A Robust *hp*-Adaptation Method for Discontinuous Galerkin Discretizations Applied to Aerodynamic Flows

by

Marco Antonio de Barros Ceze

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2013

For my amazing family

ACKNOWLEDGEMENTS

First and foremost, I express my gratitude to my advisor, Professor Krzysztof Fidkowski. His guidance during the past four years was instrumental to the success of this thesis work. He was also very skilled in keeping me on track while I tried different ideas and in motivating me during the difficult times. It was an honor being part of his research group.

I would like to thank my committee members, Professor Bram van Leer, Professor David Zingg and Professor Eric Johnsen. Their questions and comments during our meetings helped shape this work. I thank Professor Bram van Leer for recruiting me for graduate school and for our discussions in the early stages of my research. I thank Professor David Zingg for the challenging questions during our meetings and for our discussions about the physicality constrained solver during my time at NASA Ames. I also thank Professor Eric Johnsen for agreeing to be a cognate committee member.

I am also very grateful for the support given by Michael Aftosmis, Dr. Marian Nemec and Professor Thomas Pulliam from NASA Ames. They were great hosts during my time there and the computational resources provided by NASA were essential for the development of this work. At Michigan, Professor Joaquim Martins taught us an excellent optimization course that sparked the ideas used in the constrained solver and in the line-search.

My friends deserve recognition for making my life very enjoyable. They are also saints for putting up with my annoying habits. Matt Holzel and Bojana Drincic introduced me to the fun part of graduate school. Then Tony D'Amato and Erin

Farbar came along and fueled my silly sense of humor. Ashley Verhoff was always there to listen to my frustrations and to review my texts. Tim Eymann played a central role in keeping me motivated and giving me interesting perspectives on work and life. Steve Kast and Johann Dahm made me look lazy in the research group because of their hard work. Asad Ali and Khaled Aljanaideh were awesome officemates. Darshan Karwat always showed me the good side of everything. Paul Giuliano made everybody laugh with his impressions. Thanks to John Hwang for our discussions about coupled adjoints. To those that I may have forgotten, please do not dislike me!

I thank my family for the endless support. My parents, Carlos and Aparecida, provided me with a good education and I only had the easy task of loving them. My siblings, Luis and Ana, were always there for me even when I was unbearably pessimistic. And finally, I would like to thank Anna Luisa for making me happy and helping me with keeping my mind off work when I needed rest.

Funding for this work was provided by the Regents of the University of Michigan and by the U.S. Air Force Office of Scientific Research under contract FA9550-10-C-0040.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

A Robust *hp*-Adaptation Method for Discontinuous Galerkin Discretizations Applied to Aerodynamic Flows

by

Marco Antonio de Barros Ceze

Chair: Krzysztof J. Fidkowski

Quantitatively accurate results from realistic Computational Fluid Dynamics (CFD) simulations are often accompanied by high computational expense. Higher-order methods are good candidates for providing accurate solutions at reduced cost. However, these methods are still not robust for industrial applications.

This thesis presents a solution advancement method that improves robustness of discontinuous Galerkin (DG) discretizations in the iteration to steady-state. The method includes physical realizability constraints in the solution path and provides the solver with the ability of circumventing non-physical regions of the solution space that can occur during the solution transient.

Affordable accurate solutions for challenging problems are obtained via output-based *hp*-adaptation. The adaptation method proposed in this thesis directly targets output error by locally choosing between subdividing an element or raising the approximation order. The decision is made by finding the refinement option that maximizes a merit function that involves output sensitivity and computational cost. Results in two and three dimensions show savings of up to an order of magnitude in terms of number of degrees of freedom and at least a factor of two in terms of computational time when compared to uniform refinement.

CHAPTER 1

Introduction

1.1 Context and Motivation

The presence of Computational Fluid Dynamics (CFD) tools in the engineering environment has steadily increased in the past few decades. With the evolution of algorithms and the substantial enhancement of computational power, CFD tools now provide the ability to explore new configurations and test flow conditions that may be otherwise difficult to produce experimentally. As the range of applications becomes wider and the number of simulations increases, requirements of high-accuracy and robustness present challenges for the CFD development community [12].

Analyses using Breguet's range equation show that variations in aerodynamic drag of a large transport airplane as small as 1% affect its payload by $\sim 7\%$ [77, 19], which clearly impacts the profitability of its operation as airlines rarely achieve profit margins superior to 8% (Figure 1.1). Therefore, accurate prediction of an aircraft's performance in the design stage is important for its market success.

Aerodynamic flow over an aircraft, as for many other cases, exhibits features with unknown spatial distribution, and the range of the features' length scales can easily span six orders of magnitude. Furthermore, flows can exhibit singularities that pose additional challenges for output prediction. The trivial solution to these problems is to globally refine the mesh. However, this strategy is generally inefficient due to

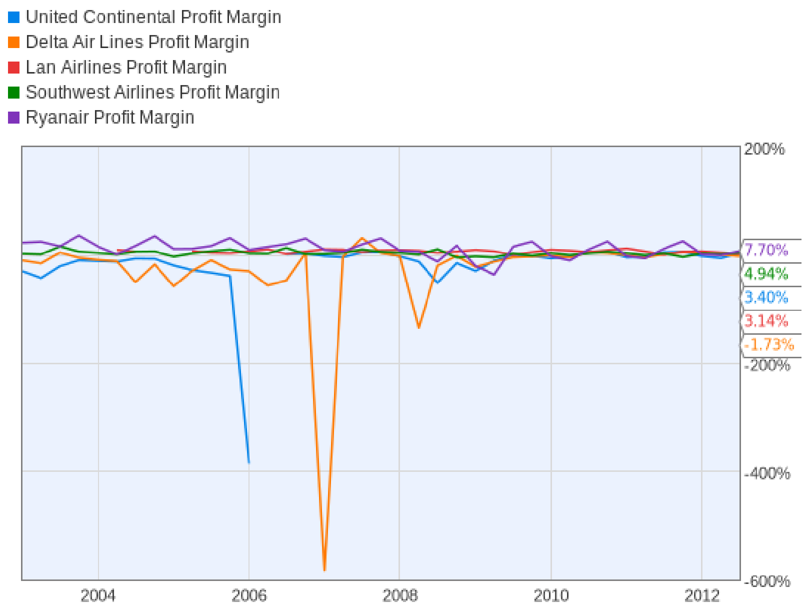


Figure 1.1: Profit margin over the past 10 years for various airlines. Source: `http://www.ycharts.com`

very large grid sizes required to appropriately resolve the relevant flow features and to accurately predict the outputs of interest.

The American Institute of Aeronautics and Astronautics (AIAA) organizes drag and lift prediction workshops (DPW and HLPW) with the purpose of assessing the capability of state-of-the-art computational methods and turbulence modeling for predicting forces and moments on relevant geometries in the aeronautical industry. In these workshops, starting meshes are generated based on industry's best practices and mesh independence is generally sought via uniform refinement studies. Nevertheless, the spread of results is significant [49, 48, 78, 79]. Figure 1.2 shows the evolution of the finest-mesh-sizes and the spread[1] of results computed with those meshes for the past four workshops. After the first DPW, when the best-practices were formulated, there was a large reduction in the spread of computed drag values. However, over the course of the next three workshops the spread was reduced by a factor of ~ 10 while the finest-mesh size increased by a factor of ~ 100. It is worth emphasizing that the

[1]Difference between maximum and minimum computed values.

values shown in Figure 1.2 for the fourth workshop correspond to the data submitted by a smaller number of participants that had enough computational resources to use the extra-fine mesh level. Therefore, the computational cost incurred in uniform refinement strategies clearly makes those simulations far from routine.

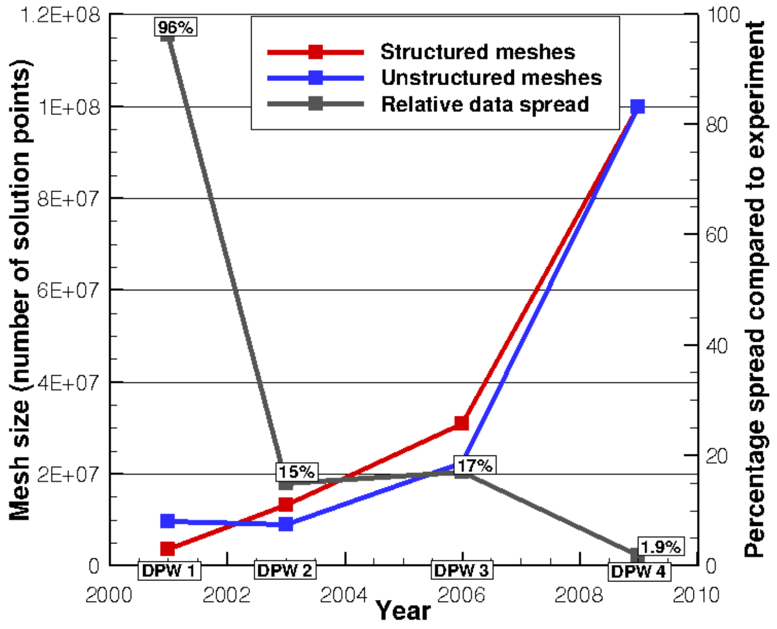


Figure 1.2: Spread of drag values computed with the finest, provided mesh for the AIAA drag prediction workshops – DPW 1: DLR-F4 wing-body geometry; DPW 2 and 3: DLR-F6 wing-body geometry; DPW 4: NASA CRM wing-body-horizontal-tail geometry.

Mesh adaptation methods present an attractive alternative for robust and accurate calculations on affordable grid sizes. These methods rely on the definition of an adaptive indicator which localizes the regions of the computational domain that need mesh modification through refinement, coarsening, or node movement. An effective indicator is obtained through output-based error estimation methods, which have already been demonstrated for many complex problems, including those in aerospace applications [22]. The goal of these methods is to provide confidence measures in the form of error bars for scalar outputs of engineering interest. In addition, one can use the error contributions of different elements or volumes of the computational mesh as an adaptive indicator that specifically targets errors in the outputs of interest [82, 61, 33, 21, 57, 14].

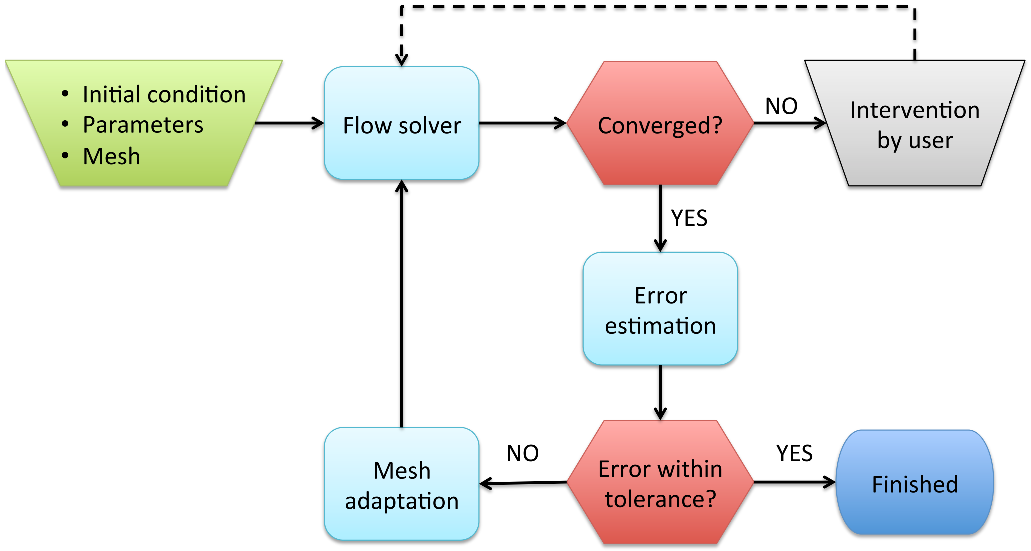


Figure 1.3: General solution process with error estimation and mesh adaptation.

Despite the effectiveness of output-based error estimation and mesh adaptation methods, they are not typically used in CFD practice. When they are used, they usually follow the process illustrated in Figure 1.3. For efficiency, the output-based adaptation processes should start with the coarsest mesh that still enables a solution [12] and should improve spatial resolution in areas of the computational domain that are relevant to the particular calculation being performed.

General gridding guidelines for complex problems are not widely available, and this presents the question of what is a *fine enough* mesh that allows the flow solver to converge. In the context of mesh adaptation, this solution does not have to be accurate, and most likely will not be on the initial meshes, but it should allow the adaptive algorithm to proceed to meshes that do allow accurate predictions of engineering quantities. When the flow solver fails to converge, a typical approach is for the user to modify the solver parameters or the mesh in an attempt to help the solver. These modifications are generally heuristic and difficult to automate, and they can be avoided by improving the robustness of the flow solver and by identifying areas of the mesh that are hampering convergence.

Solvers typically fail to converge due to under-resolved flow features such as shocks, boundary layers, and wakes. These features often cause oscillations in the numerical solution whose amplitudes can lead to non-physical values [12] and prevent the solver from providing a solution.

1.2 Objective

The objective of this work is to improve the robustness with which computational fluid dynamics methods solve problems of engineering relevance. This involves development of effective mesh adaptation methods and improvements in nonlinear solvers.

1.2.1 Output-Based Error Estimation and Mesh Adaptation

Output-based error estimation techniques identify all areas of the domain that are important for the accurate prediction of an output. The resulting estimates properly account for error propagation effects that are inherent to hyperbolic problems, and they can be used to ascribe confidence levels to outputs or to drive adaptation. A key component of output error estimation is the solution of an adjoint equation for the output of interest.

Intuitively, an adjoint (or dual) solution represents the sensitivity of an output with respect to residual perturbations. The output error is estimated via an inner product between the adjoint and residual perturbations produced by injecting the approximate flow solution into an enriched approximation space that serves as a surrogate for the continuum. This procedure is referred to as the Dual-Weighted Residual (DWR) method. For discretization methods that subdivide the domain into smaller elements or volumes, the elemental contributions to that error estimate can be treated as adaptive indicators. That is, large contributions to the error estimate indicate under-resolved areas of the domain that affect the accurate prediction of the output.

