^1}\right]^T & 0 & \cdots & 0 & 0 \\ 0 & \left[\frac{\partial \mathbf{R}_{eH}^{2}}{\partial \tilde{\mathbf{u}}_H^2}\right]^T & \left[\frac{\partial \mathbf{R}_{eH}^{3}}{\partial \tilde{\mathbf{u}}_H^2}\right]^T & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & & \left[\frac{\partial \mathbf{R}_{eH}^{n-1}}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right]^T & \left[\frac{\partial \mathbf{R}_{eH}^{n}}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right]^T \\ 0 & 0 & \cdots & \cdots & 0 & \left[\frac{\partial \mathbf{R}_{eH}^{n}}{\partial \tilde{\mathbf{u}}_H^{n}}\right]^T \end{bmatrix}_{\tilde{\mathbf{u}}_H} \begin{bmatrix} \boldsymbol{\lambda}_h^1 \\ \boldsymbol{\lambda}_h^2 \\ \vdots \\ \boldsymbol{\lambda}_h^{n-1} \\ \boldsymbol{\lambda}_h^n \end{bmatrix} = \begin{bmatrix} \frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^1}^T \\ \frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^2}^T \\ \vdots \\ \frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^{n-1}}^T \\ \frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^n}^T \end{bmatrix}_{\tilde{\mathbf{u}}_H} \tag{6.13}$$

This matrix form implies that the unsteady adjoint problem can be solved by block back substitution, cooresponding to a backward integration in time. In particular, the adjoint problem at the final time step n is first solved by:

$$\left[\frac{\partial \mathbf{R}_{eH}^{n}}{\partial \tilde{\mathbf{u}}_H^n}\right]_{\tilde{\mathbf{u}}_H}^T \boldsymbol{\lambda}_H^n = \left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H^n}\right)_{\tilde{\mathbf{u}}_H}^T \tag{6.14}$$

Then, the adjoint solution at the previous time step $n-1$ is solved as:

$$\left[\frac{\partial \mathbf{R}_{eH}^{n-1}}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right]_{\tilde{\mathbf{u}}_H}^T \boldsymbol{\lambda}_H^{n-1} = \left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right)_{\tilde{\mathbf{u}}_H}^T - \left[\frac{\partial \mathbf{R}_{eH}^{n}}{\partial \tilde{\mathbf{u}}_H^{n-1}}\right]_{\tilde{\mathbf{u}}_H}^T \boldsymbol{\lambda}_H^n \tag{6.15}$$

Thus, the adjoint solution at time step $n-1$ requires the adjoint solution at time step n, which has been already solved. This procedure is performed repeatedly until the initial time step is reached. Note that the formulation for the linearization of the unsteady residual vector with respect to the unsteady flow variables at two successive time steps can be derived based on Eq. (5.6) as:

$$\left[\frac{\partial \mathbf{R}_{eH}^{i}}{\partial \tilde{\mathbf{u}}_H^i}\right]_{\tilde{\mathbf{u}}_H}^T = \left[\frac{M}{\Delta t_H}\right]^T + \left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}}\right]_{\tilde{\mathbf{u}}_H^i}^T \tag{6.16}$$

$$\left[\frac{\partial \mathbf{R}_{eH}^{i}}{\partial \tilde{\mathbf{u}}_H^{i-1}}\right]_{\tilde{\mathbf{u}}_H}^T = -\left[\frac{M}{\Delta t_H}\right]^T \tag{6.17}$$

where M and $\mathbf{R}$ represent the mass matrix and the spatial residual, respectively. $[\partial \mathbf{R}/\partial \tilde{\mathbf{u}}]_{\tilde{\mathbf{u}}_H^i}$ corresponds to the Jacobian of the spatial residual alone (c.f. Eq. (2.15)) which must be evaluated at the appropriate unsteady flow solution state, $\tilde{\mathbf{u}}_H^i$, at time step i. Once the coarse level unsteady adjoint solution, $\{\boldsymbol{\lambda}_H^i, i = 1, \cdots, n\}$, is solved, an approximation to the fine level adjoint solution is then obtained by performing a reconstruction postprocessing procedure of the coarse level unsteady adjoint solution onto the refined temporal resolution mesh via cubic spline interpolation, denoted as:

$$\boldsymbol{\lambda}_H^h = \tilde{\mathcal{I}}_H^h \boldsymbol{\lambda}_H, \tag{6.18}$$

Replacing the exact fine level unsteady adjoint solution with the reconstructed adjoint solution, Eq. (6.9) yields:

$$L_h^f(\tilde{\mathbf{u}}_h) \approx L_h^f(\tilde{\mathbf{u}}_H^h) \underbrace{- {\boldsymbol{\lambda}_H^h}^T \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a^t} \tag{6.19}$$

or

$$L_h^f(\tilde{\mathbf{u}}_h) - L_H^f(\tilde{\mathbf{u}}_H) \approx L_h^f(\tilde{\mathbf{u}}_H^h) - L_H^f(\tilde{\mathbf{u}}_H) \underbrace{- {\boldsymbol{\lambda}_H^h}^T \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a^t} \tag{6.20}$$

where ε_a^t denotes the computable error correction term which is expressed as the inner product of the local unsteady residuals with the reconstructed unsteady adjoint solution for all the children time-steps at the coarse temporal resolution level, expressed as:

$$\varepsilon_a^t = \sum_{i=1}^{n} (\boldsymbol{\lambda}_H^h)^{i^T} \mathbf{R}_{eh}^i(\tilde{\mathbf{u}}_H^h) \tag{6.21}$$

ε_a^t results in a functional error distribution in time, which enables the identification of regions corresponding to relatively large error distributions.

An Alternate Approach to the Adjoint Problem

In the preceding section, a general approach is described for using the projected coarse time level adjoint solution as an approximation to the fine time level adjoint solution. However, in our experience, for the first-order BDF1 scheme the cubic spline interpolation used for projecting the coarse time level adjoint solution onto

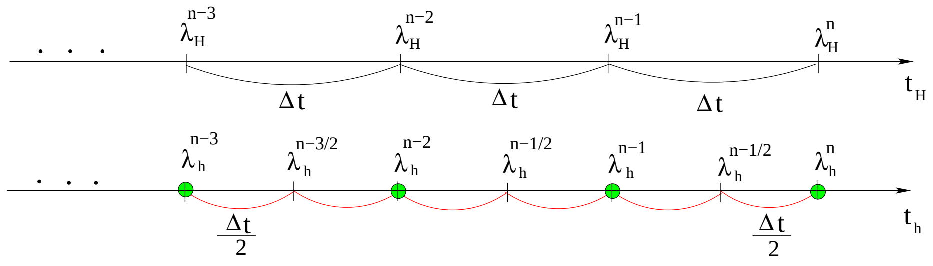


Figure 6.1: Illustration of the adjoint problems in the coarse and fine level time domains.

the fine temporal resolution level is not sufficiently accurate to provide a satisfactory approximation to the fine time level adjoint solution. Increasing the order of the interpolation function from cubic to quintic has been found to have little beneficial effect. This is due to the fact that unsteady flow features that can be captured in a coarse time level do not include sufficient information about the fine time level for complex unsteady flow problems. In order to obtain more information provided in the fine time level, while still restricting the overall computational cost, we propose an alternate approach, in which the adjoint problem is formulated in the fine time level but only solved at the discrete locations which coincide with the coarse level time-steps, as illustrated by the green points in Fig. 6.1. In this approach, the adjoint solution at the final time step n is first evaluated as:

$$\left[\frac{\partial \mathbf{R}_{eh}^{\;n}}{\partial \tilde{\mathbf{u}}_h^n}\right]^T_{\tilde{\mathbf{u}}_H^h} \boldsymbol{\lambda}_H^n = \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^n}\right)^T_{\tilde{\mathbf{u}}_H^h} \tag{6.22}$$

Noting that the adjoint problem at the final time step of the fine time level is similar to the corresponding coarse level adjoint problem, however, the fine level time-step size, Δt_h, must be employed in the evaluation of the Jacobian matrix, i.e. $[\partial \mathbf{R}_{eh}^{\;n}/\partial \tilde{\mathbf{u}}_h^n] = M/\Delta t_h + [\partial \mathbf{R}/\partial \tilde{\mathbf{u}}]_{(\tilde{\mathbf{u}}_H^h)^n}$. For the earlier time steps ($i \in [1, n-1]$), the discrete adjoint problem is solved as:

$$\left[\frac{\partial \mathbf{R}_{eh}^{\ i}}{\partial \tilde{\mathbf{u}}_h^i}\right]_{\tilde{\mathbf{u}}_H^h}^T \boldsymbol{\lambda}_H^i = \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^i}\right)_{\tilde{\mathbf{u}}_H^h}^T - \left[\frac{\partial \mathbf{R}_{eh}^{\ i+\frac{1}{2}}}{\partial \tilde{\mathbf{u}}_h^i}\right]_{\tilde{\mathbf{u}}_H^h}^T \left(s_1\boldsymbol{\lambda}_H^i + s_2\boldsymbol{\lambda}_H^{i+1} + s_3\boldsymbol{\lambda}_H^{i+2}\right) \quad \text{or} \quad (6.23)$$

$$\left(\left[\frac{\partial \mathbf{R}_{eh}^{\ i}}{\partial \tilde{\mathbf{u}}_h^i}\right]_{\tilde{\mathbf{u}}_H^h}^T + s_1\left[\frac{\partial \mathbf{R}_{eh}^{\ i+\frac{1}{2}}}{\partial \tilde{\mathbf{u}}_h^i}\right]_{\tilde{\mathbf{u}}_H^h}^T\right)\boldsymbol{\lambda}_H^i = \left(\frac{\partial L_h^f}{\partial \tilde{\mathbf{u}}_h^i}\right)_{\tilde{\mathbf{u}}_H^h}^T - \left[\frac{\partial \mathbf{R}_{eh}^{\ i+\frac{1}{2}}}{\partial \tilde{\mathbf{u}}_h^i}\right]_{\tilde{\mathbf{u}}_H^h}^T \left(s_2\boldsymbol{\lambda}_H^{i+1} + s_3\boldsymbol{\lambda}_H^{i+2}\right) \quad (6.24)$$

In the above equations, the unknown adjoint variables, $\boldsymbol{\lambda}_h^{i+1/2}$, are replaced by a quadratic interpolation approximation, $s_1\boldsymbol{\lambda}_H^i + s_2\boldsymbol{\lambda}_H^{i+1} + s_3\boldsymbol{\lambda}_H^{i+2}$, in the right-hand-size, where s_1, s_2 and s_3 denote the stencil coefficients for this interpolation function. Recursively integrating this approach backwards to the first time step, the adjoint problem is solved for a total of n time steps, thus ensuring similar cost as the coarse time level adjoint solution. The fine time level adjoint approximation ($\boldsymbol{\lambda}_H^h$) is then obtained by reconstructing the coarse level unsteady adjoint solution onto the fine time level mesh by the cubic interpolation procedure. Based on our experience, this approach provides improved accuracy for estimating functional values and the subsequent adaptation procedure. Thus this method is used in the numerical test cases later discussed for the BDF1 temporal discretization.