The adaptive indicators can be used to define a metric field that represents the spatial resolution desired for the modified mesh [22]. This field is then used to re-mesh the entire domain. Another approach is to locally modify the mesh. These modifications can be classified into two types: element size (h modification) or local polynomial approximation order (p modification). Within h-adaptation, elements can be subdivided/agglomerated, thus modifying the total number of degrees of freedom (DOF), or the nodes can move to modify the element size distribution at a constant number of DOF's.

The choice between resizing the elements or locally changing the scheme's discretization order is not trivial and has been the subject of much previous research [8, 37, 65, 39, 28, 10]. Bey [8] uses the error equidistribution principle to first subdivide elements and then increase the polynomial order where the solution is deemed smooth. Conversely, Heuveline and Rannacher [37] propose a process that prioritizes p-refinement and only subdivides an element when the previous step leads to an increase in the elemental error indicator. Houston and Süli [39] present two methods for assessing the local smoothness of the solution using Legendre series expansions and for obtaining estimates of the local regularity of the underlying analytical solution. In that same article, they also provide an overview of different strategies for deciding the refinement type. Burgess and Mavriplis [10] use a solution-jump indicator to decide between h and p refinements. Following a different approach, Rachowicz *et al.* [65]

choose h or p refinement based on an estimated lowest interpolation error.

Aerodynamic flow features also exhibit anisotropy, that is, variations of disparate magnitudes in different directions. The dominant method for detecting anisotropy has relied on estimates of the directional interpolation error of a representative scalar, such as the Mach number [63, 52]. When used alone, this technique reduces to equidistributing the interpolation error of the chosen scalar over the computational domain, with the absolute level of interpolation error prescribed by the user [13, 32]. Alternately, this technique can be combined with output-based error estimation by using the output adaptive indicator to set the element size and the directional interpolation error to set the element stretching [80, 82]. The same idea can be extended to high-order discretizations [20, 18], although the measurement of directional interpolation error becomes more tedious. A more fundamental problem with this approach in the context of output-based adaptation is the assumption that mesh anisotropy should be governed by the directional interpolation error of one scalar quantity. This assumption is heuristic because it does not take into account the process by which interpolation errors create residuals that affect the output of interest. As a result, recent research has turned to adaptation algorithms that directly target the output error.

Formaggia *et al.* [25, 24, 23, 53] combine Hessian-based interpolation error estimates with output-based *a posteriori* error analysis to arrive at an output-based error indicator that explicitly includes the anisotropy of each element. Park [62] introduces an algorithm that directly targets the output error through local mesh operators of element swapping, node movement, element collapse, and element splitting. Using the output error indicator to rank elements and nodes, these operations are performed in sequence and automatically result in mesh anisotropy. Schneider and Jimack [71] calculate the sensitivities of the output error estimate with respect to node positions and formulate an optimization problem to reduce the output error estimate by redistributing the nodes. They then combine this node repositioning with isotropic local mesh refinement sequentially in a hybrid optimization/adaptation algorithm. Yano [85] uses the duality between simplex meshes and their implied metric field to formulate a mesh optimization problem that minimizes output error while maintaining the number of degrees of freedom. More degrees of freedom are introduced if the output error estimate does not satisfy the user-specified tolerance.

This thesis presents a method for concurrent mesh and polynomial-order adaptation with the objective of direct minimization of output error. The method uses a selection process for choosing the optimal refinement option from a discrete set of choices that includes directional h-refinement and p-increment. No attempt is made, however, to measure the solution anisotropy or smoothness directly or to incorporate it into the scheme. Rather, mesh anisotropy and approximation order distribution arise naturally from the optimization of a merit function that incorporates both output sensitivity and measures of solution cost.

1.2.2 Robust High-Order Implicit Methods

The dominant method for solving flow problems in the aeronautical industry is the finite-volume method (FVM). This method is generally limited to second-order-accurate variants. That is, if the underlying exact solution is smooth, the discretization error is expected to decrease quadratically as the mesh is uniformly refined. However, for many problems of practical interest, accuracy requirements are increasingly more stringent and second-order accuracy may not suffice [12].

In finite-volume and finite-difference schemes, higher orders of accuracy are generally achieved by extending the approximation stencil. This extension does not come free as it interferes with parallelization, hinders the treatment of boundary conditions, and, more importantly, requires time-integration methods with stronger stability properties.

Alternatively, finite-element methods (FEM) can achieve higher orders of accuracy with a fixed, element-wise compact stencil by approximating the flow field using polynomials with local support. The discrete system is coupled by either enforcing solution continuity across element boundaries or by defining unique numerical fluxes between elements. The latter choice yields the discontinuous Galerkin method (DG), which is specifically suited for aerodynamics as it provides stability for convection-dominated problems. Yet, DG methods still present robustness challenges that prevent them from being widely used to solve industry problems. In fact, one of the findings of a recent workshop[2] was that high-order methods are still not as robust as second-order finite-volume methods for problems with turbulence.

An important ingredient for the robustness of second-order finite-volume methods is the advent of limiters. Cockburn *et al.* [16] extended that idea, originally proposed by van Leer [75], to the discontinuous Galerkin finite-element discretization with Runge-Kutta time stepping. This method is know as RKDG and it preserves monotonicity of mean values. More recently, Kuzmin[46] proposed a form of RKDG that uses a hierarchical derivative limiting approach. This is convenient with Taylor basis functions since the limiter acts directly on the degrees of freedom by a process that is equivalent to p-coarsening (lowering the polynomial order) the cells where the solution is not monotone. Unfortunately, limiting methods are not mature yet in higher-order implicit DG formulations.

Schemes with limiters are robust in time-accurate calculations because they enforce monotonicity restrictions on the discrete solution. However, the steady solution of the discrete residual may not be monotonic [83]. Therefore, enforcing monotonicity can prevent the solver from converging. Conversely, without limiters, numerical oscillations can lead to violations of physicality constraints and also prevent convergence.

Alternatively, artificial dissipation is often used as an attempt to smooth out os-

[2]The 1st International Workshop on High-Order CFD Methods was part of the 50th AIAA Aerospace Sciences Meeting held in January 2012.

cillations. Originally proposed by Von Neumann and Richtmyer [58] for capturing shocks and explored by many others, artificial dissipation methods generally use discontinuity sensors that control even-order derivative terms that damp wave-lengths of the order of the local mesh resolution. This is achieved by augmenting the residual expression with dissipative terms that are negligible in smooth regions of the flow and are triggered at regions with certain features, such as strong gradients or lack of smoothness. Because of the residual modification, these methods do not prevent Newton-based methods from converging to steady-state. The challenge, however, is to determine the level of artificial dissipation that is adequate for robustness but not too large to destroy solution accuracy. In the finite volume community, this balance was found in a seminal paper by Jameson *et al.* [40]. In high-order finite elements discretizations, robust artificial dissipation methods are still being pursued for complex problems [4, 64].

Full nonlinear convergence of the residual to machine precision levels is not strictly necessary for most flow simulations in the design environment. However, in some practical cases of the aeronautical industry, quantities such as drag and moment vary significantly despite the residual being reduced by several orders of magnitude[12]. Additionally, the theory of error estimation makes use of Galerkin orthogonality which is only theoretically valid if the discrete residual is zero. Therefore, the development of solution advancement methods that robustly drive the residual to *zero* (up to machine precision) is an important step in increasing the prevalence of high-order methods in the aeronautical industry.

This thesis presents a method for directly incorporating physicality constraints in the iterative solution path. This improves robustness by providing the solver with the ability of circumventing non-physical regions of the solution space that can occur during the iterations towards *zero residual*.

1.3 Scope

Robustness of high-order solvers and output-based error estimation/mesh adaptation are broad topics of research. This work restricts these topics to the following:

- **Discontinuous Galerkin discretization:** DG methods provide the ability of locally changing the discretization order while maintaining a compact stencil and for straightforward treatment of hanging-nodes. Additionally, output-based error estimation fits naturally into the scheme due to the Galerkin orthogonality property.

- **Steady problems:** the majority of the computational analyses in the aeronautical industry are of steady, compressible, turbulent flow. This work focuses on improving the robustness of the nonlinear solution technique in achieving spatial residual convergence for these types of flows.

- **Quadrilateral and hexahedral meshes:** such meshes naturally yield high-quality anisotropic elements and lend themselves to hanging-node refinement. In addition, many boundary-conforming quadrilateral or hexahedral meshes and associated meshing programs exist in the structured CFD community.

1.4 Thesis Overview

This thesis addresses the development of an optimization-based *hp*-adaptation framework that directly targets output error. The guiding philosophy for this work was to successfully apply the methods developed here to industry-relevant problems. The specific contributions of this thesis are as follows:

- A time integration technique that accounts for realizability constraints in the solution path.

- A line-search update method that extends the sphere of convergence of Newton-based solvers.

- An algorithm that reduces the heuristics on deciding between h and p refinements.

- A mesh partitioning method that effectively accounts for inhomogeneous computational cost that arises from *hp*-adaptation.

Chapter 2 describes the discontinuous Galerkin method used for the spatial discretization. The constrained time-integration and the solution update methods are described in Chapter 3. Chapter 4 presents the output error estimation and the *hp*-adaptation methods followed by the description, in Chapter 5, of certain implementation aspects of this work. No specific chapter is dedicated to results. Instead, we present them along with discussions in the pertinent chapters. Finally, Chapter 6 presents conclusions and ideas for future work.

CHAPTER 2

Flow Equations and Discretization

The solver robustness improvements presented in this thesis are applied to a discontinuous Galerkin spatial discretization of the Navier-Stokes equations. These equations are augmented with the Spalart-Allmaras turbulence model for simulating Reynolds-averaged turbulent flow. This chapter reviews the flow equations and presents the discontinuous Galerkin (DG) discretization along with the shock-capturing scheme used. The chapter ends with a procedure for scaling the discrete equations resulting from the DG discretization.

2.1 Compressible Navier-Stokes Equations

The Navier-Stokes equations include convective and diffusive terms. In compact, conservative form, they can be written as

$$\partial_t \mathrm{u}_s + \partial_i \mathcal{C}_{is}(\mathbf{u}) - \partial_i \mathcal{D}_{is}(\mathbf{u}) = 0, \tag{2.1}$$

where $i \in [1, .., \dim]$ indexes the spatial dimensions and s indexes the equations of conservation of mass, momentum, and energy. Accordingly, the state vector is denoted by $\mathbf{u} = [\rho, \rho v_i, \rho E]^T$, where ρ is the density, v_i are the spatial components of the velocity and E is the specific total energy. In Eqn. 2.1, the convective and diffusive fluxes are denoted by $\mathcal{C}$ and $\mathcal{D}$ respectively. These terms correspond to the following conservation statements:

- Conservation of mass, $s = 1$:

$$\mathcal{C}_{i1} = \rho v_i, \quad \mathcal{D}_{i1} = 0. \tag{2.2}$$

- Conservation of momentum, $s = 2 \rightarrow$dim+1:

$$\mathcal{C}_{is} = \rho v_{s-1} v_i + \delta_{i\ (s-1)} p, \quad \mathcal{D}_{is} = \tau_{i\ (s-1)}. \tag{2.3}$$

- Conservation of energy, $s = \text{dim} + 2$:

$$\mathcal{C}_{is} = \rho v_i H, \quad \mathcal{D}_{is} = \kappa_T \partial_i T + v_j \tau_{ij}. \tag{2.4}$$

Note, δ_{ij} is the Kronecker delta symbol. For inviscid calculations, the physical diffusion term $\mathcal{D}$ is not included in the equation set. In viscous calculations, we consider Newtonian fluids for which the viscous stress tensor is given by

$$\tau_{ij} = \mu(\partial_i v_j + \partial_j v_i) + \lambda \delta_{ij} \partial_k v_k, \tag{2.5}$$

where μ and λ are the dynamic and bulk viscosities, respectively. Note that the diffusive flux is non-linear with respect to the state, but linear with respect to the gradient of the state.

We assume a calorically and thermally perfect gas to close the system in Eqn. 2.1. This allows us to relate the pressure, p, temperature, T and specific total enthalpy,

H, to the conserved variables as follows:

$$\begin{aligned} p &= (\gamma - 1)\left(\rho E - \rho \frac{v_i v_i}{2}\right), && (2.6)\\ H &= E + \frac{p}{\rho}, && (2.7)\\ T &= \frac{p}{R\rho}. && (2.8)\end{aligned}$$

For all results presented in this thesis, the fluid is air and its physical properties are:

$$\begin{aligned} \text{Dynamic viscosity: } \mu &= \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}}\right)^{1.5} \left(\frac{T_{\text{ref}} - T_{\text{s}}}{T + T_{\text{s}}}\right), \\ & \quad \text{(Sutherland's law: } T_{\text{ref}} = 288.15K,\ T_{\text{s}} = 110K) \\ \text{Bulk viscosity coefficient: } \lambda &= -\frac{2}{3}\mu, \\ \text{Thermal conductivity: } \kappa_T &= \frac{\gamma \mu R}{(\gamma - 1) Pr}, \\ \text{Specific-heat ratio: } \gamma &= 1.4, \\ \text{Prandtl number: } Pr &= 0.71, \end{aligned}$$

where R is the gas constant.

2.2 Spalart-Allmaras Turbulence Model

In this work we use the Spalart-Allmaras (SA) turbulence model [73]. The model takes the form of a partial-differential-equation (PDE) for the quantity $\tilde{\nu}$ which represents a turbulent kinematic viscosity. This quantity is non-dimensionalized by to the physical kinematic viscosity to yield $\chi = \tilde{\nu}/\nu$.

The turbulent flows presented in this thesis are assumed to be fully-turbulent. Hence, we do not include turbulent transition approximation terms in the model. Oliver and Allmaras [59] proposed modifications to the original SA model that ensure stability of the model at negative $\tilde{\nu}$. Those modifications are adopted in this work as they are specifically suited for discontinuous Galerkin discretizations.