6.2.3 Fourth-order Implicit Runge-Kutta Scheme

Formulation

The fine time level functional values obtained using higher-order implicit Runge-Kutta temporal schemes can be approximated using the same approach as described for the first-order BDF1 scheme, and a similar formulation equivalent to Eq. (6.9) can be obtained. However, both unsteady flow solution and unsteady adjoint solution in the IRK schemes involve multistage computations at each time step. Due to the specific ESDIRK class of RK schemes employed in the current work, the flow solution at the first stage of each time step is in fact equal to the solution at the final stage of the previous time step, denoted as $\tilde{\mathbf{u}}_H^{(1),i} = \tilde{\mathbf{u}}_H^{(6),i-1} = \tilde{\mathbf{u}}_H^{i-1}$, where the first bracketed superscript represents an intermediate stage index, and the second superscript denotes the time step index. In this manner, the vectors of the unsteady flow solution and the full unsteady residual for the IRK4 scheme denoted in Eq. (5.11) can be expressed as: $\tilde{\mathbf{u}}_H = \{\tilde{\mathbf{u}}_H^{(s),i}, s = 2, 3, \cdots, 6; i = 1, 2, \cdots, n\}$ and $\mathbf{R}_{eH} = \{\mathbf{R}_{eH}^{(s),i}, s = 2, 3, \cdots, 6; i = 1, 2, \cdots, n\}$, respectively, where n denotes the total number of time steps. Thus, the unsteady adjoint formulation (equivalent to Eq. (6.11)) at the coarse temporal resolution level has the following matrix form for a simplified time domain, involving only two time steps ($n = 2$):

$$
\begin{bmatrix}
\left[\frac{\partial \mathbf{R}_{e\,H}^{(2),1}}{\partial \tilde{\mathbf{u}}_H^{(2),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(3),1}}{\partial \tilde{\mathbf{u}}_H^{(2),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),1}}{\partial \tilde{\mathbf{u}}_H^{(2),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),1}}{\partial \tilde{\mathbf{u}}_H^{(2),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),1}}{\partial \tilde{\mathbf{u}}_H^{(2),1}}\right]^T & 0 & 0 & 0 & 0 & 0 \\
0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(3),1}}{\partial \tilde{\mathbf{u}}_H^{(3),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),1}}{\partial \tilde{\mathbf{u}}_H^{(3),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),1}}{\partial \tilde{\mathbf{u}}_H^{(3),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),1}}{\partial \tilde{\mathbf{u}}_H^{(3),1}}\right]^T & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),1}}{\partial \tilde{\mathbf{u}}_H^{(4),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),1}}{\partial \tilde{\mathbf{u}}_H^{(4),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),1}}{\partial \tilde{\mathbf{u}}_H^{(4),1}}\right]^T & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),1}}{\partial \tilde{\mathbf{u}}_H^{(5),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),1}}{\partial \tilde{\mathbf{u}}_H^{(5),1}}\right]^T & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),1}}{\partial \tilde{\mathbf{u}}_H^{(6),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(2),2}}{\partial \tilde{\mathbf{u}}_H^{(6),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(3),2}}{\partial \tilde{\mathbf{u}}_H^{(6),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),2}}{\partial \tilde{\mathbf{u}}_H^{(6),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),2}}{\partial \tilde{\mathbf{u}}_H^{(6),1}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),2}}{\partial \tilde{\mathbf{u}}_H^{(6),1}}\right]^T \\
0 & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(2),2}}{\partial \tilde{\mathbf{u}}_H^{(2),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(3),2}}{\partial \tilde{\mathbf{u}}_H^{(2),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),2}}{\partial \tilde{\mathbf{u}}_H^{(2),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),2}}{\partial \tilde{\mathbf{u}}_H^{(2),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),2}}{\partial \tilde{\mathbf{u}}_H^{(2),2}}\right]^T \\
0 & 0 & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(3),2}}{\partial \tilde{\mathbf{u}}_H^{(3),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),2}}{\partial \tilde{\mathbf{u}}_H^{(3),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),2}}{\partial \tilde{\mathbf{u}}_H^{(3),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),2}}{\partial \tilde{\mathbf{u}}_H^{(3),2}}\right]^T \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(4),2}}{\partial \tilde{\mathbf{u}}_H^{(4),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),2}}{\partial \tilde{\mathbf{u}}_H^{(4),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),2}}{\partial \tilde{\mathbf{u}}_H^{(4),2}}\right]^T \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(5),2}}{\partial \tilde{\mathbf{u}}_H^{(5),2}}\right]^T & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),2}}{\partial \tilde{\mathbf{u}}_H^{(5),2}}\right]^T \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{e\,H}^{(6),2}}{\partial \tilde{\mathbf{u}}_H^{(6),2}}\right]^T
\end{bmatrix}_{\tilde{\mathbf{u}}_H}
\begin{bmatrix}
\boldsymbol{\lambda}_H^{(2),1} \\ \boldsymbol{\lambda}_H^{(3),1} \\ \boldsymbol{\lambda}_H^{(4),1} \\ \boldsymbol{\lambda}_H^{(5),1} \\ \boldsymbol{\lambda}_H^{(6),1} \\ \boldsymbol{\lambda}_H^{(2),2} \\ \boldsymbol{\lambda}_H^{(3),2} \\ \boldsymbol{\lambda}_H^{(4),2} \\ \boldsymbol{\lambda}_H^{(5),2} \\ \boldsymbol{\lambda}_H^{(6),2}
\end{bmatrix}
=
\begin{bmatrix}
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(2),1}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(3),1}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(4),1}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(5),1}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(6),1}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(2),2}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(3),2}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(4),2}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(5),2}} \\
\frac{\partial L_H^{f\;T}}{\partial \tilde{\mathbf{u}}_H^{(6),2}}
\end{bmatrix}_{\tilde{\mathbf{u}}_H}
\tag{6.25}
$$

where the lower triangular part of the transpose of the full unsteady Jacobian matrix, $[\partial \mathbf{R}_{eH}/\partial \tilde{\mathbf{u}}_H]^T_{\tilde{\mathbf{u}}_H}$, in the IRK4 scheme also contains only zero entries. The complete matrix for arbitrary time domains ($n > 2$) follows the form given here in a similar manner. Additionally, each individual matrix-block in Eq. (6.25) involves the linearization of an unsteady residual component with respect to the unsteady flow variables at a given intermediate stage of a time step, which can be obtained based on Eq. (5.11) as:

$$\left[\frac{\partial \mathbf{R}_{eH}^{(s),i}}{\partial \tilde{\mathbf{u}}_H^{(s),i}}\right]^T = \left[\frac{M}{\Delta t_H}\right]^T + a_{ss}\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}_H}\right]^T_{\tilde{\mathbf{u}}_H^{(s),i}} \qquad (s \in [2,6], i \in [1,n]) \tag{6.26}$$

$$\left[\frac{\partial \mathbf{R}_{eH}^{(s),i}}{\partial \tilde{\mathbf{u}}_H^{(m),i}}\right]^T = a_{sm}\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}_H}\right]^T_{\tilde{\mathbf{u}}_H^{(m),i}} \qquad (2 \leq m < s, i \in [1,n]) \tag{6.27}$$

$$\left[\frac{\partial \mathbf{R}_{eH}^{(s),i}}{\partial \tilde{\mathbf{u}}_H^{(6),i-1}}\right]^T = -\left[\frac{M}{\Delta t_H}\right]^T + a_{s1}\left[\frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{u}}_H}\right]^T_{\tilde{\mathbf{u}}_H^{(6),i-1}} \qquad (s \in [2,6], i \in [1,n]) \tag{6.28}$$

In the above equations, $[\partial \mathbf{R}/\partial \tilde{\mathbf{u}}_H]^T_{\tilde{\mathbf{u}}_H^{(s),i}}$ corresponds to the transpose of the Jacobian of the spatial residual alone (c.f. Eq. (2.15)) which must be evaluated at the appropriate stage value (s) and time step value (i). $\{a_{sm}\}$ corresponds to the Butcher coefficients of the particular fourth-order implicit IRK4 scheme used in this work, and the values are given in Table 5.2. Since the computation for the adjoint solution at an intermediate stage s of the time step i requires the adjoint solution at later stages, the unsteady adjoint solution for the IRK4 scheme results in a backward integration in time. The formulation for evaluating the first adjoint variables when sweeping backwards in time, which in fact corresponds to the adjoint solution at the final time step, is given by:

$$\left[\frac{\partial \mathbf{R}_{eH}^{(6),n}}{\partial \tilde{\mathbf{u}}_H^{(6),n}}\right]^T \boldsymbol{\lambda}_H^{(6),n} = \frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H^{(6),n}}^T \tag{6.29}$$

Then, the adjoint solution at other intermediate stages in the final time step can be evaluated following the backwards recursion relation as:

$$\left[\frac{\partial \mathbf{R}_{eH}^{(s),n}}{\partial \tilde{\mathbf{u}}_H^{(s),n}}\right]^T \boldsymbol{\lambda}_H^{(s),n} = \frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H^{(s),n}}^T - \sum_{m=s+1}^{6} \left[\frac{\partial \mathbf{R}_{eH}^{(m),n}}{\partial \tilde{\mathbf{u}}_H^{(s),n}}\right]^T \boldsymbol{\lambda}_H^{(m),n} \quad (s = 5, 4, \cdots, 2) \tag{6.30}$$

For earlier time steps $i \in [1, n-1]$, the multistage adjoint solution in the IRK4 scheme is computed based on the following formulations:

$$\left[\frac{\partial \mathbf{R}_{eH}^{(6),i}}{\partial \tilde{\mathbf{u}}_H^{(6),i}}\right]^T \boldsymbol{\lambda}_H^{(6),i} = \frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H^{6,i}}^T - \sum_{m=2}^{6} \left[\frac{\partial \mathbf{R}_{eH}^{(m),i+1}}{\partial \tilde{\mathbf{u}}_H^{(6),i}}\right]^T \boldsymbol{\lambda}_H^{(m),i+1} \tag{6.31}$$

$$\left[\frac{\partial \mathbf{R}_{eH}^{(s),i}}{\partial \tilde{\mathbf{u}}_H^{(s),i}}\right]^T \boldsymbol{\lambda}_H^{(s),i} = \frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H^{s,i}}^T - \sum_{m=s+1}^{6} \left[\frac{\partial \mathbf{R}_{eH}^{(m),i}}{\partial \tilde{\mathbf{u}}_H^{(s),i}}\right]^T \boldsymbol{\lambda}_H^{(m),i} \quad (s \in [2, 5]) \tag{6.32}$$

Note that the right-hand-size of the adjoint formulation in Eq. (6.29) for evaluating the adjoint problem at the final time step ($s = 6, i = n$) is distinct from the one in Eq. (6.31) for the evaluation at an intermediate time step ($s = 6, i \neq n$). The approximated fine level adjoint variables for the IRK4 scheme are obtained by locally interpolating the coarse level adjoint variables to the fine time level using a quintic spline function, expressed as, $\boldsymbol{\lambda}_H^{h*} = \bar{\mathcal{I}}_H^h \boldsymbol{\lambda}_H$, and then performing a few passes (usually $2 \sim 3$ Gauss-Seidel smoothings) on the refined level to update the projected coarse adjoint solutions, expressed as $\boldsymbol{\lambda}_H^h = \mathcal{P}(\boldsymbol{\lambda}_H^{h^*})$. We have found that this fine-level update approach performs more effectively and is simpler to implement than the extended strategy used for the BDF1 scheme. The discrete functional error between the fine time level functional evaluated with the exact fine level flow solution and the fine time level functional evaluated with the projected coarse time level flow solution is approximated as:

$$L_h^f(\tilde{\mathbf{u}}_h) - L_h^f(\tilde{\mathbf{u}}_H^h) \approx \underbrace{-\boldsymbol{\lambda}_H^h \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a^t} \tag{6.33}$$

or

$$L_h^f(\tilde{\mathbf{u}}_h) - L_H^f(\tilde{\mathbf{u}}_H) \approx L_h^f(\tilde{\mathbf{u}}_H^h) - L_H^f(\tilde{\mathbf{u}}_H) \underbrace{-\boldsymbol{\lambda}_H^h \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)}_{\varepsilon_a^t} \tag{6.34}$$

where $\tilde{\mathbf{u}}_H^h$ denotes the projected coarse time level flow solution ($\tilde{\mathbf{u}}_H^h = \bar{\mathcal{I}}_H^h \tilde{\mathbf{u}}_H$) which is obtained by using the aforementioned quintic spline interpolation function. ε_a^t represents the adjoint correction term for the IRK4 scheme, which is expressed by the same formulation as denoted for the first-order accurate BDF1 scheme, however, noting that the adjoint correction term in the IRK4 scheme involves the inner product of the temporal adjoint variables and local unsteady residuals for all intermediate stages in the time domain, expressed as:

$$\varepsilon_a^t = \sum_{i=1}^{n} \sum_{s=2}^{6} -(\boldsymbol{\lambda}_H^h)^{(s),i^T} \mathbf{R}_{e_h}^{(s),i}(\tilde{\mathbf{u}}_H^h) \tag{6.35}$$

6.3 Estimation of Algebraic Error Due to Partial Convergence

The preceding sections provide the fundamental approach for estimating the temporal error in a specified time-dependent functional output. This section considers another error source, namely that which arises from incomplete or partial convergence in the implicit flow problem at each time step.