The SA equation is written in conservation form as

$$\partial_t(\rho\tilde{\nu}) + \partial_i \mathcal{C}_i^{(\mathrm{SA})}(\mathbf{u}) - \partial_i \mathcal{D}_i^{(\mathrm{SA})}(\mathbf{u}) = \mathcal{S}^{(\mathrm{SA})}(\mathbf{u}), \tag{2.9}$$

where the state, $\mathbf{u}$, now contains the conserved quantity $\rho\tilde{\nu}$. In the remainder of the text, $\tilde{\nu}$ will be referred as the SA working variable. The convective and diffusive fluxes are respectively given by

$$\mathcal{C}_i^{(\mathrm{SA})}(\mathbf{u}) = \rho v_i \tilde{\nu}, \quad \mathcal{D}_i^{(\mathrm{SA})}(\mathbf{u}) = \frac{\eta}{\sigma}\partial_i \tilde{\nu}, \tag{2.10}$$

where σ is a closure parameter and η is a diffusivity defined as,

$$\eta = \begin{cases} \mu(1+\chi), & \chi \geq 0 \\ \mu(1+\chi+\frac{1}{2}\chi^2), & \chi < 0. \end{cases} \tag{2.11}$$

The source term is given by a balance of production, distribution and destruction terms,

$$\mathcal{S}^{(\mathrm{SA})}(\mathbf{u}) = \mathcal{P}^{(\mathrm{SA})}(\mathbf{u}) + \mathcal{B}^{(\mathrm{SA})}(\mathbf{u}) - \mathcal{T}^{(\mathrm{SA})}(\mathbf{u}). \tag{2.12}$$

The distribution term remains unaltered from the original model,

$$\mathcal{B}^{(\mathrm{SA})}(\mathbf{u}) = \sigma^{-1} c_{b2} \rho \partial_j \tilde{\nu} \partial_j \tilde{\nu}. \tag{2.13}$$

The production and destruction terms, however, are modified to ensure stability of the magnitude of $\tilde{\nu}$.

The modified production term is given by

$$\mathcal{P}^{(\mathrm{SA})}(\mathbf{u}) = \begin{cases} c_{b1}\tilde{s}\rho\tilde{\nu} & \chi \geq 0, \\ c_{b1}\tilde{s}\rho\tilde{\nu} g_n(\chi) & \chi < 0, \end{cases} \tag{2.14}$$

and

$$\tilde{s} = \begin{cases} |\omega| + \bar{s} & \bar{s} \geq -c_{v2}|\omega|, \\ |\omega| + \dfrac{|\omega|(c_{v2}^2|\omega| + c_{v3}\bar{s})}{(c_{v3} - 2c_{v2})|\omega| - \bar{s}} & \bar{s} < -c_{v2}|\omega|, \end{cases} \tag{2.15}$$

where $|\omega| = \sqrt{2\Omega_{ij}\Omega_{ij}}$ is the vorticity magnitude and the function g_n provides C^1-continuity to $\mathcal{P}^{(\mathrm{SA})}$ at $\tilde{\nu} = 0$,

$$g_n(\chi) = 1 - \frac{10^3\chi}{1+\chi^2}. \tag{2.16}$$

The modified destruction term is

$$\mathcal{T}^{(\mathrm{SA})}(\rho\tilde{\nu}) = \begin{cases} c_{w1} f_w \dfrac{\rho\tilde{\nu}^2}{d_w^2} & \chi \geq 0, \\ -c_{w1}\dfrac{\rho\tilde{\nu}^2}{d_w^2} & \chi < 0, \end{cases} \tag{2.17}$$

where d_w is distance to the nearest wall and the wall function f_w is given by

$$f_w = g\left(\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right)^{\frac{1}{6}}, \quad g = r + c_{w2}(r^6 - r), \text{ and } \quad r = \frac{\tilde{\nu}}{\tilde{s}\kappa^2 d_w^2}. \tag{2.18}$$

The closure functions are

$$\bar{s} = \frac{\tilde{\nu} f_{v2}}{\kappa^2 d_w^2}, \quad f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \tag{2.19}$$

and the coefficients for the SA model are given in Table 2.1.

Table 2.1: Spalart-Allmaras turbulence model closure parameters.

Parameter	Value
c_{b1}	0.1355
c_{b2}	0.622
σ	2/3
κ	0.41
c_{w1}	$c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma$
c_{w2}	0.3
c_{w3}	2.0
c_{v1}	7.1
c_{v2}	0.7
c_{v3}	0.9

Finally, the SA equation is coupled with the Navier-Stokes system through the diffusion term in the momentum equation. We use Boussinesq's assumption and augment the viscous stress tensor in Eqn. 2.5 with the eddy viscosity μ_t as follows:

$$\tau_{ij} = (\mu + \mu_t)(\partial_i v_j + \partial_j v_i) + \lambda\delta_{ij}\partial_k v_k, \tag{2.20}$$

where μ_t is also modified according to

$$\mu_t = \begin{cases} \rho\tilde{\nu} f_{v1} & \tilde{\nu} > 0 \\ 0 & \tilde{\nu} \leq 0. \end{cases} \tag{2.21}$$

2.3 Discontinuous Galerkin Spatial Discretization

In this section, we describe the discontinuous Galerkin (DG) spatial discretization of the flow equations. Let $\mathcal{V}^{H,p}$ be the space of piecewise polynomials of degree p with local support on each element $\kappa^H \in T^H$, where T^H is the set of elements resulting from a non-overlapping discretization of the domain, D. Using the method of weighted residuals, the spatial terms of the flow equations are written in the following semilinear form:

$$\mathbb{R}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p}) = \sum_{\kappa^H \in T^H} \mathbb{C}_{\kappa^H}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p}) - \mathbb{D}_{\kappa^H}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p}) = 0, \tag{2.22}$$

where $\mathrm{w}_s^{H,p}$ and $\mathrm{u}_s^{H,p}$ are piecewise polynomials that reside in $\mathcal{V}^{H,p}$ and s is a local index in each element that corresponds to the conserved state components. $\mathbb{C}$ and $\mathbb{D}$ are weighted residual statements of the convective and diffusive terms.

The weak form of the convective term splits into volume and boundary integrals via integration by parts and by invoking Gauss's theorem,

$$\begin{aligned}\mathbb{C}_{\kappa^H}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p}) \quad = \quad & -\int\limits_{\kappa^H} \partial_i \mathrm{w}_s^{H,p} \mathcal{C}_{is}(\mathbf{u}^{H,p}) d\mathbf{x} \\ & + \int\limits_{\partial\kappa^H} \mathrm{w}_s^{(H,p)+} \widehat{\mathcal{C}}_{is}(\mathbf{u}^{(H,p)+}, \mathbf{u}^{(H,p)-}, \mathbf{n}) \; ds.\end{aligned} \tag{2.23}$$

The superscripts + and − respectively indicate values corresponding to the interior and exterior of element κ^H on the boundary $\partial\kappa^H$ with outward normal $\mathbf{n}$. The Riemann flux, $\widehat{\mathcal{C}}_{is}$, is approximated with Roe's [68] solver in which the SA working variable is transported as a conserved scalar.

The diffusion term is discretized using the second form of Bassi & Rebay [6] (BR2) in which $\mathcal{D}_{is}$ is expressed as

$$\mathcal{D}_{is}(\mathbf{u}) = \mathcal{A}_{isjk}(\mathbf{u}) \partial_j \mathrm{u}_k, \tag{2.24}$$

where the tensor $\mathcal{A}_{isjk}$ is a nonlinear function of the state vector. Note, i, j index the spatial dimension and s, k index the state vector.

The weak form of the diffusion term is obtained via integration by parts and by invoking Gauss's theorem.

$$\begin{aligned}
\mathbb{D}_{\kappa^H}(\mathbf{u}^{H,p},\mathbf{w}^{H,p}) = \; & -\int\limits_{\kappa^H} \partial_i \mathrm{w}_s^{H,p} \mathcal{A}_{isjk}(\mathbf{u}^{H,p}) \partial_j \mathrm{u}_k^{H,p} d\mathbf{x} \\
& + \int\limits_{\partial\kappa^H} \partial_i \mathrm{w}_s^{(H,p)+} \mathcal{A}_{isjk}(\mathbf{u}^{(H,p)+}) \mathrm{u}_k^{(H,p)+} \mathrm{n}_j \, ds \\
& - \int\limits_{\partial\kappa^H} \partial_i \mathrm{w}_s^{(H,p)+} \widehat{\mathcal{A}_{isjk} \mathrm{u}_k^{H,p}} \mathrm{n}_j \, ds + \int\limits_{\partial\kappa^H} \mathrm{w}_s^{(H,p)+} \widehat{\mathcal{D}}_{is} \mathrm{n}_i \, ds
\end{aligned} \tag{2.25}$$

In Equation 2.25, $\widehat{\cdot}$ indicates flux averaging of discontinuous quantities and $\widehat{\mathcal{D}}_{is}$ is a viscous flux that includes jump stabilization terms. For the conservation of mass, momentum, and energy, the choice of flux averages that yields a compact, primal and dual-consistent discretization is shown in Table 2.2 [20].

Table 2.2: Viscous fluxes for Navier-Stokes.

	$\widehat{\mathcal{D}}_{is}$	$\widehat{\mathcal{A}_{isjk} \mathrm{u}_k^{H,p}}$
Interior faces	$\{\mathcal{A}_{isjk}(\mathbf{u}^{H,p}) \partial_j \mathrm{u}_k^{H,p}\} - \eta^f \{\delta_{is}^{(H,p)f}\}$	$\mathcal{A}_{isjk}(\mathbf{u}^{(H,p)+}) \{\mathrm{u}_k^{H,p}\}$
Dirichlet boundary	$\mathcal{A}_{isjk}(\mathbf{u}^{(H,p)b}) \partial_j \mathrm{u}_k^{(H,p)+} - \eta^{bf} \delta_{is}^{(H,p)bf}$	$\mathcal{A}_{isjk}(\mathbf{u}^{(H,p)b}) \{\mathrm{u}_k^{(H,p)b}\}$
Neumann boundary	$\mathcal{A}_{isjk}(\mathbf{u}^{(H,p)b}) \partial_j \mathrm{u}_k^{(H,p)b}$	$\mathcal{A}_{isjk}(\mathbf{u}^{(H,p)+}) \mathrm{u}_k^{(H,p)+}$

In Table 2.2, $\{\cdot\}$ is the arithmetic average $\{\cdot\} = \frac{1}{2}((\cdot)^+ + (\cdot)^-)$, and the superscript b indicates constructed states that enforce the boundary conditions. η^f and η^{bf} are stability constants defined by:

$$\eta^f = \kappa_{\mathrm{BR2}} \cdot \max(N_{\kappa^{H+}}^{\mathrm{face}}, N_{\kappa^{H-}}^{\mathrm{face}}), \qquad \eta^{bf} = \frac{N_{\kappa^{H+}}^{\mathrm{face}}}{2}, \tag{2.26}$$

where $N_{\kappa^H}^{\mathrm{face}}$ is the number of faces of element κ^H and $\kappa_{\mathrm{BR2}} \geq 1$ is a user-defined stabilization factor. $\delta_{is}^{(H,p)f}$, $\delta_{is}^{(H,p)bf}$ are auxiliary variables corresponding to internal and boundary faces respectively that satisfy the integral equations $\forall \mathrm{v}_{is} \in \mathcal{V}^{H,p}$,

$$\int_{\kappa^{H+}} \delta_{is}^{(H,p)f+} \mathrm{v}_{is} d\mathbf{x} + \int_{\kappa^{H-}} \delta_{is}^{(H,p)f-} \mathrm{v}_{is} d\mathbf{x} = \int_{\sigma^f} \{\mathrm{v}_{is} \mathcal{A}_{isjk}(\mathbf{u}^{H,p})\}(\mathrm{u}_k^{(H,p)+} - \mathrm{u}_k^{(H,p)-})\mathrm{n}_j \; d\mathbf{s},$$
$$\int_{\kappa^{H}} \delta_{is}^{(H,p)bf} \mathrm{v}_{is} d\mathbf{x} = \int_{\sigma^{bf}} \mathrm{v}_{is} \mathcal{A}_{isjk}(\mathbf{u}^{(H,p)b})(\mathrm{u}_k^{(H,p)+} - \mathrm{u}_k^{(H,p)b})\mathrm{n}_j \; d\mathbf{s},$$

where the superscripts f^+ and f^- indicate each side of an interior face and σ^f and σ^{bf} indicate interior and boundary faces respectively. Note that v_{is} are components of a vector test function where i indexes the spatial directions and s indexes the state components.

For the diffusive fluxes and the source terms in the SA model, we use Oliver's [59] dual-inconsistent formulation due to its simpler implementation and similar output-adapted results when compared to dual-consistent formulations which introduce additional terms into $\mathbb{R}(\mathbf{u}^{H,p}, \mathbf{w}^{H,p})$.

In each element, the state $\mathrm{u}_s^{H,p}$ and the weight functions $\mathrm{w}_s^{H,p}$ are expanded in terms of the basis functions $\phi_b^{H,p}$ as follows:

$$\mathrm{u}_s^{H,p} = \mathrm{U}_{sb}\phi_b^{H,p}(\mathbf{x}), \quad \mathrm{w}_s^{H,p} = \mathrm{W}_{sb}\phi_b^{H,p}(\mathbf{x}), \tag{2.27}$$

where U_{sb} is the unknown discrete state. Note that the number of unknowns per element is $N_b \times N_s$, where N_b is the number of basis functions in the element and N_s is the number of components in the conserved state vector. The discrete residual operator is obtained by substituting the expressions in Eqn. 2.27 into the semi-linear form of Eqn. 2.22 and choosing, in each element, W_{sb} to have the value of 1 for each combination of state and basis components. Finally, the semi-discrete flow equations are written as:

$$\mathbf{M}\mathrm{d}_t\mathbf{U} = -\mathbf{R}(\mathbf{U}), \tag{2.28}$$

where $\mathbf{R}$ is the discrete residual operator and $\mathbf{M}$ is the block diagonal mass matrix that corresponds to the volume integral of basis function products on each element in the mesh. In the interest of notation, we will refer to the discrete residual and state as vectors that correspond to unrolling U_{sb},

$$\mathrm{U}_l \Leftarrow \mathrm{U}_{sb}\mathrm{V}_{sbl}, \tag{2.29}$$

where V_{sbl} is a bookkeeping tensor that encodes the unrolling and the index l ranges from 1 to the total number of unknowns in the discrete system. The time integration of Eqn. 2.28 is described in Chapter 3.

2.3.1 Physicality constraints

The flow field is subject to physicality constraints that are not guaranteed to be satisfied as the discretized equations only enforce conservation. In this section we review these constraints for the RANS-SA equations, and in Section 3.2 we present the method by which we incorporate the constraints into the solver.

2.3.1.1 Thermodynamic realizability

The thermodynamic realizability constraints are:

$$\begin{aligned} \frac{p(\mathbf{u}(t,\mathbf{x}))}{p_\infty} &> 0, \\ \frac{\rho(\mathbf{u}(t,\mathbf{x}))}{\rho_\infty} &> 0, \end{aligned} \tag{2.30}$$

where p_∞ and ρ_∞ refer to free-stream pressure and density, respectively. These quantities are included here only for non-dimensional convenience and they clearly do not alter the positivity constraints. Note that ρ is a conserved variable and, therefore, its extrema match the extrema of the corresponding position in the conserved state $\mathbf{u}$. Pressure, however, does not have this property. In fact, its curvature along a spatial direction ζ is given by

$$\frac{\partial^2 p}{\partial \zeta^2} = \left(\frac{\partial \mathbf{u}}{\partial \zeta}\right)^T \underbrace{\frac{\partial^2 p}{\partial \mathbf{u} \partial \mathbf{u}^T}}_{\mathcal{H}_p} \left(\frac{\partial \mathbf{u}}{\partial \zeta}\right) + \left(\frac{\partial p}{\partial \mathbf{u}}\right)^T \frac{\partial^2 \mathbf{u}}{\partial \zeta^2}. \tag{2.31}$$

The eigenvalues of the Hessian of the pressure with respect to the state are

$$\text{eig}_s(\mathcal{H}_p) = \begin{cases} 0, & \text{for } s = 1, 2, \\ -\dfrac{\gamma - 1}{\rho}, & \text{for dim} > 1,\ s = 3 \rightarrow \text{dim} + 1, \\ -\dfrac{(\gamma - 1)(1 + v_i v_i)}{\rho}, & \text{for } s = \text{dim} + 2. \end{cases} \tag{2.32}$$

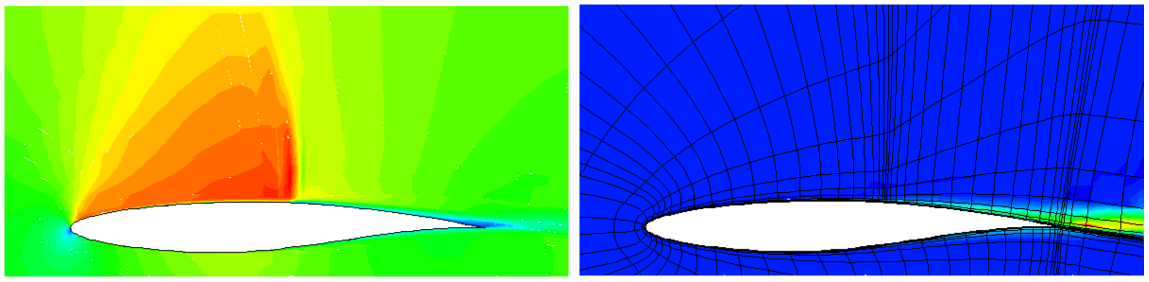
Note that for a linear distribution of state quantities along ζ, the only local extremum possible in pressure between two points is a maximum since the eigenvalues of $\mathcal{H}_p$ are non-positive. Therefore, when the state is linearly distributed, we only need to check the pressure constraint at the end points. However, it is difficult to ensure positivity for generic state distributions because the second term in Eqn. 2.31 involves the sum of positive and negative terms.

2.3.1.2 RANS

Physical intuition indicates that eddy viscosity should be constrained similarly to pressure and density, *i.e.* $\nu_t > 0$. Despite the fact that Oliver's modifications already impose that constraint, we found that enforcing positive total viscosity,

$$\frac{\nu(\mathbf{u}(t, \mathbf{x})) + \nu_t(\mathbf{u}(t, \mathbf{x}))}{\nu(\mathbf{u}(t, \mathbf{x}))} > 0, \tag{2.33}$$

in the solution path helps convergence in many flow cases. Similarly to the thermodynamic constraints, the physical kinematic viscosity in the denominator of Eqn. 2.33 makes the constraint non-dimensional. An example of a case where the constraint above helps convergence is shown in Figure 2.1 which presents turbulent, transonic flow over the RAE2822 airfoil. A $p = 1$ discretization is used to solve the flow equations with and without imposing Eqn. 2.33 and the residual convergence is shown in Figure 2.2 for both cases. The inclusion use of constraint 2.33 reduces the frequency at which the solution update is limited (explained in Chapter 3) which, in turn, helps convergence.



(a) Mach contours. (b) Mesh and $\rho\tilde{\nu}$ contours.

Figure 2.1: RAE2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$: Mach and $\rho\tilde{\nu}$ contours.

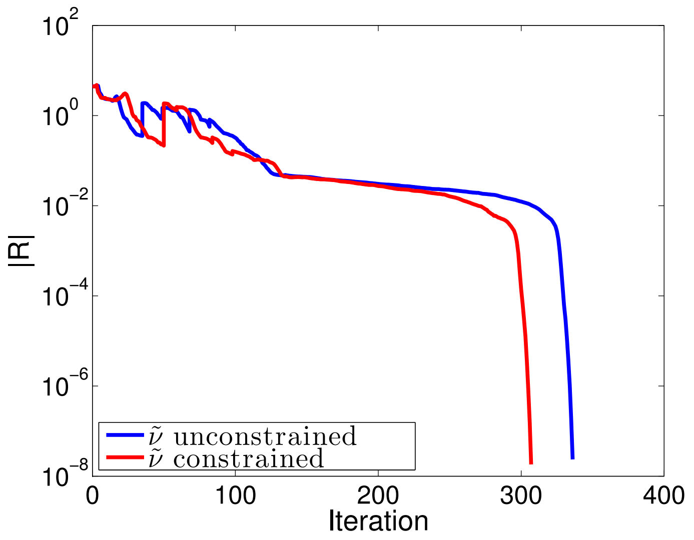


Figure 2.2: RAE2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$: residual convergence using $p = 1$ with and without the total viscosity constraint (Eqn. 2.33).

2.3.2 Shock-capturing

Some results presented in this thesis use the artificial viscosity method of Persson and Peraire [64] for capturing shocks. In this method, a regularity sensor is defined as the ratio

$$S_{\kappa^H} = \frac{(\rho^{H,p} - \tilde{\rho}^{H,p-1}, \rho^{H,p} - \tilde{\rho}^{H,p-1})|_{\kappa^H}}{(\rho^{H,p}, \rho^{H,p})|_{\kappa^H}} \tag{2.34}$$

where $(\cdot,\cdot)|_{\kappa^H}$ indicates an inner product restricted to the element κ^H, and $\tilde{\rho}^{H,p-1}$ is the solution for the mass conservation equation, $\rho^{H,p}$ projected to $\mathcal{V}^{H,p-1}$.

The following diffusion term is added to the left-hand side of the flow equations:

$$\mathcal{L}_s = -\partial_i(\varepsilon \partial_i \mathrm{u}_s), \tag{2.35}$$

where ε is the artificial diffusivity that, in general, can be an anisotropic tensor. However, this work uses an element-wise constant, isotropic viscosity that is defined by the switch,

$$\varepsilon_{\kappa^H} = \varepsilon_0 \frac{f^2}{f+1}, \qquad f \equiv \frac{S_{\kappa^H}}{S_0}. \tag{2.36}$$

The parameters S_0 and ε_0 are defined heuristically as,

$$S_0 = 0.5 \times 10^p, \quad \text{and} \quad \varepsilon_0 = \frac{\lambda_{\max} h_{\kappa^H}}{p},$$

where p is the element's polynomial order, $\lambda_{\max}$ is the element's maximum characteristic speed, and h_{κ^H} is the element's hydraulic diameter.

2.3.3 Scaling of the Discrete Equations

The Navier-Stokes equations are discretized in their dimensional form. In order to prevent ill-conditioning and to improve floating point precision, appropriate scales are chosen so that the numeric values of the conserved variables have similar orders of magnitude. These scales are not unique and can be dependent on the flow problem.

The scales used for the cases presented in this work were defined according to the following steps, which are geared for solutions to external flow problems:

- Given: free-stream Mach number (M_∞), Reynolds number (Re), reference length (c), angle of attack (α), and sideslip angle (β – 3D only).
- Set free-stream density and speed to $\rho_\infty = 1$ and $|\mathbf{v}_\infty| = 1$.
- Calculate the momentum components with the angle of attack α and angle of sideslip β:

$$\begin{aligned}
\text{2D:}\quad \rho_\infty v_{x_\infty} &= \rho_\infty |\mathbf{v}_\infty| \cos(\alpha), \\
\rho_\infty v_{y_\infty} &= \rho_\infty |\mathbf{v}_\infty| \sin(\alpha). \\
\text{3D:}\quad \rho_\infty v_{x_\infty} &= \rho_\infty |\mathbf{v}_\infty| \cos(\alpha)\cos(\beta), \\
\rho_\infty v_{y_\infty} &= \rho_\infty |\mathbf{v}_\infty| \cos(\alpha)\sin(\beta), \\
\rho_\infty v_{z_\infty} &= \rho_\infty |\mathbf{v}_\infty| \sin(\alpha).
\end{aligned}$$

- Use the specific heat ratio for air, $\gamma = 1.4$, and set the gas constant to $R = 0.4$. Note that this corresponds to setting the specific heats to $C_p = 1.4$ and $C_v = 1.0$.
- Calculate the free-stream pressure and temperature in the newly defined units:

$$p_\infty = \frac{\rho_\infty}{\gamma}\left(\frac{|\mathbf{v}_\infty|}{M_\infty}\right)^2, \quad T_\infty = \frac{p_\infty}{\rho_\infty R}.$$

- Calculate the total energy per unit volume:

$$\rho_\infty E_\infty = \frac{p_\infty}{\gamma - 1} + \rho_\infty \frac{|\mathbf{v}_\infty|^2}{2}.$$

- The temperature constants in Sutherland's law must be converted from known physical units into the new units. This requires knowledge of the free-stream temperature in the physical units, e.g. T_∞^{Kelvin}, in which case the conversion from ($T_{\text{ref}}^{\text{Kelvin}}$, $T_{\text{s}}^{\text{Kelvin}}$) to ($T_{\text{ref}}$, T_{s}) reads:

$$T_{\text{ref}} = T_{\text{ref}}^{\text{Kelvin}} \frac{T_\infty}{T_\infty^{\text{Kelvin}}}, \quad T_{\text{s}} = T_{\text{s}}^{\text{Kelvin}} \frac{T_\infty}{T_\infty^{\text{Kelvin}}}.$$

- Calculate the free-stream viscosity and the reference viscosity for Sutherland's law in the new units using the reference length c and the Reynolds number:

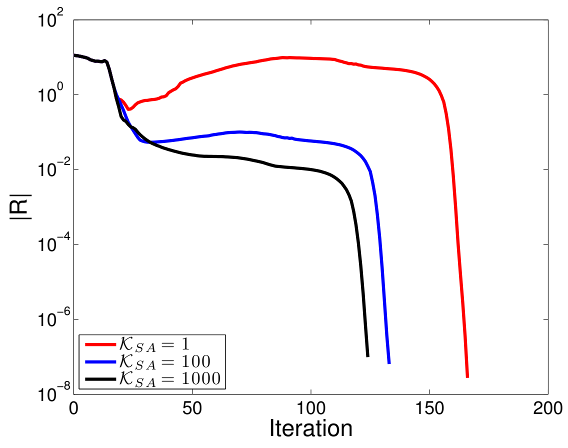
$$\mu_\infty = \frac{\rho_\infty |\mathbf{v}_\infty| c}{Re}, \quad \mu_{\text{ref}} = \frac{\mu_\infty}{\frac{T_{\text{ref}}+T_{\text{s}}}{T+T_{\text{s}}} \left(\frac{T_\infty}{T_{\text{ref}}}\right)^{1.5}}.$$

Most practical cases in the aeronautical industry are in the Reynolds number regime of $10^6 \to 10^7$. In this regime, $\tilde{\nu}/\nu_\infty$ typically ranges 4 to 5 orders of magnitude. Therefore, it is also desirable to choose an appropriate scale for $\tilde{\nu}$. The scale used in this work is

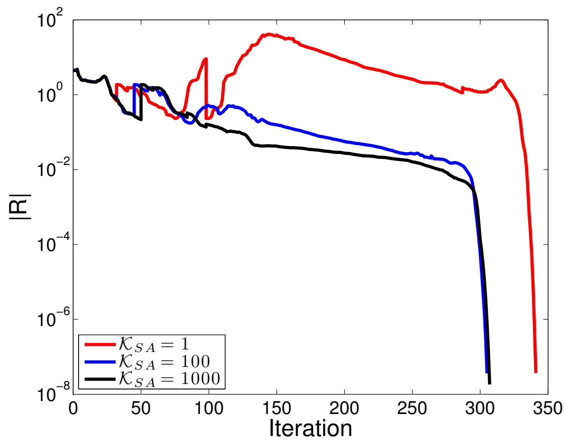
$$(\rho\tilde{\nu})' = \frac{\rho\tilde{\nu}}{\kappa_{\text{SA}} \mu_\infty}, \tag{2.37}$$

where $(\rho\tilde{\nu})'$ is the scaled conserved variable that is stored and evolved by the solver. κ_{SA} is a scaling factor and μ_∞ is the dynamic viscosity expressed in the units defined above. Essentially, we are non-dimensionalizing $\rho\tilde{\nu}$ by a factor larger than the physical viscosity.

To exemplify the effect of κ_{SA}, we show in Figure 2.3 the residual history for two flows at $Re = 6.5 \times 10^6$, one subsonic and one transonic. For each case, three scaling factors were used, $\kappa_{\text{SA}} = 1,\ 100,\ 1000$. Note that κ_{SA} significantly affects the convergence history. Specifically, the larger values of κ_{SA} ameliorate the secondary transient observed in RANS computations using DG [11].



(a) RAE2822 - $M_\infty = 0.3$, $Re = 6.5 \times 10^6$, $\alpha = 2.31°$.



(b) RAE2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$.

Figure 2.3: Residual convergence using $p = 1$ for different $\tilde{\nu}$ scaling factors (κ_{SA}).

The drag and lift coefficients (C_D and C_L respectively) for both flow conditions are shown in Table 2.3 and Table 2.4. As expected, the scaling factor has virtually no effect on the results. However, it makes the conserved variables closer in magnitude which, in turn, helps implicit time integration methods.

Table 2.3: RAE2822 - $M_\infty = 0.3,\ Re = 6.5 \times 10^6,\ \alpha = 2.31°$ – Comparison of force coefficients and maximum values of x-momentum and SA working variable for different scaling factors.