Partial convergence at each implicit time step of an unsteady flow problem expedites the nonlinear problem solution but simultaneously introduces additional error to the flow solution and therefore may affect the accuracy of the functional of interest. However, a quantitative accuracy study can be performed in order to ensure that the use of partial convergence tolerances produces negligibly small errors in functional values, compared with functional errors arising from other sources, such as temporal discretization errors. In this context, the adjoint-based error estimation technique developed in the previous sections is extended to provide an estimate of error due to partial convergence.

Consider an unsteady flow solution, $\underline{\tilde{\mathbf{u}}}_H$, obtained by partially converging each implicit problem in the current time domain using a specified convergence tolerance set, denoted as $\{tol_i, i = 1, 2, \cdots, n\}$, where n denotes the total number of time steps.

The coarse time level functional evaluated with a fully converged (to machine zero) flow solution $\tilde{\mathbf{u}}_H$ can be expanded with respect to the partially converged flow solution based on a Taylor series expansion as:

$$L_H^f(\tilde{\mathbf{u}}_H) = L_H^f(\underline{\tilde{\mathbf{u}}}_H) + \left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H}\right)_{\underline{\tilde{\mathbf{u}}}_H} (\tilde{\mathbf{u}}_H - \underline{\tilde{\mathbf{u}}}_H) + \cdots \tag{6.36}$$

The coarse level residual can also be expanded with respect to the partially converged flow solution, $\underline{\tilde{\mathbf{u}}}_H$, as:

$$\mathbf{R}_{eH}(\tilde{\mathbf{u}}_H) = \mathbf{R}_{eH}(\underline{\tilde{\mathbf{u}}}_H) + \left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]_{\underline{\tilde{\mathbf{u}}}_H} (\tilde{\mathbf{u}}_H - \underline{\tilde{\mathbf{u}}}_H) + \cdots = 0 \tag{6.37}$$

The discrete solution error between fully and partially converged solutions is approximated as:

$$\tilde{\mathbf{u}}_H - \underline{\tilde{\mathbf{u}}}_H \approx -\left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]^{-1}_{\underline{\tilde{\mathbf{u}}}_H} \mathbf{R}_{eH}(\underline{\tilde{\mathbf{u}}}_H) \tag{6.38}$$

Note that $\mathbf{R}_{eH}(\underline{\tilde{\mathbf{u}}}_H) \neq 0$ since $\underline{\tilde{\mathbf{u}}}_H$ is not the exact discrete solution. Substitute Eq. (6.38) into Eq. (6.36), the functional is approximated as:

$$L_H^f(\tilde{\mathbf{u}}_H) \approx L_H^f(\underline{\tilde{\mathbf{u}}}_H) - \left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H}\right)_{\underline{\tilde{\mathbf{u}}}_H} \left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]^{-1}_{\underline{\tilde{\mathbf{u}}}_H} \mathbf{R}_{eH}(\underline{\tilde{\mathbf{u}}}_H) \tag{6.39}$$

Replacing the term $\left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H}\right)_{\underline{\tilde{\mathbf{u}}}_H} \left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]^{-1}_{\underline{\tilde{\mathbf{u}}}_H}$ with the adjoint variables $(\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H}$ yields:

$$\left[\frac{\partial \mathbf{R}_{eH}}{\partial \tilde{\mathbf{u}}_H}\right]^{T}_{\underline{\tilde{\mathbf{u}}}_H} (\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H} = \left(\frac{\partial L_H^f}{\partial \tilde{\mathbf{u}}_H}\right)^{T}_{\underline{\tilde{\mathbf{u}}}_H} \tag{6.40}$$

Due to the fact that the transpose of the unsteady Jacobian matrix and the objective sensitivity vector are evaluated with the partially converged flow solution rather than the exact flow solution, the adjoint solution is denoted as $(\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H}$ to be distinguished from $\boldsymbol{\lambda}_H$. Moreover, the computational procedure to obtain $(\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H}$ follows the same procedure as discussed for either the BDF1 scheme or the IRK4 scheme in the preceding sections, but only operates in the current discretized time domain. The functional evaluated by the exact discrete flow solution is thus approximated as:

$$L_H^f(\tilde{\mathbf{u}}_H) \approx L_H^f(\underline{\tilde{\mathbf{u}}}_H) \underbrace{-(\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H}^T \mathbf{R}_{eH}(\underline{\tilde{\mathbf{u}}}_H)}_{\varepsilon_a^c} \tag{6.41}$$

where ε_a^c represents the adjoint-based error correction term which provides the estimated functional error invoked by the use of the set of residual convergence tolerances, and can be further expressed for the respective BDF1 and the IRK schemes as:

$$\text{BDF1} : \varepsilon_a^c = \sum_{i=1}^{n} -(\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H}^{i}{}^T \mathbf{R}_{eH}^{\,i}(\underline{\tilde{\mathbf{u}}}_H) \tag{6.42}$$

$$\text{IRK4} : \varepsilon_a^c = \sum_{i=1}^{n} \sum_{s=2}^{6} -(\boldsymbol{\lambda}_H)_{\underline{\tilde{\mathbf{u}}}_H}^{(s),i^T} \mathbf{R}_{eH}^{(s),i}(\underline{\tilde{\mathbf{u}}}_H) \tag{6.43}$$

Although this can be used for adaptively reducing the algebraic error [75], the evaluation of the term ε_a^c is primarily used to ensure the algebraic error has negligible effects on the predicted temporal functional error in this work.

6.4 Refinement Criteria

The adjoint correction ε_a^t results in a functional error distribution at the coarse temporal resolution level. An individual error contribution associated with the time step i ($i \in [1, n]$) in the BDF1 temporal scheme is evaluated as:

$$\varepsilon_{a,i}^t = -(\boldsymbol{\lambda}_H^h)^{i^T} \mathbf{R}_{eh}^{\,i}(\tilde{\mathbf{u}}_H^h) \tag{6.44}$$

where the local time-step error indicator for the time step i is expressed as the inner product of the local unsteady residual vector with the reconstructed adjoint variables within the children time-steps of the coarse time step i. For the fourth-order IRK4 scheme, the evaluation of this local error indicator for the time step i involves the inner product of the corresponding state variables at all intermediate stages within the children time-steps, shown as:

$$\varepsilon_{a,i}^t = -\sum_{s=2}^{\mathcal{S}} (\boldsymbol{\lambda}_H^h)^{(s),i^T} \mathbf{R}_{eh}^{(s),i}(\tilde{\mathbf{u}}_H^h) \tag{6.45}$$

In order to adaptively reduce error in the objective functional, the error-balancing refinement criterion [49] used in the adjoint-based spatial mesh adaptation is employed in the current work, where a time step is marked for refinement if the absolute value of a local error indicator satisfies,

$$|\varepsilon_{a,i}^{t}| > \frac{E_{tol}^{t}}{n} \tag{6.46}$$

where E_{tol}^{t} denotes a positive user-specified global tolerance for the functional output. This criteria identifies the time steps in the coarse temporal resolution level, which have relatively large error contributions to the global functional error. For time-steps where the relation of Eq. (6.46) holds, a marked time step is refined by dividing the local time step into two equivalent time steps. The adaptive algorithm terminates if all local error contributions are within the maximum allowable equidistributed local error, E_{tol}^{t}/n.

6.5 Results

Numerical test cases are used to examine the effectiveness of the error correction term provided by the discrete adjoint sensitivity technique, and to demonstrate the performance of the adaptive refinement schemes versus a uniform time-step refinement method. The first test case revisits the shedding flow over a triangular wedge used in Section 5.4.2 for the first-order BDF1 scheme, while the second test case considers the convection of an isentropic vortex described in Section 5.4.1 for the fourth-order implicit IRK4 scheme.

6.5.1 Shedding Flow over a Triangular Wedge

The problem of flow over a triangular wedge is computed using a mesh of 3684 elements with a third-order ($p = 2$) spatial discretization scheme and a first-order temporal discretization scheme. Fig. 6.2 illustrates a typical solution snapshot for this problem.

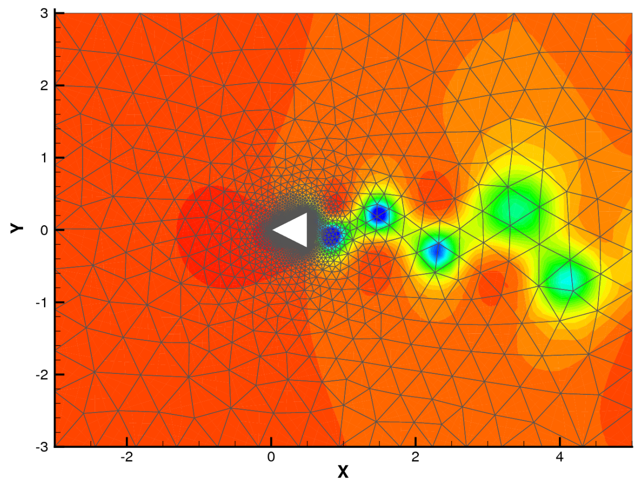


Figure 6.2: Computational mesh and density contours for shedding flow over a triangular wedge.

Error Prediction for Two Time Resolution Levels

The effectiveness of the adjoint-based error correction term for the BDF1 scheme is examined in this section for two test functional outputs. The functional of interest in Case I is set to be the drag at the final time $T = 5.0$, and in Case II the functional of interest is set to be the time-integrated drag over the time interval $[0, T]$, expressed as:

$$L^f = \int_0^T D(t)dt \tag{6.47}$$

where $D(t)$ denotes the function of time-dependent drag and $T = 4.2$. The convergence tolerance at each time step is set to be machine precision of 10^{-16} in order to eliminate any sources of algebraic error due to incomplete convergence.

Tables 6.1 and 6.2 show functional error computed between two successively refined discretization levels in the time domain, for Case I and Case II respectively. In these two tables, the first two columns refer to the successively refined temporal resolution levels and the corresponding total numbers of time steps to evaluate the specified objective functional, respectively. The temporal resolution, represented by the number of time steps n, is successively increased by uniformly refining each

Table 6.1: Case I: comparisons of the error corrections computed by using exact fine adjoint solution and reconstructed fine adjoint solution, with the computed functional error between two successively refined levels. Objective functional is set to be the drag at the final time, $T = 5$. ε_{a*}^t and ε_a^t correspond to the error correction term denoted in Eq. (6.9) and Eq. (6.19), respectively.