Quantity	$\kappa_{\text{SA}} = 1$	$\kappa_{\text{SA}} = 100$	$\kappa_{\text{SA}} = 1000$
C_D	0.0122	0.0122	0.0122
C_L	0.4507	0.4507	0.4506
$(\rho v_x)_{\text{max}}$	1.25182	1.25182	1.25192
$(\rho\tilde{\nu})'_{\text{max}}$	1.03775×10^3	1.03775×10^1	1.03783

Table 2.4: RAE2822 - $M_\infty = 0.734,\ Re = 6.5 \times 10^6,\ \alpha = 2.79°$ – Comparison of force coefficients and maximum values of x-momentum and SA working variable for different scaling factors.

Quantity	$\kappa_{\mathrm{SA}} = 1$	$\kappa_{\mathrm{SA}} = 100$	$\kappa_{\mathrm{SA}} = 1000$
C_D	0.0198	0.0198	0.0198
C_L	0.7334	0.7334	0.7334
$(\rho v_x)_{\max}$	1.11808	1.11808	1.11811
$(\rho\tilde{\nu})'_{\max}$	1.64361×10^3	1.64362×10^1	1.64378

CHAPTER 3

Time Integration

This chapter describes the solution advancement scheme. First, we review the pseudo-transient continuation (PTC) method and then we show how the physicality constraints are incorporated in the solution path. We then present different solution update methods and compare them for a set of cases ranging from intermediate to difficult.

3.1 Pseudo-transient Continuation

Since we are interested in the steady-state solution of the flow equations, high-accuracy is not required for discretizing the unsteady term of Eqn. 2.28. Instead, stability is the main attribute which makes backward Euler an attractive choice. The fully-discrete form of Eqn. 2.28 is then:

$$\mathbf{M}\frac{1}{\Delta t}(\mathbf{U}^{n+1} - \mathbf{U}^{n}) + \mathbf{R}(\mathbf{U}^{n+1}) = 0, \tag{3.1}$$

where $\mathbf{M}$ is the mass matrix and n indexes the time step.

The residual at the future state in Eqn. 3.1 is expanded about the current state and the steps in the iterative procedure require linear solves for the update $\Delta\mathbf{U}^k$,

$$\left(\mathbf{M}\frac{1}{\Delta t} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_{\mathbf{U}^k}\right)\Delta\mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k), \tag{3.2}$$

where k is used for the nonlinear iteration number to distinguish the method from the time-accurate backward Euler case. Note that for $\Delta t \to \infty$ the iterative procedure of Eqn. 3.2 reduces to Newton's root-finding method.

The linearization of the residual operator involves simplifications due to non-differentiable terms in numerical flux functions and artificial dissipation sensors. Additionally, the sparse structure of the linear system given in Eqn. 3.2 depends on the type of spatial scheme used for $\mathbf{R}$, and an appropriate choice of iterative solver and preconditioner must be made. In this work, a restarted Generalized Minimal Residual (GMRES) linear solver [70, 69], aided by a line-Jacobi preconditioner [19], solves the linear system at each step. The DG discretization described in Chapter 2 produces a residual Jacobian that is block-sparse: degrees of freedom in an element are coupled only to degrees of freedom in neighbor elements. Within each block, sparsity may exist for certain choices of basis functions, but we do not take advantage of such sparsity.

In the first stages of calculations initialized by states that do not satisfy all boundary conditions, strong transients are observed due to the propagation of boundary information into the domain. To alleviate those transients, small time steps are used in an attempt to make the solution follow a physical path. This causes a diagonal dominance in the coefficient matrix in Eqn. 3.2 and makes the calculation closer to time-accurate if Δt does not vary spatially. As an alternative to global time stepping, element-wise time steps can be used by setting a global CFL number defined as:

$$\mathrm{CFL} = \frac{\lambda_{\max}\Delta t_{\kappa^H}}{L_{\kappa^H}}, \tag{3.3}$$

where $\lambda_{\max}$ is the maximum wave speed and L_{κ^H} is a measure of element size, *e.g.* hydraulic diameter.

At each iteration, k, the flow state vector $\mathbf{U}^k$ is updated with $\Delta\mathbf{U}^k$. For robustness purposes, an under-relaxation parameter, ω^k, is used to ensure a physical solution at the next iteration (Eqn. 3.4).

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \omega^k \Delta\mathbf{U}^k \tag{3.4}$$

The methods for calculating ω^k are presented in Section 3.3.

3.1.1 CFL evolution strategies

In a pseudo-transient continuation method, the continuation parameter is the CFL number. Hence, a strategy must be chosen to evolve the CFL from its initial value to a large value such that Eqn. 3.2 becomes Newton's method and the state approaches the steady solution.

Switched Evolution Relaxation - SER

Many strategies for evolving the time step are available [9, 43]. Amongst them, a widely used strategy is the *Switched Evolution Relaxation* (SER) method proposed by Mulder and van Leer [55]. The general idea of SER is to change the time step or the CFL number based on a measure of convergence which is inferred from the reduction in a residual norm between consecutive iterations. Typically, the L_2 norm is used. The algorithm reads as follows:

$$\mathrm{CFL}^k = \min\left(\mathrm{CFL}^{k-1}\frac{|\mathbf{R}^{k-1}|_{L_2}}{|\mathbf{R}^k|_{L_2}}, \mathrm{CFL}_{\max}\right). \tag{3.5}$$

SER is an effective time step evolution strategy. However, the physicality constraints are verified after the direction $\Delta\mathbf{U}$ is computed and the relaxation parameter

ω in Eqn. 3.4 must be such that the updated state is physical. In the event of ω becoming too small and the time step not changing significantly, a contingency plan needs to be designed so that the direction $\Delta\mathbf{U}$ changes. This is discussed in Section 3.3.

Exponential progression with under-relaxation - EXPur

Alternatively, the CFL evolution can be based on the value of the under-relaxation parameter. Specifically, the CFL increases by a factor $\beta > 1$ if a full update ($\omega = 1$) happened in the previous step of the solver. On the other end, if $\omega < \omega_{\min}$ the CFL is reduced by multiplying it by $\kappa < 1$ and the solver step is repeated. The relaxation factor is limited such that the solution stays physical at selected limit points of the interpolated field $\mathbf{u}^{H,p}$. The methods for computing the under-relaxation factor are described in Section 3.3

This strategy accounts for the physical feasibility constraints for the next update. However, it is an indirect way of avoiding non-physical states in the flow field since the direction $\Delta\mathbf{U}^k$ may still produce states that are closer to becoming non-physical even at the minimum CFL. In particular, this is observed in highly under-resolved meshes.

The CFL evolution strategy is summarized below:

$$\mathrm{CFL}^{k+1} = \begin{cases} \beta \cdot \mathrm{CFL}^k & \text{for } \beta > 1 \quad \text{if} \quad \omega^k = 1 \\ \mathrm{CFL}^k & \qquad\qquad\quad \text{if} \quad \omega_{\min} < \omega^k < 1 \\ \kappa \cdot \mathrm{CFL}^k & \text{for } \kappa < 1 \quad \text{if} \quad \omega^k < \omega_{\min} \end{cases} . \tag{3.6}$$

The parameters are typically set to: $\omega_{\min} = 0.01$, $\beta = 1.05 \leftrightarrow 2.0$, and $\kappa = 0.1$.

Residual Difference Method - RDM

This CFL evolution strategy is based on a method described in Ref. [9] and it is a blend of EXP and SER. Similarly to SER, this strategy monitors the solution evolution and increases/decreases the CFL when the residual norm is reduced/increased. However, the maximum change in CFL is limited from above by a factor β. The algorithm reads as follows:

$$\mathrm{CFL}^k = \min\left(\mathrm{CFL}^{k-1} \cdot \beta^{\frac{|\mathbf{R}^{k-1}|_{L_2} - |\mathbf{R}^k|_{L_2}}{|\mathbf{R}^{k-1}|_{L_2}}}, \mathrm{CFL}_{\max}\right) \quad \text{for } \beta > 1. \tag{3.7}$$

Note that $\frac{\mathrm{CFL}^k}{\mathrm{CFL}^{k-1}} \leq \beta$, where the equality corresponds to $\mathbf{R}^k$ vanishing completely. A monotonic variant of RDM is obtained by setting the exponent in Eqn. 3.7 to zero when $|\mathbf{R}^k|_{L_2} > |\mathbf{R}^{k-1}|_{L_2}$. This variant will be referred as mRDM in the remainder of the text.

3.1.2 Optimization aspect of PTC

Assume the coefficient matrix in Eqn. 3.2 is real and non-singular and the update direction $\Delta\mathbf{U}^k$ is not zero. Multiplying the left-hand side of Eqn. 3.2 by its transpose gives:

$$\Delta\mathbf{U}^{kT} \underbrace{\left(\mathbf{M}\frac{1}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}\right)^T}_{A^T} \underbrace{\left(\mathbf{M}\frac{1}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}\right)}_{A} \Delta\mathbf{U}^k = -\Delta\mathbf{U}^{kT} \underbrace{A^T\mathbf{R}(\mathbf{U^k})}_{\frac{\partial f}{\partial \mathbf{U}}\big|_{\mathbf{U}^k}} > 0. \tag{3.8}$$

We note that the inequality comes from the fact that A^TA on the left-hand side is a symmetric positive-matrix. Therefore, $\Delta\mathbf{U}^k$ is a descent direction for the scalar function $f(\tilde{\mathbf{U}})$ defined by its gradient in the right-hand side of Eqn. 3.8. Here, $\tilde{\mathbf{U}}$ is a trial future state.

Now, consider the unsteady residual,

$$\mathbf{R}_t(\tilde{\mathbf{U}}) = \mathbf{M}\frac{1}{\Delta t}(\tilde{\mathbf{U}} - \mathbf{U}^k) + \mathbf{R}(\tilde{\mathbf{U}}), \tag{3.9}$$

By taking one step of Newton's method for $\mathbf{R}_t(\tilde{\mathbf{U}}) = 0$ with initial guess $\tilde{\mathbf{U}}^0 = \mathbf{U}^k$, we conclude that,

$$f(\tilde{\mathbf{U}}) = \frac{1}{2}|\mathbf{R}_t(\tilde{\mathbf{U}})|^2_{L_2} = \frac{1}{2}\mathbf{R}_t(\tilde{\mathbf{U}})^T\mathbf{R}_t(\tilde{\mathbf{U}}). \tag{3.10}$$

This is verified by differentiating Eqn. 3.10 and setting $\tilde{\mathbf{U}} = \mathbf{U}^k$. Consequently, there is a trial state $\tilde{\mathbf{U}}$ along the direction $\Delta\mathbf{U}^k$ such that $f(\tilde{\mathbf{U}}) < f(\mathbf{U}^k)$.

3.2 Incorporating constraints

The minimization character of the PTC method motivates the use of constraint handling techniques from the optimization field. Non-physical states (*e.g.* negative pressure) can lead to instability[3], and therefore we need to keep the iterates within the physical region of the solution space. Interior penalty methods [36] are attractive because of their simplicity and efficiency in acknowledging feasibility constraints in the solution path. These methods augment a scalar objective function with a term that tends to infinity as the solution path approaches a feasibility boundary creating a repelling effect with respect to prohibited regions of the domain.

A different approach for incorporating constraints into pseudo-transient methods was proposed by Kelley *et al.* [44]. Their approach involves a step that projects the state into the feasible domain after each non-linear iteration and the fundamental difference between their method and the method we propose here is that we incorporate the constraints when computing the solution update.

3.2.1 Scalar penalization

A simple way of incorporating the realizability constraints in the solution path is to formulate an optimization problem that reads:

$$\boxed{\begin{array}{rl} \text{minimize:} & f(\mathbf{U}) = |\mathbf{R}_t(\mathbf{U})|^2_{L_2} \\[1em] \text{by varying:} & \mathbf{U},\ \text{and}\ \Delta t \to \infty \\[1em] \text{subject to:} & c_i(\mathbf{u}^{H,p}(t,\mathbf{x})) > 0. \quad \forall \mathbf{x} \in D \end{array}}$$

The constraints, $c_i(\mathbf{u}^{H,p}(t,\mathbf{x})) > 0$, are dependent on the equations being solved. In this work, we consider the RANS-SA constraints presented in Section 2.3.1.

An interior penalty method handles the constraints by augmenting the objective function, $f(\mathbf{U})$, with an inverse-barrier function of $c_i(\mathbf{u}^{H,p}(t,\mathbf{x}))$. Since the constraints are applied to a functional representation of the state, an integral of the inverse barrier would have to be evaluated in order to enforce the constraints everywhere in the domain. We approximate this integral by using a quadrature rule and the penalty function is written as,

$$\mathbb{P}(\mathbf{U},\mu) = \mu \sum_{\kappa^H \in T^H} \sum_{i}^{N_c} \sum_{q}^{N_q} \frac{w_q}{c_i(\mathbf{u}^{H,p}(x_q))}, \tag{3.11}$$

where N_q is the number of quadrature points $\mathbf{x}_q$ with weights w_q, N_c is the number of constraints indexed by i, and μ, in this context, is a scalar penalty factor. Note that $\mathbb{P}$ in Eqn. 3.11 tends to infinity as the constraints approach zero from the positive side. The augmented function is then given by:

$$g(\mathbf{U},\mu) = f(\mathbf{U}) + \mathbb{P}(\mathbf{U},\mu). \tag{3.12}$$

The main idea of the interior penalty method is to solve a sequence of optimization problems for diminishing penalty factors, $\mu^{j+1} < \mu^j$. For each optimization problem j, Newton's approach can be used to compute a search direction for a minimizer of $g(\mathbf{U}, \mu^j)$,

$$\mathcal{H}_g(\mathbf{U}^k, \mu^j)\Delta\mathbf{U}^k = -\frac{\partial g}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k,\mu^j}, \tag{3.13}$$

where $\mathcal{H}_g$ is the Hessian of the augmented objective function,

$$\mathcal{H}_g(\mathbf{U}, \mu) = \underbrace{\frac{\partial \mathbf{R}_t}{\partial \mathbf{U}}\left(\frac{\partial \mathbf{R}_t}{\partial \mathbf{U}}\right)^T + \frac{\partial^2 \mathbf{R}_t}{\partial \mathbf{U}\partial \mathbf{U}^T}\mathbf{R}_t(\mathbf{U})}_{\mathcal{H}_f(\mathbf{U})} + \mathcal{H}_{\mathbb{P}}(\mathbf{U}, \mu) \tag{3.14}$$

and the Hessian of the penalty function, $\mathcal{H}_{\mathbb{P}}$, is a block diagonal matrix due to the local support of $\mathbf{u}^{H,p}$.

It is customary in nonlinear least-squares problems to use the Gauss-Newton [44] approximation to the Hessian, *i.e.*to ignore the second derivative of $\mathbf{R}_t$ with respect to $\mathbf{U}$:

$$\mathcal{H}_g \approx \tilde{\mathcal{H}}_g(\mathbf{U}, \mu) = \frac{\partial \mathbf{R}_t}{\partial \mathbf{U}}\left(\frac{\partial \mathbf{R}_t}{\partial \mathbf{U}}\right)^T + \mathcal{H}_{\mathbb{P}}(\mathbf{U}, \mu). \tag{3.15}$$

In spite of the regularization effect of $\mathcal{H}_{\mathbb{P}}$ and $\mathbf{M}\frac{1}{\Delta t}$ via their diagonal dominance, in general, the simplified full Hessian in Eqn. 3.15 is very ill-conditioned due to the squaring of the residual Jacobian, and solving the linear system in Eqn. 3.13 is very computationally expensive. This is the disadvantage of having a scalar function in Eqn. 3.10 which permits a simple penalization for enforcing the constraints. Additionally, factorizing $\tilde{\mathcal{H}}_g$ would generally require the explicit construction of that matrix which can be computationally intensive even for small problems. For this reason, the pure optimization approach is intractable for any realistic problem.

3.2.2 Vector penalization

As an alternative to the scalar penalization, we propose augmenting the residual with a penalty vector to account for the constraints:

$$\mathbf{R}_p(\mathbf{U}) = \mathbf{R}(\mathbf{U}) + \mathbf{P}(\mathbf{U}, \mu). \tag{3.16}$$

In order to have the repelling effect with respect to non-feasible regions of the domain, the penalization vector $\mathbf{P}$ must have a positive projection on the direction of the residual vector $\mathbf{R}$. To satisfy this requirement, we define the penalization vector as:

$$\mathbf{P}(\mathbf{U}, \mu) = \Phi(\mathbf{U}, \mu)\ \mathbf{R}(\mathbf{U}), \tag{3.17}$$

where μ is a penalty factor and Φ is a diagonal matrix of the same size as the residual Jacobian with the elemental penalties $\mathbb{P}_{\kappa^H}$ for each row corresponding to an element κ^H.

$$\Phi_{ij}(\mathbf{U}, \mu) = \begin{cases} \mu\ \mathbb{P}_{\kappa^H}(\mathbf{U}) & \text{if } i = j \in \text{dof}(\kappa^H) \\ 0 & \end{cases} \tag{3.18}$$

Note that $j \in \text{dof}(\kappa^H)$ denotes the degrees of freedom, in global ordering, pertinent to κ^H. The elemental penalty is given by:

$$\mathbb{P}_{\kappa^H}(\mathbf{U}) = \sum_{i}^{N_i} \sum_{q}^{N_q} \frac{w_q}{c_i(\mathbf{u}^{H,p}(\mathbf{x}_q))}. \tag{3.19}$$

Equation 3.19 involves a summation over quadrature points, $\mathbf{x}_q$, that lie inside κ^H, with weights w_q. This summation corresponds to integrating the inverse barrier function in a reference element with unitary volume.

Note that the projection of $\mathbf{P}$ – as defined in Eqn. 3.17 – onto the residual vector is always positive for non-zero $\mathbf{R}$ since the elemental penalties are strictly positive in the feasible domain, *i.e.*, physical states.

A root of the residual operator corresponds to a root of $\mathbf{R}_p$, so that the steady-state solution is independent of the values of the elemental penalties. We emphasize that the objective of this method is to change the path to the solution, not the solution itself. By applying the pseudo-transient continuation procedure (Eqn. 3.2) to $\mathbf{R}_p$ we are including physicality constraints in the solution path from the initial condition to steady state. The update direction along that path at step k satisfies

$$\left(\underbrace{(\mathbf{I}+\Phi^k)^{-1}\frac{\mathcal{M}}{\Delta t}}_{a} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k} + \underbrace{(\mathbf{I}+\Phi^k)^{-1}\left(\frac{\partial \Phi}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}\mathbf{R}(\mathbf{U}^k)\right)}_{b} \right)\Delta\mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k), \quad (3.20)$$

where $\mathbf{I}$ is the identity matrix and $\Phi^k = \Phi(\mathbf{U}^k, \mu^k)$. The equation above is derived by the substituting $\mathbf{R}_p$ into Eqn. 3.2 and by separating the terms such that the unpenalized residual, $\mathbf{R}$, is on the right hand-side. This adds the implementation convenience of simply adding entries to the coefficient matrix of the linear systems solved at each step k.

The terms "a" and "b" in Eqn. 3.20 are block diagonal for the DG method in this work. Additionally, the elemental CFL number gets amplified by $(1+\mu\,\mathbb{P}_{\kappa^H})$ as $\mathbf{I}+\Phi^k$ is a diagonal matrix. In the limit of an infinite time step, the solution path seeks a minimum of $|\mathbf{R}_p|_{L_2}$. Similarly, the time continuation term "a" vanishes at elements where the solution approaches a non-physical region while the penalization term "b" does not vanish because the function value of inverse barrier penalties (Eqn. 3.19) tends to infinity at a slower rate than the magnitude of its derivative. In the remainder of the text, we will refer to the method in Eqn. 3.20 as *Constrained Pseudo-transient Continuation* (CPTC).

The penalty factor is evolved using a form of SER:

$$\mu^{k+1} = \min\left(\mu^k \frac{1+\mu^k \max(\mathbb{P}_{\kappa^H}(\mathbf{U}^k))}{1+\mu^{k-1}\max(\mathbb{P}_{\kappa^H}(\mathbf{U}^{k-1}))}, \mu_{\max}\right). \tag{3.21}$$

This evolution strategy for μ tries to make the solver acknowledge the presence of a feasibility constraint by increasing its repelling effect as the solution path goes towards a non-physical state anywhere in the domain. This latter point is due to the use of the max function over the elements in Eqn. 3.21. Conversely, if the solution path is moving away from a feasibility boundary the repelling effect decreases.

The penalty factor is initialized such that $(1 + \mu \ \max(\mathbb{P}_{\kappa^H})) = \mathcal{O}(1)$ but $\mu > 0$. This keeps the pseudo-transient term active and alleviates the initial solution transients while helping the spectral conditioning of initial linear systems. For all the results in this thesis, μ is initialized according to:

$$1 + \mu^0 \max(\mathbb{P}_{\kappa^H}(\mathbf{U}^0)) = 10^{0.25}. \tag{3.22}$$

The CPTC method is summarized in Algorithm 3.1. The unconstrained PTC method follows a similar algorithm, where the steps related to the penalty factor (steps 3 and 10) are ignored and the update direction (step 6) is computed using Eqn. 3.2. For all the cases presented here, the CFL is reduced by a factor $\kappa = 0.1$ when the under-relaxation factor is below $\omega_{\min} = 0.01$. At that point the state is reverted to a safe state stored when a full update occurs.

ALGORITHM 3.1 Constrained PTC

```
1: Choose initial CFL and its evolution strategy (Section 3.1.1)
2: Set a residual tolerance, $\varepsilon_{\text{res}}$
3: Initialize $\mu$ according to Eqn. 3.22
4: Initialize $\mathbf{U}_{\text{safe}}$ to initial condition
5: while $|\mathbf{R}(\mathbf{U}^k)| > \varepsilon_{\text{res}}$, $k <$ maximum iterations do
6:     Compute $\Delta\mathbf{U}^k$ by solving Eqn. 3.20 using GMRES
7:     Compute under-relaxation parameter $\omega^k$ (Section 3.3)
8:     if $\omega^k \geq \omega_{\min}$ then
9:         $\mathbf{U}^{k+1} \leftarrow \mathbf{U}^k + \omega^k \Delta U^k$.
10:        Evolve $\mu$ using Eqn. 3.21
11:        Evolve CFL with chosen strategy
12:        if $\omega^k = 1$ then
13:            $\mathbf{U}_{\text{safe}} \leftarrow \mathbf{U}^{k+1}$                ▷ Store a safe state
14:        end if
15:    else
16:        $\text{CFL}^k \leftarrow \kappa\ \text{CFL}^k$ for $\kappa < 1$
17:        $\mathbf{U}^{k+1} \leftarrow \mathbf{U}_{\text{safe}}$                ▷ Revert to last safe state
18:        Return to step 6
19:    end if
20:    $k \leftarrow k + 1$
21: end while
```

3.3 Solution update

The solution update methods described here use two main ingredients. First, they require interpolating the state, $\mathbf{u}^{H,p}$, and its update, $\Delta\mathbf{u}^{H,p}$, at certain points, $\mathbf{x}_m$. This involves evaluating the basis functions at $\mathbf{x}_m$ and using the unrolling tensor in

Eqn. 2.29 to transfer the discrete vectors $\mathbf{U}$ and $\Delta\mathbf{U}$ to their field representations, $\mathbf{u}^{H,p}$ and $\Delta\mathbf{u}^{H,p}$. The second ingredient is an update limiter that restricts the changes in primitive variables to a maximum fraction, $\eta_{\max}$, of the current values. This procedure is described for the RANS-SA equations in Algorithm 3.2.

ALGORITHM 3.2 Limit physical update

```
1:  Given $\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$, $\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$, and a fraction $\eta_{\max} < 1$
2:  $\omega_{\kappa^H} \leftarrow 1$
3:  **for all** $\mathbf{x}_m \in \kappa^H$ **do**
4:      $\omega_\rho \leftarrow 1$                                              ▷ $\omega_\rho$ is the step size for density
5:      $\rho_m = \rho(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$                                        ▷ Current density at $\mathbf{x}_m$
6:      $\tilde{\rho}_m = \rho(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H} + \omega_\rho \Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$               ▷ Trial density at $\mathbf{x}_m$
7:      **if** $\tilde{\rho}_m$ is not within $\eta_{\max}$ of $\rho_m$ **then**
8:          Reduce $\omega_\rho$ such that $\tilde{\rho}_m$ is within $\eta_{\max}$ of $\rho_m$
9:      **end if**
10:     $\omega_p \leftarrow \omega_\rho$                                             ▷ $\omega_p$ is the step size for pressure
11:     $p_m = p(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$                                        ▷ Current pressure at $\mathbf{x}_m$
12:     $\tilde{p}_m = p(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H} + \omega_p \Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$               ▷ Trial pressure at $\mathbf{x}_m$
13:     **while** $\tilde{p}_m$ is not within $\eta_{\max}$ of $p_m$ **do**
14:         $\omega_p \leftarrow \frac{\omega_p}{2}$
15:         $\tilde{p}_m = p(\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H} + \omega_p \Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H})$
16:     **end while**
17:     Limit $\omega_{\tilde{\nu}}$ such that $\chi$ changes by a maximum fraction $\eta_{\max}$
18:     $\omega_{\kappa^H} \leftarrow \min(\omega_\rho, \omega_p, \omega_{\tilde{\nu}}, \omega_{\kappa^H})$
19: **end for**
20: **return** $\omega_{\kappa^H}$
```

Some clarifications are in order. First, the maximum fractional change is fixed at $\eta_{\max} = 10\%$ – based on experimentation – for all cases presented in this thesis. Also, the points $\mathbf{x}_m$ can be chosen arbitrarily, the more the better, and we select them to be the quadrature points used for computing the interior and boundary integrals involved in the residual calculation. Finally, the bisection method is used in step 13 of Algorithm 3.2 because pressure is a nonlinear function of the state.

3.3.1 Maximum Primitive Change

The *Maximum Primitive Change* (MPC) method limits the step size such that the physical update limiter (Algorithm 3.2) passes for all the elements in the mesh. This procedure is summarized in Algorithm 3.3.

ALGORITHM 3.3 Maximum Primitive Change

```
1: $\omega^k \leftarrow 1$                                         ▷ Assume full update initially
2: for all $\kappa^H \in T^H$ do                                 ▷ Loop over element in the mesh
3:     Select limit points, $\mathbf{x}_m$
4:     Evaluate $\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$ and $\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$
5:     Call Algorithm 3.2                                         ▷ Limit change in primitive state
6:     $\omega^k \leftarrow \min(\omega_{\kappa^H}, \omega^k)$
7: end for
8: return $\omega^k$
```

3.3.2 Line-search

In optimization problems, line-searches are used to find a step-size along a descent direction that sufficiently reduces the value of the objective function and its gradient. These conditions are known as the *Wolfe conditions*. When solving systems of nonlinear equations, line-searches improve the global convergence properties of Newton-based methods [42].

The line-search algorithm developed in this work is based on Modisette's method [54], and it relies on the optimization character of pseudo-transient continuation (Section 3.1.2). In short, both algorithms satisfy Armijo's rule [2] by back-tracking from an initial step-size until an update leads to a reduction in the 2-norm of the unsteady residual.

The difference between Modisette's method and ours is in computing an initial guess for the step size. While Modisette uses the ω^k computed with MPC as a starting step-size, our method only checks the physicality for elements whose update directions are deemed “unsafe”. This is assessed by computing the projection,

$$\theta = \Delta\mathbf{U}^k \cdot \frac{\partial \mathbb{P}}{\partial \mathbf{U}}\Big|_{\mathbf{U}^k}, \tag{3.23}$$

where $\mathbb{P}$ is the sum of all elemental penalties $\mathbb{P}_{\kappa^H}$. The elements with positive contributions to θ have update directions deemed “unsafe” since $\Delta\mathbf{U}^k$ will lead to an increase in $\mathbb{P}_{\kappa^H}$ for those elements.

The line-search algorithm is summarized below.