Levels	Steps	$L_h^f(\tilde{\mathbf{u}}_h) - L_h^f(\tilde{\mathbf{u}}_H^h)$	ε_{a*}^t (based on $\boldsymbol{\lambda}_h$)	ε_a^t (based on $\boldsymbol{\lambda}_H^h$)
1	20	6.409423983344E-03	5.8953862304640E-03	6.1370526380238E-03
2	40	3.363150411947E-03	3.2090618101287E-03	3.3192713144015E-03
3	80	1.740612661084E-03	1.6772693740095E-03	1.6976435495301E-03
4	160	9.307328650283E-04	9.1940300005702E-04	9.1424908868082E-04
5	320	5.035059191103E-04	5.0213552906005E-04	5.0091023003573E-04
6	640	2.653264156995E-04	2.6527239219869E-04	2.6507952427098E-04

Table 6.2: Case II: comparisons of the error corrections computed by using exact fine adjoint solution and reconstructed fine adjoint solution, with the computed functional error between two successively refined levels. Objective functional is set to be the time-integrated drag over the time interval $[0, 4.2]$. ε_{a*}^t and ε_a^t correspond to the error correction term denoted in Eq. (6.9) and Eq. (6.19), respectively.

Levels	Steps	$L_h^f(\tilde{\mathbf{u}}_h) - L_h^f(\tilde{\mathbf{u}}_H^h)$	ε_{a*}^t (based on $\boldsymbol{\lambda}_h$)	ε_a^t (based on $\boldsymbol{\lambda}_H^h$)
1	70	4.594582618541E-03	4.509605838075E-03	4.521545814133E-03
2	140	2.4913814884640E-03	2.456291402865E-03	2.451742657086E-03
3	280	1.324672974414E-03	1.317402757018E-03	1.316237351589E-03
4	560	6.728072622733E-04	6.756449437449E-04	6.761561824491E-04
5	1120	3.282112538086E-04	3.302315506682E-04	3.304402325325E-04
6	2240	1.579403063223E-04	1.589476088374E-04	1.590088887837E-04

time step through a 1:2 subdivision. $L_h^f(\tilde{\mathbf{u}}_h) - L_h^f(\tilde{\mathbf{u}}_H^h)$ in the third column denotes functional value difference between the fine time level functional evaluated using the computed exact fine time level flow state and the fine time level functional evaluated using the projected coarse time level flow state. This corresponds to the term with which the adjoint correction term should be compared (c.f. Eq.(6.19)). The last two (i.e. the fourth and the fifth) columns correspond to the predicted error correction terms obtained by using the fine time level adjoint solution and the reconstructed fine level adjoint solution computed based on the extending approach (c.f. Equations (6.22)-(6.24)). It is shown that the reconstructed adjoint error estimate becomes more accurate with increasing levels of time-step refinement. In Case I, the correction term evaluated using the reconstructed adjoint solution agrees to within 96.0% of the corresponding error evaluated using the fine time level adjoint solution at the first level. However this agreement increases to 99.9 % at the sixth level. This indicates that the reconstruction procedure described in the alternate approach (c.f. Equations (6.22) – (6.24)) for the fine time level adjoint solution is sufficient to approximate the corresponding functional error. Moreover, the computable adjoint error correction term becomes more accurate for estimating the true functional error with increasing refinement levels. This is due to the fact that the correction term is derived based on the linearization of the objective functional. For example, in Case II, the effectiveness of the adjoint correction term, defined as $\varepsilon_a^t / \left(L_h^f(\tilde{\mathbf{u}}_h) - L_h^f(\tilde{\mathbf{u}}_H^h) \right)$, is 0.9840 at the first level and 1.0006 at the sixth level.

Adaptive Time-step Refinement

The adjoint-based error estimates, which have been validated in the previous section, are now used to drive an adaptive time-step refinement scheme for reducing the temporal error in the functional output at reduced computational cost, for the shedding flow over a triangular wedge using the BDF1 scheme. A uniform time-step of $\Delta t = 0.06$ is used initially in the BDF1 temporal discretization scheme. The objective functional of interest is set to be the time-integrated drag from $t = 0$ to $t = 5.1$. The p-Multigrid algorithm driven by the element Gauss-Seidel scheme is employed to ac-

Table 6.3: Adjoint-based functional error predictions for all cycles of adaptive time-step refinement in the case of shedding flow over a triangular wedge.

Cycle	ε_a^t	ε_a^c
0	5.5165152746945223E-3	-7.9451904055911315E-6
1	2.8930705943262991E-3	-1.3844562551600945E-6
2	1.1870898137999135E-3	-2.3243979370230301E-8
3	7.8343237101801062E-4	1.4215369712263815E-7
4	2.0210165705981993E-4	2.1716977219027283E-7

celerate convergence of each non-linear problem arising from the implicit method. An initial convergence tolerance of 10^{-6} is employed at each individual time-integration step.

Although the adjoint procedure can be used to estimate and adaptively reduce temporal and algebraic error simultaneously [75], in this case the adjoint-based algebraic error estimate is used only to verify that the convergence tolerance is set adequately, in that the algebraic errors in the functional output are much smaller than the temporal error, while the temporal error is reduced through adaptive time-step refinement. This is clearly indicated in Table 6.3, where the estimates of both temporal and algebraic error at each refinement level are compared, showing that the algebraic errors are consistently several orders of magnitude lower than the temporal error, thus justifying the use of prescribed convergence tolerance of 10^{-6}.

Fig. 6.3 shows the functional temporal error distributions in time for all temporal adaptation cycles, where it is clearly shown that a relatively large functional error occurs in the time interval $[0, 1]$, as illustrated by the red line in this figure. This is due to the fact that the flow separation from the two sharp corners of the triangular wedge results in increased pressure drag and therefore the resulting period of time is of significant importance to the functional accuracy. On the other hand, the fourth adaptation cycle reveals a much more uniform functional error distribution. Figure 6.4 depicts the resulting adaptively determined time-step sizes in the time domain for all adaptation cycles, compared with the initial constant time-step size. Rather than choosing a smaller time-step size for all integration steps, the adaptive time-step

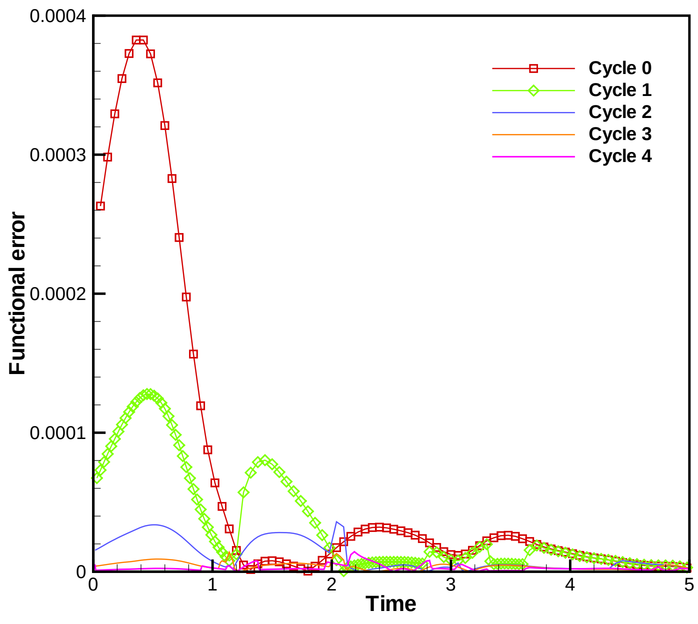


Figure 6.3: Functional error distributions for all temporal adaptation cycles in the case of shedding flow over a triangular wedge.

refinement scheme only refines steps with relatively large error contributions, and thus Fig. 6.4 shows that most time-step size refinement occurs in the time interval $[0, 2]$.

Next, the convergence behavior of the objective functional error for the adaptively refined time-step simulations is compared with that achieved using uniformly refined time-step simulations, where each time step in the time-integration scheme is refined by the aforementioned 1:2 subdivision, regardless of the objective functional error contribution. The implicit flow problem at each time step of the BDF1 scheme in the uniform time-step refinement approach is fully converged to machine precision of 10^{-16}. The objective functional value obtained from the uniform time-step refinement approach represents the optimal functional error reduction achievable for the corresponding adaptive time-step refinement approach. Fig. 6.5 illustrates the results of functional error convergence for the uniform time-step refinement approach, the adaptive time-step refinement approach, and for the adaptive approach corrected with the adjoint error estimate. In terms of total number of time steps or degrees of freedom (DOF), as illustrated in the left-hand-side figure, the adaptively refined time-step approach achieves equivalent functional error reductions as the uniformly

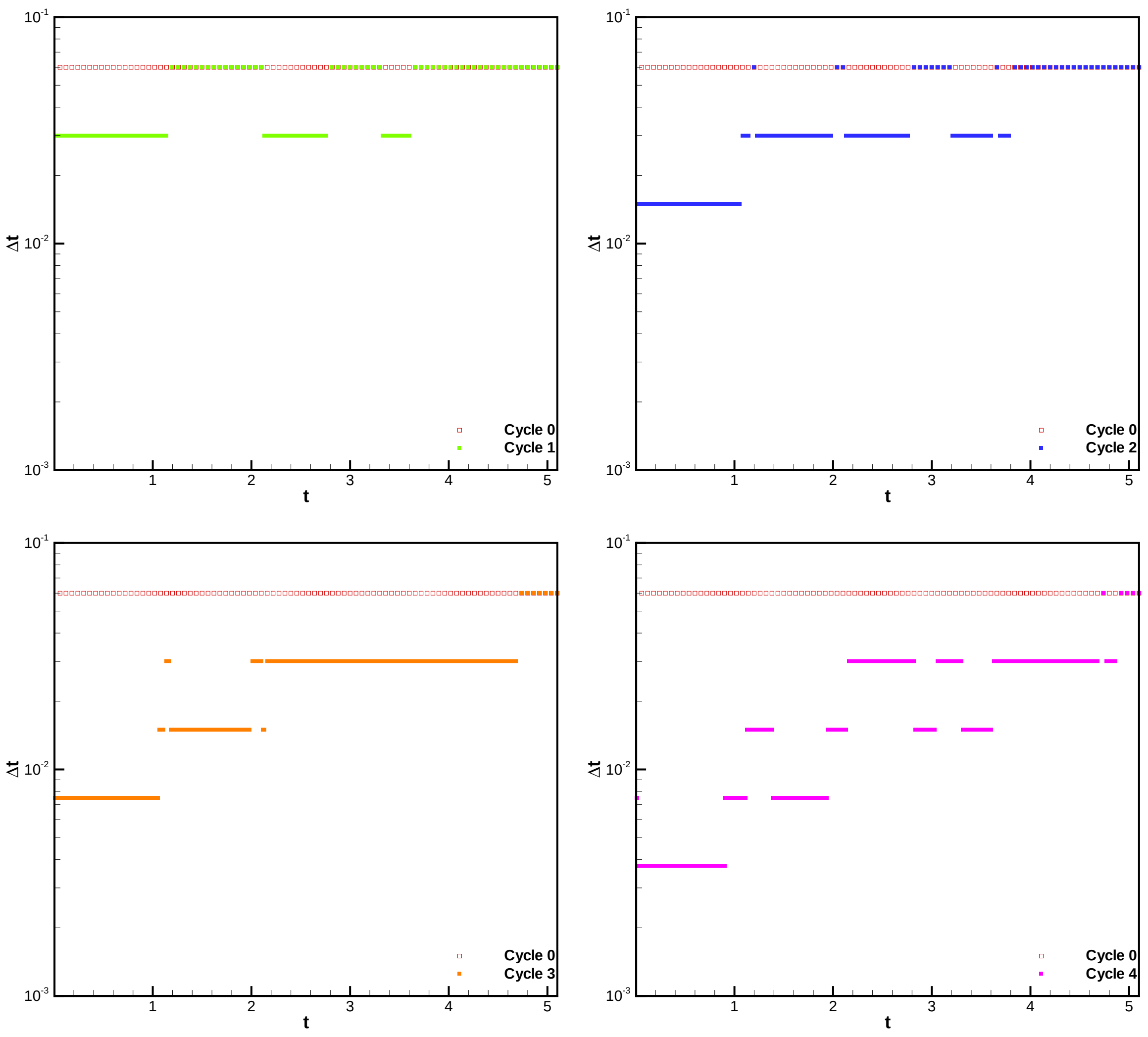
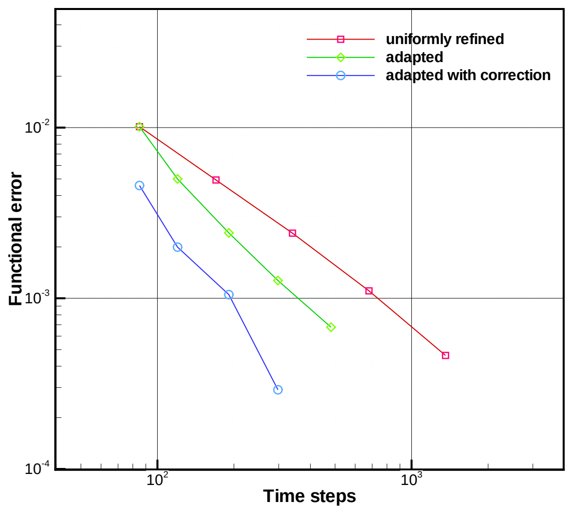


Figure 6.4: Distribution of time-step sizes for all temporal adaptation cycles in the case of shedding flow over a triangular wedge.