ALGORITHM 3.4 Line-search

```
1: $\omega_{\text{phys}} \leftarrow 1$                                        ▷ Initial guess for physical update
2: for all $\kappa^H$ with positive contribution to $\theta$ do
3:     Select limit points, $\mathbf{x}_m$
4:     Evaluate $\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$ and $\Delta\mathbf{u}^{H,p}(\mathbf{x}_m)|_{\kappa^H}$
5:     Call Algorithm 3.2                                        ▷ Limit physical update
6:     $\omega_{\text{phys}} \leftarrow \min(\omega_{\kappa^H}, \omega_{\text{phys}})$
7: end for
8: $\omega^k \leftarrow \omega_{\text{phys}}$
9: $\tilde{\mathbf{U}} \leftarrow \mathbf{U}^k + \omega^k \Delta\mathbf{U}^k$                                        ▷ Trial state vector
10: while $|\mathbf{R}_t(\tilde{\mathbf{U}})|_{L_2} > |\mathbf{R}(\mathbf{U}^k)|_{L_2}$ OR $\tilde{\mathbf{U}}$ is not physical do
11:     $\omega^k \leftarrow \frac{\omega^k}{2}$
12:     $\tilde{\mathbf{U}} \leftarrow \mathbf{U}^k + \omega^k \Delta\mathbf{U}^k$
13: end while
14: return $\omega^k$
```

Note that step 10 in Algorithm 3.4 checks if the trial state, $\tilde{\mathbf{U}}$, is physical. This check involves verifying if the physicality constraints (Section 2.3.1) are satisfied at the limit points. Also, we separate the 2-norm of residual into the individual conservation equations and require a drop in each of those norms. This reduces the effect of badly-scaled discrete systems that cause the residual norm to be dominated by the worst residual component which is also observed by Modisette [54].

3.3.2.1 Greedy algorithm

The physical update limiter in Algorithm 3.2 is heuristic and the line-search algorithm described above can prematurely exit with $\omega^k = \omega_{\text{phys}}$ while $\omega_{\text{phys}} < 1$. This can slow-down the convergence and increase the susceptibility to limit cycles. To address this possibility, a greedy algorithm is introduced. This algorithm amplifies ω^k while Armijo's rule is satisfied or until a full update is obtained, $\omega^k = 1$.

A safety check based on the projection in Eqn. 3.23 is performed before amplifying ω^k. Specifically, a negative projection, θ, indicates that is globally "safe" to proceed along $\Delta\mathbf{U}^k$. However, even with $\theta < 0$, amplifying ω^k may not be locally safe because certain elements may have positive contributions to θ and $\Delta\mathbf{U}^k$ may lead to non-physical states on those elements. The algorithm is summarized below.

ALGORITHM 3.5 Greedy algorithm

```
1: if $\omega^k = \omega_{\text{phys}}$ AND $\theta < 0$ then          ▷ $\theta$ is defined in Eqn. 3.23
2:     while $\omega^k \leq 1$ do
3:         $\omega^k \leftarrow \beta_\omega \cdot \omega^k$          ▷ For all cases, we use $\beta_\omega = 1.1$
4:         $\tilde{\mathbf{U}} \leftarrow \mathbf{U}^k + \omega^k \Delta \mathbf{U}^k$
5:         if $\tilde{\mathbf{U}}$ is not physical then
6:             $\omega^k \leftarrow \frac{\omega^k}{2}$
7:             return $\omega^k$
8:         end if
9:         if $|\mathbf{R}_t(\tilde{\mathbf{U}})|_{L_2} > |\mathbf{R}(\mathbf{U}^k)|_{L_2}$ then
10:            $\omega^k \leftarrow \frac{\omega^k}{\beta_\omega}$
11:            return $\omega^k$
12:        end if
13:    end while
14: end if
15: return $\omega^k$
```

3.4 Test-suite results

We now present results for a set of flow cases ranging from intermediate to difficult.[1] For each case, we combine the PTC and CPTC methods with each of the CFL evolution strategies and solution update algorithms. We identify each run with a sequence of 3 digits (Table 3.1) that respectively correspond to the continuation method, the solution update method, and the CFL evolution strategy.

For all cases, the residual convergence criterion is 9 orders of magnitude reduction compared to its initial norm. In order to compare the methods under equal-footing, the discretization and the GMRES parameters are the same for all runs in Table 3.1

[1]Level of difficulty assessed by the 1st International Workshop on High-Order CFD Methods.

Table 3.1: Summary of combinations of algorithms; PTC: pseudo-transient continuation; CPTC: constrained pseudo-transient continuation; MPC: maximum primitive change; LS: line-search; LS+G: line-search with greedy algorithm; SER: switched evolution relaxation; EXP: exponential progression; RDM: residual difference method; mRDM: monotonic residual difference method.

Run ID	$\Delta\mathbf{U}^k$ method	ω^k method	CFL^k method
1.1.1	PTC	MPC	EXP ($\beta = 1.2$)
1.1.2	PTC	MPC	SER
1.1.3	PTC	MPC	RDM ($\beta = 2.0$)
1.1.4	PTC	MPC	mRDM ($\beta = 2.0$)
1.2.1	PTC	LS	EXP ($\beta = 1.2$)
1.2.2	PTC	LS	SER
1.2.3	PTC	LS	RDM ($\beta = 2.0$)
1.2.4	PTC	LS	mRDM ($\beta = 2.0$)
1.3.1	PTC	LS+G	EXP ($\beta = 1.2$)
1.3.2	PTC	LS+G	SER
1.3.3	PTC	LS+G	RDM ($\beta = 2.0$)
1.3.4	PTC	LS+G	mRDM ($\beta = 2.0$)
2.1.1	CPTC	MPC	EXP ($\beta = 1.2$)
2.1.2	CPTC	MPC	SER
2.1.3	CPTC	MPC	RDM ($\beta = 2.0$)
2.1.4	CPTC	MPC	mRDM ($\beta = 2.0$)
2.2.1	CPTC	LS	EXP ($\beta = 1.2$)
2.2.2	CPTC	LS	SER
2.2.3	CPTC	LS	RDM ($\beta = 2.0$)
2.2.4	CPTC	LS	mRDM ($\beta = 2.0$)
2.3.1	CPTC	LS+G	EXP ($\beta = 1.2$)
2.3.2	CPTC	LS+G	SER
2.3.3	CPTC	LS+G	RDM ($\beta = 2.0$)
2.3.4	CPTC	LS+G	mRDM ($\beta = 2.0$)

and they only vary between flow cases. Similarly, a single mesh is generated and used for all the runs in each case. The meshes used in this chapter are not chosen specifically to produce accurate solutions. Instead, they are generated so as to have enough spatial resolution to reveal the flow features relevant for each case.

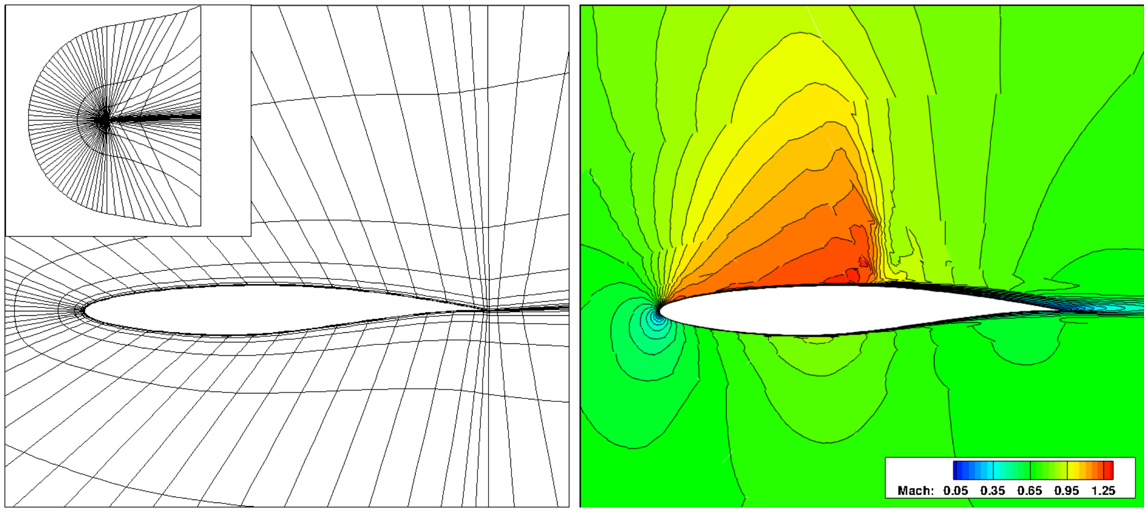
The cases are initialized with uniform flow under free-stream conditions and initial CFL$^0 = 1$. For each case, we present a color-coded table that assesses the success of all the runs. In these tables, green means the run converged, yellow means that the run reached the total wall time or maximum iterations without convergence and red means the run had either a non-physical error or the CFL is decreased below minimum ($\mathrm{CFL}_{\mathrm{min}} = 10^{-10}$ for all cases). The converged runs in each case are compared with respect to number of nonlinear iterations, number of GMRES iterations and total wall time.

3.4.1 RAE 2822 – $M_\infty = 0.734,\ Re = 6.5 \times 10^6,\ \alpha = 2.79°$

The first case is transonic, turbulent flow over the RAE 2822 airfoil. The scheme's polynomial order for this case is $p = 2$. The residual operator includes the shock-capturing term described in Chapter 2 and $\kappa_{\mathrm{BR2}} = 5$ for the viscous discretization. The SA equation is scaled by $\kappa_{\mathrm{SA}} = 1000$ and the free-stream turbulence level is 0.1%. The outer boundary of the domain is located 100 chord-lengths away from the airfoil and each edge of the mesh shown in Figure 3.1(a) is a quartic ($q = 4$) polynomial. The mesh has 990 quadrilaterals and the height of the first layer of elements off the wall is such that $y^+ \approx 5 \times 10^3$, based on a flat-plate correlation for the coefficient of friction.

Note the coarse resolution in Figure 3.1(b), this is because accuracy is not the primary goal of these cases, but rather to assess the ability of the solver to get a *zero-residual* solution.

The maximum number of iterations for each run is 10000 and the computational resources are 32 processors and a maximum wall time of 8 hours. Both PTC and CPTC converges with all solution update methods using either EXP or mRDM while none of the runs using SER and RDM converges for this case (Table 3.2). Those evolution strategies are non-monotonic as they can decrease or increase the CFL at each iteration. In this case, they are distracted by secondary transients and reduce the CFL, making the solver resolve those transients and thereby consume many iterations.



(a) Quartic mesh (990 elements).　　(b) Mach number contours.

Figure 3.1: RAE2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$, $p = 2$: mesh and Mach number contours.

The constrained version of PTC is not able to converge using MPC and mRDM (run 2.1.4) while the unconstrained version converges under the same conditions (run 1.1.4). As the penalization term increases in run 2.1.4, MPC keeps limiting the update and eventually it becomes small enough that the CFL decreases at a faster rate than the penalty increases.

Table 3.2: RAE 2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$, $p = 2$: success assessment of all runs.

	PTC			CPTC		
	MPC	LS	LS+G	MPC	LS	LS+G
EXP	🟩 run 1.1.1	🟩 run 1.2.1	🟩 run 1.3.1	🟩 run 2.1.1	🟩 run 2.2.1	🟩 run 2.3.1
SER	🟨 run 1.1.2	🟨 run 1.2.2	🟨 run 1.3.2	🟨 run 2.1.2	🟥 run 2.2.2	🟨 run 2.3.2
RDM	🟨 run 1.1.3	🟨 run 1.2.3	🟨 run 1.3.3	🟨 run 2.1.3	🟨 run 2.2.3	🟨 run 2.3.3
mRDM	🟩 run 1.1.4	🟩 run 1.2.4	🟩 run 1.3.4	🟥 run 2.1.4	🟩 run 2.2.4	🟩 run 2.3.4

Table 3.3 compares the converged runs for this case. The fastest run for this case is 1.2.1 followed by run 2.3.1. Within the runs using the greedy line-search (runs $x.3.x$ in Table 3.3), CPTC takes fewer nonlinear iterations and less time. This is due

to the greedy algorithm being triggered more often since the update direction at each nonlinear step accounts for the physicality constraints.

Table 3.3: RAE 2822 - $M_\infty = 0.734$, $Re = 6.5 \times 10^6$, $\alpha = 2.79°$, $p = 2$: metrics for converged runs normalized by run 1.1.1 – absolute values in parentheses.

Run ID	Nonlinear iterations	GMRES iterations	Wall time
1.1.1	1.000 (2184)	1.000 (41527)	1.000 ($8.886 \times 10^3 s$)
1.1.4	1.652	1.006	1.410
1.2.1	0.421	0.574	0.582
1.2.4	1.841	1.307	2.140
1.3.1	0.243	0.291	0.682
1.3.4	1.282	0.921	1.833
2.1.1	0.595	0.598	0.685
2.2.1	0.851	0.899	1.352
2.2.4	1.404	1.254	2.148
2.3.1	0.150	0.280	0.673
2.3.4	0.603	0.543	1.502

3.4.2 NACA 0012 – $M_\infty = 0.8,\ Re = 6.5 \times 10^6,\ \alpha = 0°$

The second test case is also transonic, turbulent flow over an airfoil. The scheme's polynomial order is $p = 2$ and $\kappa_{\text{BR2}} = 5$ for the viscous discretization. The SA model is scaled by $\kappa_{\text{SA}} = 100$ and the free-stream turbulence level is 0.01%. Since this is a non-lifting case, the outer boundary of the domain is located at 30 chord-lengths from the airfoil and the mesh is composed of 1740 quadrilaterals with quartic polynomial edges (Figure 3.2). The height of the first layer of elements is such that $y^+ \approx 2 \times 10^3$, based one a flat-plate correlation for the friction at the wall. We emphasize that the purpose of meshes in this chapter is not to allow for accurate solutions but to simply reveal the relevant flow features. The computational resources for this case are 40 processors for 8 hours of wall-time and the maximum number of iterations is 10000.