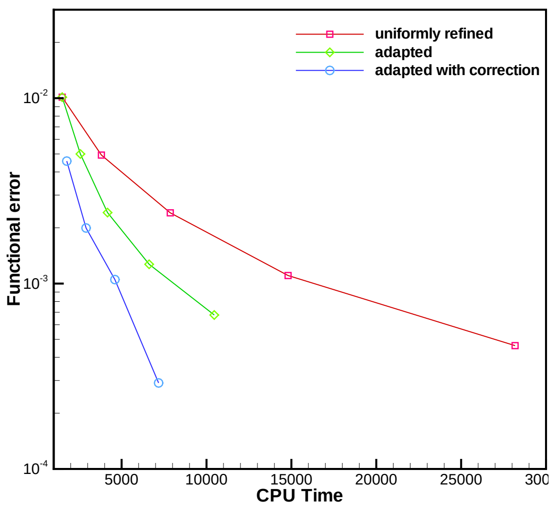
refined time-step approach at each adaptation cycle with substantially fewer time steps in the time-integration scheme, and the benefits become more evident with increasing adaptation cycles. This implies that the uniform time-step refinement approach contains excessive temporal resolution in some regions which have little influence on the time-dependent functional accuracy. Moreover, as represented by the blue line in this figure, the adjoint-based correction term provides very accurate prediction of the functional value for the next refinement level, further increasing the accuracy/efficiency of this procedure. In terms of computational cost, as depicted in the right-hand-side figure, the adaptively refined time-step approach is consistently more efficient than the uniform time-step refinement approach, although the cost for one adaptation cycle includes the cost of both the flow and the adjoint solutions at the current adaptation level as well as the cost of its previous adaptation cycles. Furthermore, the adaptive approach with adjoint-based error corrections incurs significantly lower computational cost than the other approaches to achieve the same level of accuracy. For example, to achieve an accuracy level of 6×10^{-4}, the adaptively refined time-step approach exhibits a speedup of 2.4 over the uniformly refined time-step approach, and the adaptive approach with correction exhibits a speedup of 4.5 over the uniform refinement method. Although the comparisons of computational effect include the additional cost of solving the adjoint problem in the adaptive case, the comparison may be somewhat optimistic since the uniform refinement cases are converged to the machine precision of 10^{-16}, due to the absence of an algebraic error estimate in these cases. The objective convergence is illustrated in Fig. 6.6 as a function of time steps for the involved adaptation cycles, demonstrating the effectiveness of the adjoint-based temporal error correction term for predicting the functional value at the next finer refinement level. Therefore, the adjoint-based temporal error correction can be used to predict the finer-level functional value.

6.5.2 Isentropic Vortex Convection

The next example involves a temporal adaptation test case for the fourth-order implicit Runge-Kutta scheme using the case of the convecting isentropic vortex discussed in Section 5.4.1. Similarly the main tasks include verification of the functional error predictions between two successively refined time-step levels and comparison of the adaptively refined time-step approach with the uniform time-step refinement approach. Fig. 6.7 illustrates the initial density contours and the region used to define the output functional (as represented by the dark red line in the figure). The functional of interest is set to be the pressure integrated over the length $y \in [-1, 1]$ at $x = 2.0$ at the final time $T = 4$. A fifth-order accurate ($p = 4$) spatial discretization and the implicit fourth-order IRK4 scheme are employed for all the following temporal adaptation cases.



(a) Error vs. Time steps (i.e. DOF)



(b) Error vs. Computational cost

Figure 6.5: Functional error convergence for the uniform time-step refinement approach, the adaptive time-step refinement approach and the adaptive approach corrected with the adjoint-based correction in the case of shedding flow over a triangular wedge.

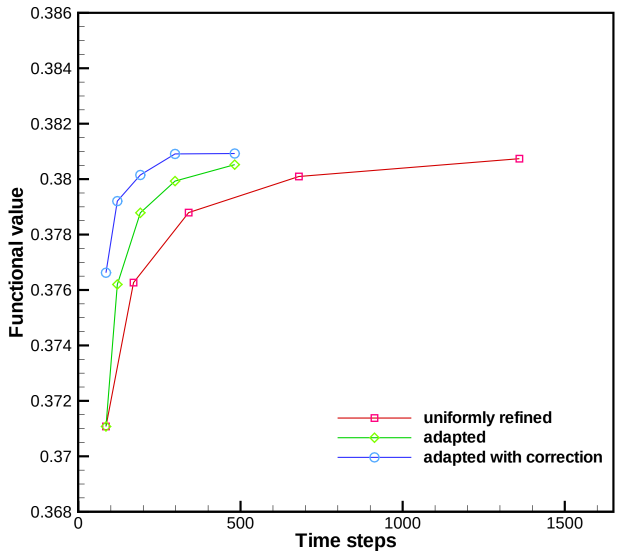


Figure 6.6: Objective functional convergence for the uniform time-step refinement approach, the adaptive time-step refinement approach and the adaptive approach corrected with the adjoint-based correction in the case of shedding flow over a triangular wedge.

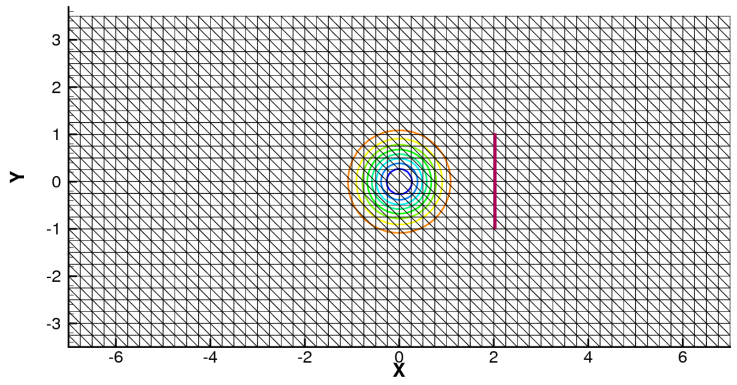


Figure 6.7: Initial density contours and location of objective functional integral as represented by the dark red line for the test case of isentropic vortex convection.

Error Prediction For Two Mesh Levels

In this section, we validate the error estimation procedure by verifying that the change in the functional value between a coarse and uniformly refined time-step simulation is well predicted by the adjoint procedure. The convergence tolerance at each intermediate stage of the IRK4 scheme is set to machine precision of 10^{-16} to eliminate the algebraic error due to incomplete convergence. Table 6.4 shows a comparison of the results for the true functional change using two successive time resolution levels constructed by uniformly refining the time-step sizes through the 1:2 subdivision, and the adjoint error correction evaluated using the fine time level adjoint solution, and also the corresponding estimate evaluated using the reconstructed fine adjoint solution (i.e. using 3 Gauss-Seidel smoothings at the fine temporal resolution level). It can be observed that the reconstructed adjoint solution provides good agreement with the fine adjoint error estimate, and moreover, the adjoint-based error estimate is very accurate compared with the computed exact functional error at a given level. The adjoint correction evaluated using the reconstructed adjoint solution agrees to 94.8% of the true two-level functional error at the first level and increases to 99.9% at the fifth level.

Table 6.4: Comparisons of the error predictions obtained by using fine adjoint and reconstructed fine adjoint, and the computed functional error of the two levels, implemented using the IRK4 scheme, where $\varepsilon_{a*}^{t} = \boldsymbol{\lambda}_h{}^T \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)$ and $\varepsilon_a^t = \boldsymbol{\lambda}_H^h{}^T \mathbf{R}_{eh}(\tilde{\mathbf{u}}_H^h)$.

Levels	Steps	$L_h^f(\tilde{\mathbf{u}}_h) - L_h^f(\tilde{\mathbf{u}}_H^h)$	ε_{a*}^t (based on $\boldsymbol{\lambda}_h$)	ε_a^t (based on $\boldsymbol{\lambda}_H^h$)
1	4	1.571246466096E-03	1.5632947696232E-03	1.4896958322805E-03
2	8	8.319785492716E-05	8.2985882045335E-05	7.9661163552977E-04
3	16	7.250435426930E-06	7.2493664466036E-06	7.2459985983245E-06
4	32	4.340636969857E-07	4.3391789467386E-07	4.3394787466313E-07
5	64	2.694129896951E-08	2.6933615668172E-08	2.6930715973195E-08

Adaptive Time-step Refinement

In this section, the performance of the temporal adaptation procedure is examined for the fourth-order implicit Runge-Kutta scheme. The adaptive refinement test starts with the initial condition illustrated in Fig. 6.7 as well as the employment of a uniform time-step size of $\Delta t = 0.8$. The objective functional used in the previous section for validating the error predictions is also employed in this example. The convergence tolerance is set to be 10^{-8} for all time steps. Similarly to the results shown in the case using the BDF1 scheme, the algebraic error resulting from this convergence tolerance level as estimated by the adjoint procedure is still significantly smaller than the predicted temporal error, and thus only time-step refinement is considered.

Fig. 6.8 illustrates the error distribution in time for all adaptation cycles. For the initial selection of time-step sizes (i.e. uniform), the functional error exhibits larger error contributions in regions close to the final time. This is because the location of the vortex at the first three time steps is not close to the location of integration for the objective functional and the objective functional is evaluated at the final time. Therefore, the solution at this region in time has little influence on the final functional accuracy. As the vortex travels closer to the target area, the accuracy of the unsteady flow solution becomes more important for the evaluation of the target functional. On the other hand, the third adaption cycle reveals a much more uniform functional error distribution due to the time-step refinement procedure.

The error convergence history of the adaptively refined time-step simulations

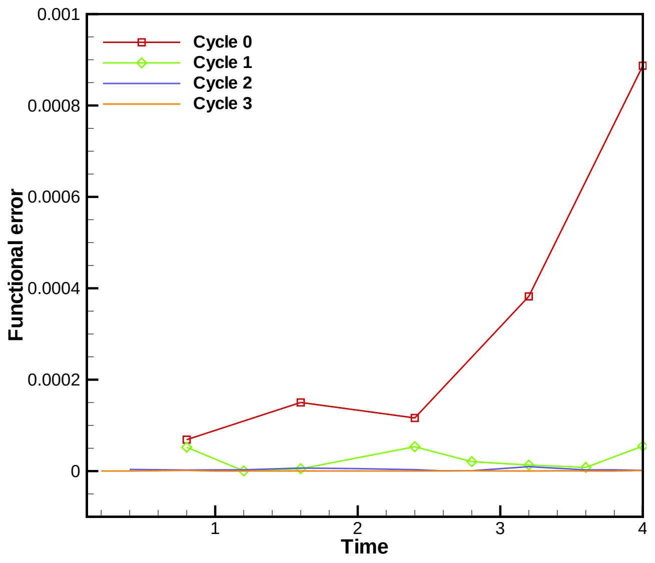
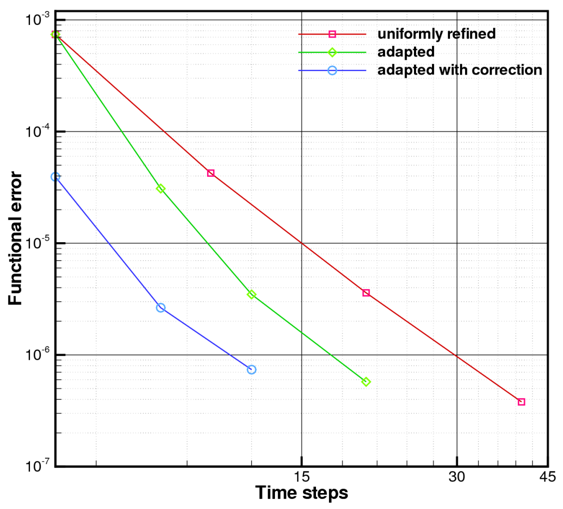
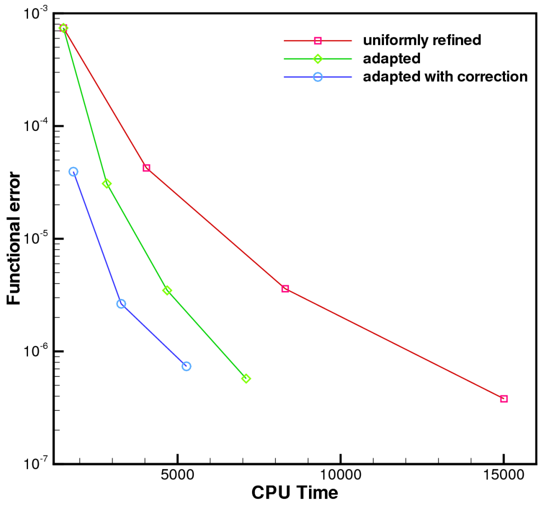


Figure 6.8: Functional error distribution in time for all adaptation cycles, in the test case of isentropic vortex convection.

in the isentropic vortex convection test case is also compared with that of the uniformly refined time-step simulations. The flow problem at each intermediate stage of the IRK4 scheme corresponding to the uniform time-step refinement approach is converged to machine precision of 10^{-16} to eliminate the presence of algebraic error. Therefore, the functional value obtained using the uniform time-step refinement is used to determine the optimal achievable error reduction possible for one temporal adaptation cycle. In terms of total number of time steps, as illustrated in Fig. 6.9(a), the adaptively refined time-step approach using the IRK4 scheme is capable of delivering accuracy levels equivalent to the uniformly refined time-step approach, while requiring fewer time steps. Moreover, the adjoint-based correction also provides an accurate prediction of the functional value evaluated on the next refinement level. In terms of CPU time, as shown in Fig. 6.9(b), the adaptive approach demonstrates superior efficiency over the uniform time-step refinement method, particular for the last two adaptation cycles, even though the cost for one adaptation cycle includes the cost of the unsteady multistage flow and adjoint solutions for the current adaptation level as well as the cost of its previous adaptation cycles. For example, to achieve a 10^{-6} functional error tolerance, the uniformly refined time-step approach incurs roughly twice the computational cost of the adaptive scheme. If the adjoint correction is taken into account, further improvement is achieved in terms of computational efficiency for a given accuracy level. Similarly to the previous case, the efficiency comparison may be optimistic since the uniform refinement cases are fully converged to eliminate algebraic errors, while the adaptive results are only partially converged, relying on the estimate of the algebraic error. The examination of the effectiveness of the error estimates provided by the adjoint-based error correction for the IRK4 adaptive scheme is demonstrated in Fig. 6.10. The functional value obtained from the adaptively refined time-step approach is close to the functional obtained by the corresponding uniformly refined time-step approach and moreover, the approximated functional, provided by the current functional value with the added correction term, is seen to be very accurate for predicting the functional value at the next finer level. The rapid convergence of the functional values with increasing number of time steps in this case is due to the fourth-order accuracy property of the underlying IRK4 time-stepping scheme.



(a) Error vs. Time steps



(b) Error vs. CPU Time

Figure 6.9: Comparison of the objective functional error reductions obtained by the uniform time-step refinement approach, the adaptive time-step refinement approach and the adaptive approach corrected with the adjoint-based correction for the test of isentropic vortex convection.

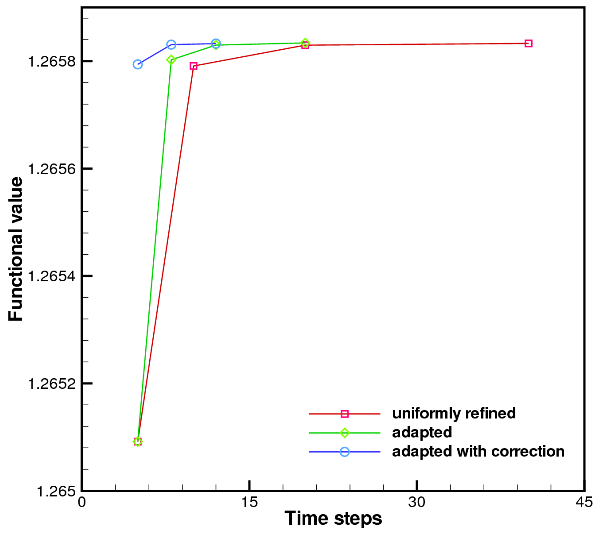


Figure 6.10: Convergence of the objective functional for the uniform time-step refinement approach, the adaptive time-step refinement approach and the adaptive approach corrected with the adjoint-based correction for the test of isentropic vortex convection.

Chapter 7

Conclusions

7.1 Summary and Conclusions

High-order accurate discontinuous Galerkin discretizations have been investigated in this work using a set of hierarchical basis functions. The optimal error convergence rate of $p + 1$ for a discretization order of p is achieved for the two-dimensional compressible Euler equations on unstructured triangular meshes. In order to make the high-order discretizations competitive, efficient solution methods have been developed, including an *hp*-Multigrid approach driven by element-Jacobi solver variants. The linearized element-Jacobi schemes deliver faster convergence histories than the non-linear counterparts, while requiring extra storage for the off-diagonal block components of the Jacobian matrix. The *hp*-Multigrid approach demonstrates both p- and h-independent convergence rates, and thus the efficiency benefits become more significant for finer meshes.

For time-dependent problems, an accurate and efficient solution requires a careful balance between both spatial and temporal errors. For problems with disparate length and time scales, implicit time-integration methods offer the most promising approach for realizing efficient and accurate solutions. In the interest of balancing spatial and temporal errors, the current work has investigated the use of high-order implicit Runge-Kutta schemes in addition to more traditional second-order temporal schemes, such as the second-order backwards differencing BDF2 scheme. Compared with the lower-order temporal schemes, the fourth-order implicit Runge-Kutta (IRK4) scheme requires more computational effort to obtain the unsteady solution at each implicit time step. However, the IRK4 scheme still outperforms the lower-order schemes due to its more favorable temporal accuracy. The second-order accurate Crank-Nicolson scheme performs better than the BDF2 scheme, although the lack of L-stability for this scheme indicates that it should be used with caution. Since the implicit systems of the unsteady problem are more diagonally dominant than those resulting from the steady-state problem, this work considers the use of a p-Multigrid approach alone for accelerating convergence of the non-linear problem arising from each implicit time step. The p-Multigrid approach demonstrates h-independent convergence rates while retaining slight dependence on time-step sizes. If the time-step size is chosen to be sufficiently large, the optimal convergence rate is expected to be asymptotically equivalent to that of the steady-state solver.

In addition to the investigation of the analysis problems, this work has developed a framework for implementing sensitivity analysis techniques to high-order discontinuous Galerkin methods, with particular emphasis on the use of a discrete adjoint method in the application areas of shape optimization and error estimation. In the context of shape optimization, the evaluation of sensitivity derivatives requires the linearization of the spatial residual with respect to modal solution coefficients and modal geometric mapping coefficients, since the set of discretized governing equations is solved in modal space, and the geometry is handled through iso-parametric mappings that may include curved elements.

Output-based error estimation is another essential application of the discrete adjoint method derived for shape sensitivities. A framework has been successfully established for providing an accurate error estimate in a simulation functional output, and for driving adaptive refinement strategies to improve functional accuracy with optimal computational cost. Although the error estimate obtained by this approach is used for predicting the functional error between two successively refined or enriched time-step levels rather than the true error measured in the continuous limit, this method has been found to be simple, yet efficient and sufficiently effective for adaptive refinement purposes. The *hp*-adaptive mesh refinement scheme is beneficial for shock-dominated problems, where a uniform $p = 0$ scheme is used to initialize the solution, and shock regions are consistently refined by an *h*-refinement scheme, and regions with smooth solutions may be enriched by a *p*-refinement scheme. The *hp*-adaptive scheme performs more efficiently than a purely *h*-refinement scheme for shock problems in that the *p*-enrichment scheme is capable of achieving optimal performance in smooth solution regions. Furthermore, in the *hp*-adaptive scheme, a $p = 0$ scheme is consistently employed in regions of shocks or discontinuities, thus avoiding the need to employ limiters.

The methodology of the adjoint-based spatial error estimation approach is also extended to temporal error estimation for a time-dependent functional of interest and to drive an adaptive time-step refinement strategy for the implicit temporal schemes implemented in this work. Although the formulation derived for adjoint-based temporal error estimation appears similar to its spatial counterpart, the procedure requires the solution of multistep or multistage implicit problems within a backward integration in time. The gain of the adaptive time-step refinement method over uniform time-step refinement depends upon the functional error distributions in the time domain or the relation of the target functional output to detailed time-dependent flow features. Specifically, if functional error is more tightly related to the flow features at a particular period of time than another period of time, the adaptive time-step refinement approach is capable of delivering substantial increases in efficiency.

In summary, the methodology developed in the current work not only provides an alternative path for achieving high accuracy for flows with wide range of scales by avoiding the use of excessive grid or time-step resolution, but also establishes a framework to further quantify the effect of discretization errors on the key output quantities of flow simulations and to improve the accuracy of simulations of interest by adaptive mesh or time-step refinement approaches. This approach can be regarded as an essential step to apply the computational fluid dynamics to practical problems.

7.2 Additional Contributions

High-order discontinuous Galekin discretizations are suitable for problems with smooth solutions. However, for problems in the presence of shocks or discontinuities, additional dissipative mechanisms [18, 47, 102] are required to stabilize high-order approximations and to prevent the generation of spurious oscillations. Preliminary investigation of a total variation based non-linear limiter [114] for limiting spurious oscillations when solving hyperbolic systems of conservation laws by high-order discontinuous Galerkin methods has been performed. The limited solution corresponds to the minimization of the total variation norm of the unlimited solution and can be obtained by solving the equivalent Euler-Lagrange (i.e. first-variation) equation using the same spatial DG scheme as the flow problem. The smoothness indicators used in the *hp*-adaptive scheme are also utilized for detecting non-smooth regions and thus limiting may be applied only in these regions to preserve high accuracy in smooth regions. In addition, the limiting procedure developed in this work is intended to be a very general approach, which can be used for any high-order discretization. Results show that the total variation based non-linear limiter is capable of limiting oscillations generated by high-order discontinuous Galerkin discretizations for strong hypersonic shocks, and also capable of capturing shocks with sub-cell resolution for two-dimensional transonic flows. A detailed description of this work is not included in this thesis since the topic is beyond the original scope of the thesis and will require considerable additional work to realize its full potential.

7.3 Future Work

There exists a number of directions for future work focusing on the improvement and extension of the methods developed in this thesis. A few ideas for future research are listed here.

1. **Improvement of the solver**

   Optimal performance of the *p*- or *hp*-Multigrid approach developed for steady-state or unsteady flow solvers relies on the accuracy of the solution at the lowest p-level (e.g. $p = 0$ level) and also requires avoiding excessive coarse level solution cost. Therefore, the optimal number of agglomerated coarser levels as well as smoothing passes in the element-Jacobi solvers remains to be further investigated.

2. **Dynamic mesh motion problems**

   The methodology formulated in the current work for time-dependent problems is restricted to the context of static mesh problems. Cases with relative body motion are very common in many engineering or aerodynamic applications. Future work is necessary to extend the methodology to dynamically deforming meshes by formulating high-order discontinuous Galerkin discretizations in Arbitrary Lagrangian Eulerian (ALE) form [115], while particular emphasis will be given on the construction of discretely conservative high-order discontinuous Galerkin methods in the presence of moving meshes and in the presence of both high-order temporal and spatial accuracy. The shape optimization technique developed for steady-state flow problems is thus readily extensible to unsteady flow problems with mesh motion, and further investigation of a better optimizer is vital for accelerating objective convergence of the design optimization procedure.

3. **Robustness of the *hp*-adaptive refinement strategy**

   The *hp*-adaptive refinement strategy enables the accurate solution of flows with strong shocks without the use of slope limiters, which can lead to loss of numerical convergence. However, successful implementation relies on the correct selection of smooth elements, since the implementation of the p-enrichment scheme must be avoided in regions of discontinuities in order to prevent the generation of unphysical solutions. In order to enhance the overall algorithm accuracy and robustness, further research is necessary to incorporate a high-order limiter (e.g. the aforementioned total variation based non-linear limiter) with the proposed *hp*-adaptive scheme for supersonic and high-speed flow problems.

Furthermore, the current implementation of the local smoothness indicators is seen to be effective for capturing discontinuous properties in the flow field, however, the parameters used in these indicators are still problem-dependent, e.g. the strength of the shocks, or distinction of shocks from stagnation points, etc. Research on defining a simple, effective and parameter-free indicator should be investigated in the future.

4. **Combination of spatial and temporal error estimation**

   The approach developed in the discrete adjoint-based error estimation procedure remains valid for errors arising from either a spatial discretization or a temporal discretization. However, the total error obtained from a numerical solution lies in both spatial and temporal discretizations. Without identifying the more important error source, mesh or time-step refinement strategies will fail to decrease functional errors in an efficient manner. Hence, of particular interest is the combination of both spatial and temporal error estimation and the quantification of errors from these sources. In this context, more effective adaptation strategies are expected to improve total error tolerances more efficiently.

5. **Extension to other sets of equations**

   Adaptive discontinuous Galerkin methods have been applied exclusively to the two-dimensional compressible Euler equations in the current work. However, the prescribed methods are intended to be general and in principle they are not restricted to these equations. Specifically, the adaptive method only requires a functional output with a well-posed adjoint problem while the adjoint problem is constructed as an exact dual to the primal problem. The techniques developed for error estimation and adaptation can be extended to other sets of equations, e.g. compressible Navier-Stokes equations. Future work should concentrate on implementing the Interior Penalty (IP) method [116, 117] for the spatial discretizations of diffusion terms and extending these methods to three-dimensional problems.

Bibliography

[1] J. Steger, R. Warming, Flux vector splitting for the inviscid gas dynamic equations with application to finite difference methods, Tech. Rep. NASA TM 78605 (1979).

[2] D. A. Anderson, J. C. Tannehill, R. H. Petcher, Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing, New York, NY, 1984.

[3] G. D. Smith, Numerical Solution of Partial Differential Equations, Clarendon Press, 1985.

[4] A. Jameson, W. Schmidt, E. Turkel, Numerical solution of the Euler equations by finite volume methods using Rung-Kutta time stepping schemes, AIAA Paper 1981–1259 (1981).

[5] J. T. Batina, Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA J. 28 (8) (1990) 1381–1288.

[6] H. K. Versteeg, W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, Longman Scientific and Technical, 1995.

[7] R. Eymard, T. Gallouet, R. Herbin, Finite volume methods, Handbook of Numerical Analysis (1997).

[8] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of 2D Euler equation, J. Comput. Phys. 138 (1997) 251–285.

[9] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method of the numerical solution of the compressible Navier-Stokes equation, J. Comput. Phys. 131 (1997) 267–279.

[10] F. Bassi, S. Rebay, Numerical evaluation of the two discontinuous Galerkin methods for the compressible Navier-Stokes equations, Int. J. Numer. Meth. Fluids. 40 (2002) 197–207.

[11] B.Berde, M. Borrel, Numerical experiments on the accuracy of a discontinuous Galerkin method for the Euler equations, Aerosp. Sci. Technol. 2 (5) (1998) 279–288.

[12] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin method for time-dependent convection-diffusion systems, SIAM J. Numer. Appl. Mech. Engrg. 35 (1998) 2440–2463.

[13] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, SIAM J. Sci. Comput. 16 (2001) 173–261.

[14] J. E. Flaherty, L. Krivodonova, J. Remacle, M. S. Shephard, Aspects of discontinuous Galerkin methods for hyperbolic conservation laws, Finite Elements Anal. Design 38 (10) (2002) 889–908.

[15] F. Brezzi, L. Marini, E. Suli, Discontinuous Galerkin methods for first-order hyperbolic problems, Math. Models Methods Appl. Sci. 14 (12) (2004) 1893–1903.

[16] C. R. Nastase, D. J. Mavriplis, High-order discontinuous Galerkin methods using an *hp*-multigrid approach, J. Comput. Phys. 213 (1) (2006) 330–357.

[17] K. J. Fidkowski, T. A. Oliver, J. Lu, D. Darmofal, *p*-multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, J. Comput. Phys. 207 (2005) 92–113.

[18] H. Luo, J. D. Baum, R. Lohner, A *p*-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, J. Comput. Phys. 211 (2) (2006) 767–783.

[19] P.-O. Persson, J. Peraire, An efficient low memory implicit DG algorithm for time dependent problems, AIAA Paper 2006-112 (Jan. 2006).

[20] L. Wang, D. J. Mavriplis, Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations, J. Comput. Phys. 225 (2) (2007) 1994–2015.

[21] A. Jameson, J. J. Alonso, J. J. Reuther, L. Martinelli, J. C. Vassberg, Aerodynamic shape optimization techniques based on control theory, AIAA Paper 1998–2538 (1998).

[22] A. Jameson, L. Martinelli, N. Pierce, Optimum aerodynamic design using the Navier-Stokes equations, Theor. Comput. Fluid Dyn. 10 (1) (1998) 213–237.

[23] A. Jameson, Aerodynamic design via control theory, J. Sci. Comput. 3 (3) (1988) 233–260.

[24] N. A. Pierce, M. Giles, Adjoint and defect error bounding and correction for functional estimates, Journal of Computational Physics 200 (2) (2004) 769–794.

[25] D. A. Venditti, D. L. Darmofal, Grid adaptation for functional outputs: Application to two-dimensional inviscid flows, J. Comput. Phys 176 (1) (2002) 40–69.

[26] T. Barth, Numerical methods and error estimation for conservation laws on structured and unstructured meshes, Lecture notes, von Karman Institute for Fluid Dynamics, Series: 2003-04, Brussels, Belgium (Mar. 2003).

[27] K. J. Fidkowski, D. Darmofal, A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations, J. Comput. Phys. 225 (2) (2007) 1653–1672.

[28] W. H. Reed, T. R. Hill, Triangular mesh methods for the neutron transport equation, Tech. Rep. LA-UR-73-479, Los Alamos Scientific Laboratory (1973).

[29] P. LeSaint, P. A. Raviart, On A Finite Element Method of Solving the Neutron Transport Equation, in Mathematical Aspects of Finite Elements in Partial Differential Equations, C. De Boor, Academic Press, 1974.

[30] C. Johnson, J. Pitkaranta, An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation, Math. Comp. 46 (173) (1986) 1–26.

[31] G. R. Richter, An optimal-order error estimate for the discontinuous Galerkin method, Math. Comp. 50 (181) (1988) 75–88.

[32] T. Peterson, A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation, SIAM J. Numer. Anal. 28 (1991) 133–140.

[33] G. Chavent, G. Salzano, A finite element method for the 1-D water flooding problem with gravity, J. Comput. Phys. 42 (3) (1982) 307–344.

[34] B. Cockburn, S. Lin, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework, Math. Comp. 52 (1989) 411–435.

[35] B. Cockburn, S. Lin, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems, J. Comput. Phys. 84 (1) (1989) 90–113.

[36] B. Cockburn, C.-W. Shu, The Runge-Kutta dinscontinuous Galerkin method for conservation laws V: Multidimensional systems, J. Comput. Phys. 141 (2) (1998) 199–224.

[37] B. Cockburn, G. Karniadakis, C. Shu, Discontinuous Galerkin methods. Theory, Computation and Applications, Lecture Notes in Computational Science and Engineering, Springer-Verlag, New York (2000).

[38] J. Oden, I. Babuska, C. Baumann, A discontinuous *hp* finite element method for diffusion problems, J. Comput. Phys. 146 (2) (1998) 491–519.

[39] B. Riviere, M. Wheeler, V. Girault, Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. part I, Comp. Geo. 3 (3/4) (1999) 337–360.

[40] P. Castillo, B. Cockburn, I. P. D. Schotzau, An *a priori* error analysis of the local discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 38 (2000) 1676–1706.

[41] B. Cockburn, G. Kanschat, I. Perugia, D. Schotzau, Superconvergence of the local discontinuous Galerkin method for elliptic problems on cartesian grids, SIAM J. Numer. Anal. 39 (2001) 264–285.

[42] C. R. Nastase, D. J. Mavriplis, Discontinuous Galerkin methods using an *hp*-multigrid solver for inviscid compressible flows on three-dimensional unstructured meshes, AIAA Paper 2006-0107 (Jan. 2006).

[43] E. F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer-Verlag, Berlin, Heidelberg, 1997.

[44] R. J. LeVeque, Wave propagation algorithms for multi-dimensional hyperbolic systems, J. Comput. Phys. 131 (2) (1997) 327–353.

[45] P. Batten, N. Clarke, C. Lambert, D. M. Causon, On the choice of wavespeeds for the HLLC Riemann solver, SIAM J. Sci. Comput. 18 (2) (1997) 1553–1570.

[46] E. F. Toro, M. Spruce, W. Spears, Restoration of the contact surface in the HLL-Riemann solver, Shock Waves 4 (1) (1994) 25–34.

[47] C.-W. Shu, Essentially Non-oscillatory and Weighted Essentially Non-oscillatory Schemes for Hyperbolic Conservation Laws, ICASE Report No. 97-65, NASA/CR-97-206253, 1997.

[48] C. R. Nastase, D. J. Mavriplis, A parallel *hp*-multigrid solver for three-dimensional discontinuous Galerkin discretizations of the Euler equations, AIAA Paper 2007-0512 (Jan. 2007).

[49] P. Houston, E. Suli, *hp*-adaptive discontinuous Galerkin finite element methods for first-order hyperbolic problems, SIAM J. Sci. Comput. 23 (4) (2001) 1226–1252.

[50] P. Houston, E. Suli, A note on the design of *hp*-adaptive finite element methods for elliptic partial differential equations, Comput. Methods Appl. Mech. Engrg. 194 (2-5) (2005) 229–243.

[51] P. Houston, D. Schotzau, T. P. Wihler, An *hp*-adaptive mixed discontinuous Galerkin FEM for nearly incompressible linear elasticity, Comput. Methods Appl. Mech. Engrg. 195 (25-28) (2006) 3224–3246.

[52] B. Cockburn, Devising discontinuous Galerkin methods for non-linear hyperbolic conservation laws, J. Comput. Phys. 128 (1-2) (2001) 187–204.

[53] E. J. Kubatko, C. Dawson, J. J. Westerink, Time step restrictions for Runge-Kutta discontinuous Galerkin methods on triangular grids, J. Comput. Phys. 227 (23) (2008) 9697–9710.

[54] V. Dolejsi, M. Feistauer, A semi-implicit discontinuous Galerkin finite element method for the numerical solution of inviscid compressible flow, J. Comput. Phys. 198 (2) (2004) 727–746.

[55] H. Bijl, M. H. Carpenter, V. N. Vatsa, C. A. Kennedy, Implicit time integration schemes for the unsteady compressible Navier-Stokes equations: Laminar flow, J. Comput. Phys. 179 (1) (2002) 313–329.

[56] P. Rasetarinera, M. Y. Hussaini, An efficient implicit discontinuous spectral Galerkin method, J. Comput. Phys. 172 (2) (2001) 718–738.

[57] H. Bijl, Iterative methods for unsteady flow computations using implicit Runge-Kutta integration schemes, AIAA Paper 2006-1278 (Jan. 2006).

[58] G. Arabatzis, P. Vavilis, I. Toulopoulos, J. A. Ekaterinaris, Implicit high-order time marching schemes for the linearized Euler equations, AIAA Paper 2006-3548 (June 2006).

[59] O. Pironneau, Optimal Shape Design for Elliptic Systems, Springer-Verlag, New York, 1984.

[60] J. C. Newman III, A. C. Taylor III, G. W. Burgreen, An unstructured grid approach to sensitivity analysis and shape optimization using the Euler equations, AIAA Paper 1995-1646 (Jan. 1995).

[61] J. C. Newman III, A. C. Taylor III, G. W. Burgreen, Three-dimensional aerodynamic shape sensitivity analysis and design optimization using the Euler equations, AIAA Paper 1996-2464 (Jan. 1996).

[62] J. Elliott, J. Peraire, Aerodynamic optimization on unstructured meshes with viscous effects, AIAA Paper 1996–1941 (Jan. 1996).

[63] J. Elliott, J. Peraire, Practical 3D aerodynamic design and optimization using unstructured meshes, AIAA J 35 (1997) 35–9.

[64] W. Anderson, V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, AIAA Paper 1997–0643 (Jan. 1997).

[65] K. Ghayour, O. Baysal, Unsteady aerodynamics and shape optimization using modified transonic small disturbance equation, AIAA Paper 2001-2612 (June 2001).

[66] K. Yee, Y. Kim, D. Lee, Aerodynamic shape optimization of rotor airfoils undergoing unsteady motion, AIAA Paper 1999–3107 (Jan. 1999).

[67] K. Mani, D. J. Mavriplis, An unsteady discrete adjoint formulation for two-dimensional flow problems with deforming meshes, AIAA Paper 2007-0060 (Jan. 2007).

[68] M. P. Rumpfkeil, D. W. Zingg, Unsteady optimization using a discrete adjoint approach applied to aeroacoustic shape design, AIAA Paper 2008-0018 (Jan. 2008).

[69] D. Estep, A posteriori error bounds and global error control for approximation of ordinary differential equations, SIAM Journal of Numerical Analysis 32 (1995) 1–48.

[70] R. Becker, R. Rannacher, An optimal control approach to *a posteriori* error estimation in finite element methods, Vol. 10, Acta Numer., 2002.

[71] P. Houston, R. Hartmann, E. Suli, A posteriori error analysis for stabilised finite element approximations of transport problems, Comput. Methods Appl. Mech. Engrg. 190 (11-12) (2000) 1483–1508.

[72] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, J. Comput. Phys. 183 (2) (2002) 508–532.

[73] M. Nemec, M. J. Aftosmis, M. Wintzer, Adjoint-based adaptive mesh refinement for complex geometries, AIAA Paper 2008-725 (Jan. 2008).

[74] K. Mani, D. J. Mavriplis, Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems, AIAA Paper 2007-3944 (June 2007).

[75] K. Mani, D. J. Mavriplis, Error estimation and adaptation for functional outputs in time-dependent flow problems, AIAA Paper 2009-1495 (Jan. 2009).

[76] J. C.-C. Lu, An *a posteriori* error control framework for adaptive precision optimization using discontinuous Galerkin finite element method, Doctoral Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology (June 2005).

[77] P. Solin, K. Segeth, I. Dolezel, High-Order Finite Element Methods, Studies in Advanced Mathematics, Chapman and Hall, 2003.

[78] D. A. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, Int. J. Numer. Meth. Engng. 21 (6) (1985) 1129–1148.

[79] D. A. Dunavant, Economical symmetrical quadrature rules for complete polynomials over a square domain, Int. J. Numer. Meth. Engng. 21 (10) (1985) 1777–1784.

[80] K. J. Fidkowski, D. L. Darmofal, Development of a higher-order solver for aerodynamic applications, AIAA Paper 2004–0436 (Jan. 2004).

[81] B. Helenbrook, D. J. Mavriplis, H. Atkins, Analysis of "p"-multigrid for continuous and discontinuous finite element discretizations, AIAA Paper 2003-3989 (June 2003).

[82] D. J. Mavriplis, Multigrid techniques for unstructured meshes, Tech. Rep. ICASE 95-27, Institute for Computer Applications in Science and Engineering (ICASE) (1995).

[83] D. J. Mavriplis, An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers, Tech. Rep. ICASE Report 2001-12, Institute for Computer Applications in Science and Engineering (ICASE) (May 2001).

[84] D. A. Anderson, J. C. Tannehill, R. H. Pletcher, Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing Corp., 1984.

[85] C. R. Nastase, D. J. Mavriplis, High-order discontinuous Galerkin methods using a spectral multigrid approach, AIAA Paper 2005-1268 (Jan. 2005).

[86] E. J. Nielsen, W. K. Anderson, Recent improvements in aerodynamic design optimization on unstructured meshes, AIAA J. 40 (6) (2002) 1155–1163.

[87] A. Jameson, Aerodynamic shape optimization using the adjoint method, VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute For Fluid Dynamics, Belgium (February 2003).

[88] O. Soto, R. Lohner, C. Yang, An adjoint-based design methodology for CFD problems, Int. J. Numer. Method. H. 14 (6) (2003) 734–759.

[89] D. J. Mavriplis, A discrete adjoint-based approach for optimization problems on three-dimensional unstructured meshes, AIAA Paper 2006-0050 (Jan. 2006).

[90] R. M. Hicks, P. A. Henne, Wing design by numerical optimization, J. Aircraft 15 (7) (1978) 407–412.

[91] V. Venkatakrishnan, D. J. Mavriplis, Implicit method for the computation of unsteady flows on unstructured grids, J. Comput. Phys. 127 (2) (1996) 380–397.

[92] Z. Yang, D. J. Mavriplis, Unstructured dynamic meshes with higher-order time integration schemes for the unsteady Navier-Stokes equations, AIAA Paper 2005-1222 (Jan. 2005).

[93] J. Nocedal, S. J. Wright, Numerical Optimization, Springer Series in Operations Research New York: Springer-Verlag, 1999.

[94] K. Mani, D. J. Mavriplis, Linearization of the coupled unsteady fluid-structure equations: Application to flutter control, AIAA Paper 2008-6242 (August 2008).

[95] M. B. Giles, E. Suli, Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, Acta Numer. 11 (2002) 145–236.

[96] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows, J. Comput. Phys 187 (1) (2003) 22–46.

[97] M. A. Park, D. L. Darmofal, Output-adaptive tetrahedral cut-cell validation for sonic boom prediction, AIAA Paper 2008-6595 (Jan. 2008).

[98] K. J. Fidkowski, D. L. Darmofal, Output-based error estimation and mesh adaptation in computational fluid dynamics: Overview and recent results, AIAA Paper 2009-1303 (Jan 2009).

[99] P.-O. Persson, J. Peraire, Sub-cell shock capturing for discontinuous Galerkin methods, AIAA Paper 2006-112 (Jan. 2006).

[100] G. E. Barter, D. L. Darmofal, Shock capturing with higher-order, PDE-based articial viscosity, AIAA Paper 2007-3823 (June 2007).

[101] C. Mavriplis, Adaptive mesh strategies for the spectral element method, Comput. Methods. Appl. Mech. Engrg. 116 (1994) 77–86.

[102] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, J. Flaherty, Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws, Appl. Numer. Math. 48 (3) (2004) 323–338.

[103] L. Wang, D. J. Mavriplis, Adjoint-based *h-p* adaptive discontinuous Galerkin methods for the compressible Euler equations, AIAA Paper 2009-0952 (Jan. 2009).

[104] D. J. Mavriplis, Unstructured grid techniques, Annu. Rev. Fluid. Mech. 29 (1) (1997) 473–514.

[105] S. McCormick, J. Thomas, The fast adaptive composite grid (FAC) method for elliptic equations, Mathematics of Computations 46 (174) (1986) 439–456.

[106] I. Lomtev, C. B. Quillen, G. E. Karniadakis, Spectral/hp methods for viscous compressible flows on unstructured 2D meshes, J. Comput. Phys. 144 (2) (1998) 325–357.

[107] P.-O. Persson, Scalable parallel Newton-Krylov solvers for discontinuous Galerkin discretizations, AIAA Paper 2009-606 (Jan. 2009).

[108] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Springer-Verlag, Berlin, 1996.

[109] C. A. Kennedy, M. H. Carpenter, Additive Runge-Kutta schemes for convection diffusion reaction equations, Appl. Numer. Math. 44 (1).

[110] G. Jothiprasad, D. J. Mavriplis, D. A. Caughey, Higher-order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes, J. Comput. Phys. 191 (2) (2003) 542–566.

[111] F. Davoudzadeh, H. McDonald, B. E. Thompson, Accuracy evaluation of unsteady CFD numerical schemes by vortex preservation, Computers and Fluids 24 (1995) 883–895.

[112] E. Garnier, P. Sagaut, M. Deville, A class of explicit ENO filters with application to unsteady flows, J. Comput. Phys. 170 (1) (2001) 184–204.

[113] H. C. Yee, N. D. Sandham, M. J. Djomehri, Low-dissipative high-order shock-capturing methods using characteristic-based filters, J. Comput. Phys. 150 (1) (1999) 199–238.

[114] L. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Phys. D. 60 (1-4) (1992) 259–268.

[115] D. J. Mavriplis, C. R. Nastase, On the geometric conservation law for high-order discontinuous Galerkin discretizations on dynamically deforming meshes, AIAA Paper 2008-778 (Jan. 2008).

[116] K. Shahbazi, An explicit expression for the penalty parameter of the interior penalty method, J. Comput. Phys. 205 (2).

[117] K. Shahbazi, D. Mavriplis, N. K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the Navier-Stokes equations, to appear in J. Comput. Phys. (2009).