We present two sets of runs for this case. The first set uses the shock-capturing term described in Chapter 2. Figure 3.3(a) shows the Mach number contours ob-

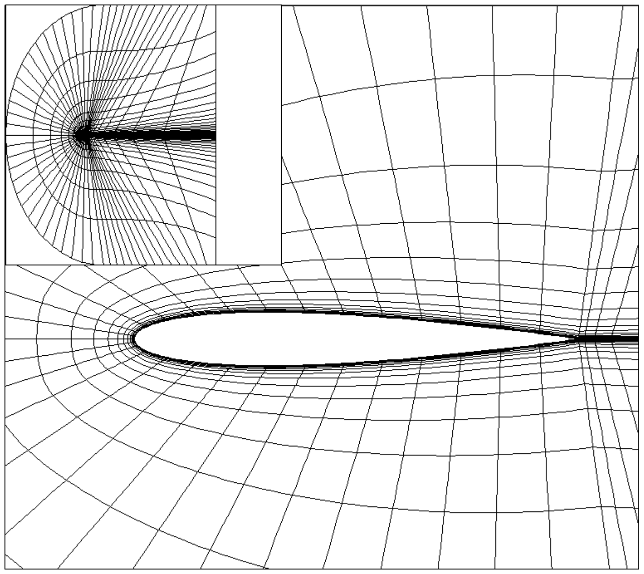
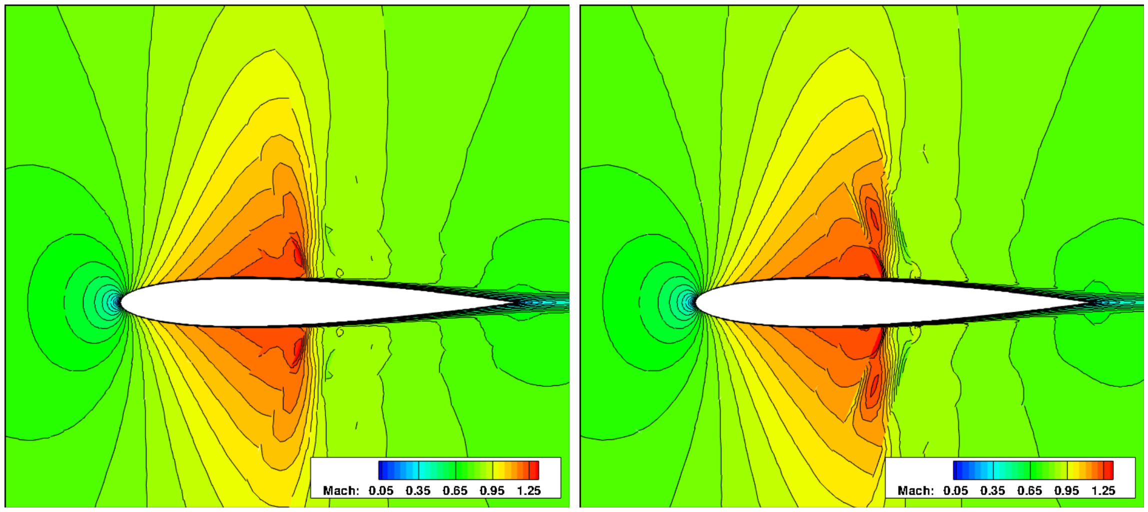


Figure 3.2: NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$: quartic mesh (1740 elements).

tained using that term. The second set of runs does not use shock capturing and Figure 3.3(b) shows the Mach contours for this condition. Note the oscillatory behavior of the solution in the vicinity of the shocks. Clearly, it is not ideal to simulate flows with shocks without shock-capturing terms. However, this exercise is useful to assess the robustness of the solution advancement methods.



(a) Mach number contours with shock-capturing. (b) Mach number contours without shock-capturing.

Figure 3.3: NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$: Mach number contours.

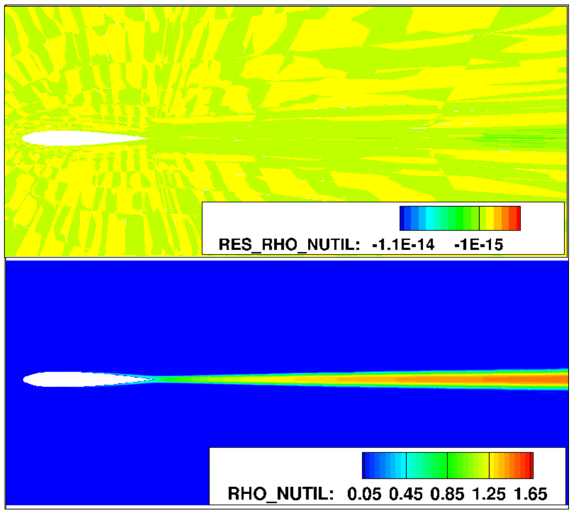
Table 3.4 shows the success of the runs using the shock-capturing term, with which most of the runs converge. Interestingly, the constrained solver with the greedy line-search and SER (run 2.3.2) exceeds the maximum allotted time while the unconstrained solver with the same update method and CFL strategy (run 1.3.2) converges within the time limit.

Table 3.4: NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ with shock-capturing: success assessment of all runs

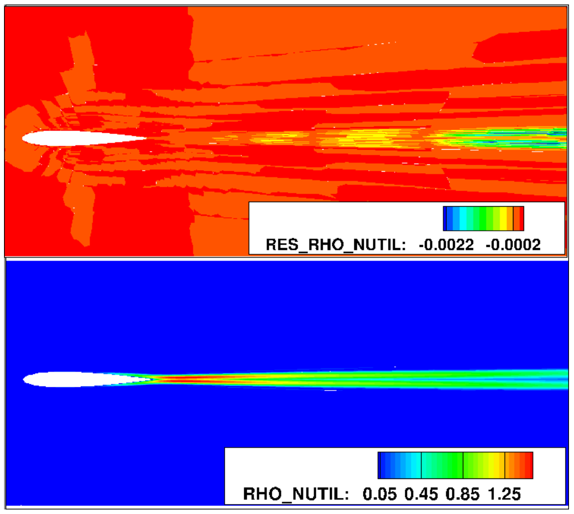
	PTC			CPTC		
	MPC	LS	LS+G	MPC	LS	LS+G
EXP	🟩 run 1.1.1	🟩 run 1.2.1	🟩 run 1.3.1	🟩 run 2.1.1	🟩 run 2.2.1	🟩 run 2.3.1
SER	🟩 run 1.1.2	🟩 run 1.2.2	🟩 run 1.3.2	🟩 run 2.1.2	🟩 run 2.2.2	🟨 run 2.3.2
RDM	🟨 run 1.1.3	🟨 run 1.2.3	🟨 run 1.3.3	🟨 run 2.1.3	🟨 run 2.2.3	🟨 run 2.3.3
mRDM	🟩 run 1.1.4	🟩 run 1.2.4	🟩 run 1.3.4	🟩 run 2.1.4	🟩 run 2.2.4	🟩 run 2.3.4

Figure 3.4 compares the final states for the runs using the greedy line-search and

SER (runs x.3.2). Note the under-development of the wake and the residual for the turbulence equation for run 2.3.2 (Figure 3.4(b)). During that run, the CFL is reduced from its initial value after the initial transient and remains below 1 for most of the iterations and, thus, not reaching Newton convergence. The same issue occurs with all runs using RDM for evolving the CFL.



(a) Contours for $\rho\tilde{\nu}$ and its residual for run 1.3.2.



(b) Contours for $\rho\tilde{\nu}$ and its residual for run 2.3.2 (not converged).

Figure 3.4: NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ with shock-capturing: comparison of final states for runs x.3.2.

Within the successful runs using shock-capturing (Table 3.5), the quickest are 2.3.1 and 2.3.4, as they take practically the same time to converge. Similarly to the previous case, the number of iterations for run 2.3.1 is smaller than its unconstrained counterpart, run 1.3.1, due to the greedy algorithm being triggered more often with the constrained solver.

We now discuss the runs without the shock-capturing scheme. Table 3.6 shows the success of all these runs. Again, the monotonic CFL strategies perform better than the non-monotonic strategies. The constrained solver is successful with all the solution update strategies using both EXP and mRDM, while the unconstrained method using MPC and EXP finishes with the CFL below minimum.

Table 3.5: NACA 0012 - $M_\infty = 0.8,\ Re = 6.5 \times 10^6,\ \alpha = 0°,\ p = 2$ with shock-capturing: metrics for converged runs normalized by run 1.1.4 – absolute values in parentheses.

Run ID	Nonlinear iterations	GMRES iterations	Wall time (seconds)
1.1.1	1.000 (1133)	1.000 (37374)	1.000 (7.301×10^3)
1.1.2	4.185	2.501	3.130
1.1.4	1.109	1.169	1.129
1.2.1	0.733	0.958	0.875
1.2.2	4.162	2.488	3.137
1.2.4	1.237	1.171	1.191
1.3.1	0.387	0.653	0.589
1.3.2	3.996	2.420	3.062
1.3.4	0.646	0.695	0.724
2.1.1	0.945	0.931	0.934
2.1.2	2.178	1.662	1.843
2.1.4	1.139	1.172	1.171
2.2.1	0.606	0.830	0.739
2.2.2	2.143	1.646	1.846
2.2.4	0.954	1.054	1.010
2.3.1	0.317	0.633	0.554
2.3.4	0.400	0.585	0.554

Table 3.7 compares the runs that converge without the shock-capturing scheme. Note that the runs using the greedy algorithm are the fastest as they save many nonlinear iterations and they also compute fewer matrix-vector products involved in the GMRES iterations. Amongst the converged runs using the greedy algorithm, the runs using the unconstrained solver (runs 1.3.x) are slightly faster than their constrained counterpart (runs 2.3.x) despite the greedy algorithm being triggered more often. This is because the constrained runs use more GMRES iterations.

3.4.3 MDA 30p30n – $M_\infty = 0.2,\ Re = 9 \times 10^6,\ \alpha = 16°$

This test case is subsonic, turbulent flow over a high-lift, multi-element airfoil. The scheme's approximation order is $p = 1$ and $\kappa_{\mathrm{BR2}} = 20$, which is much higher than the theoretical linear stability limit. However, runs with lower κ_{BR2} turned out

Table 3.6: NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ without shock-capturing: success assessment of all runs

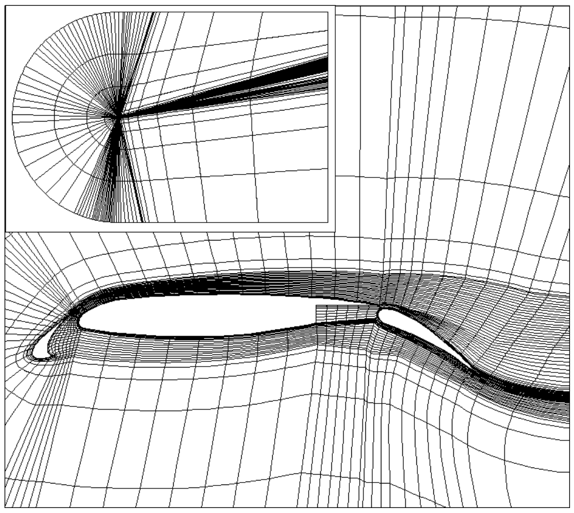
	PTC			CPTC		
	MPC	LS	LS+G	MPC	LS	LS+G
EXP	🟥 run 1.1.1	🟩 run 1.2.1	🟩 run 1.3.1	🟩 run 2.1.1	🟩 run 2.2.1	🟩 run 2.3.1
SER	🟥 run 1.1.2	🟥 run 1.2.2	🟥 run 1.3.2	🟨 run 2.1.2	🟨 run 2.2.2	🟨 run 2.3.2
RDM	🟥 run 1.1.3	🟥 run 1.2.3	🟥 run 1.3.3	🟥 run 2.1.3	🟥 run 2.2.3	🟥 run 2.3.3
mRDM	🟩 run 1.1.4	🟩 run 1.2.4	🟩 run 1.3.4	🟩 run 2.1.4	🟩 run 2.2.4	🟩 run 2.3.4

Table 3.7: NACA 0012 - $M_\infty = 0.8$, $Re = 6.5 \times 10^6$, $\alpha = 0°$, $p = 2$ without shock-capturing: metrics for converged runs normalized by run 1.1.4 – absolute values in parentheses.

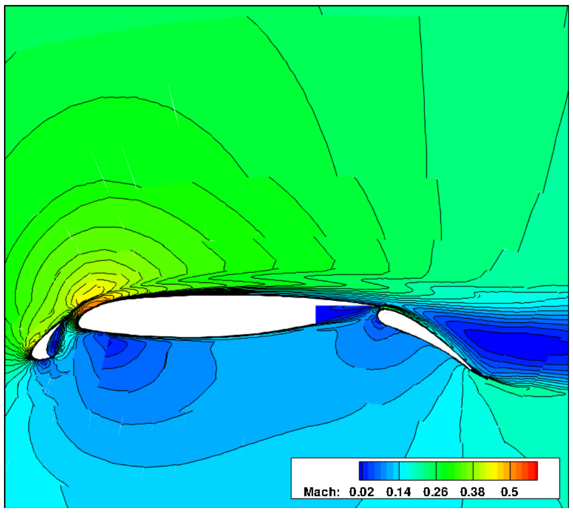
Run ID	Nonlinear iterations	GMRES iterations	Wall time
1.1.4	1.000 (3707)	1.000 (34671)	1.000 ($1.347 \times 10^4 s$)
1.2.1	0.281	0.456	0.327
1.2.4	0.454	0.557	0.477
1.3.1	0.0968	0.316	0.170
1.3.4	0.179	0.301	0.229
2.1.1	0.401	0.440	0.410
2.1.4	0.308	0.537	0.367
2.2.1	0.250	0.474	0.309
2.2.4	0.386	0.542	0.422
2.3.1	0.127	0.335	0.200
2.3.4	0.136	0.529	0.275

to be very challenging and results for lower κ_{BR2} require a sequence of runs with increasing Reynolds numbers. This Reynolds number sequencing would make the robustness assessment more difficult. Furthermore, in an adaptive framework, the effect of κ_{BR2} in the final output values is minimal since the solution jumps decrease as the mesh is refined.

The discrete equation for the turbulence model is scaled by $\kappa_{\mathrm{SA}} = 1000$ and the free-stream level of turbulence is 1%. Figure 3.5 shows the mesh and Mach number contours for this flow. A linear multi-block mesh is generated with the objective of having a good alignment of the cells with the wake. Patches of elements from the linear mesh were then agglomerated to generate the quartic mesh shown in Figure 3.5(a). The agglomerated mesh has 4070 elements and the off-wall spacing is such that $y^+ \approx 3 \times 10^3$, based on a flat-plate correlation.



(a) Quartic mesh (4070 elements).



(b) Mach number contours.

Figure 3.5: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$: mesh and Mach number contours.

For each run, 8 wall-clock hours of 16 processors are allotted and the maximum number of iterations is 10000. Table 3.8 shows the success of all the runs for this flow. Note that the rate of success is considerably smaller than in the previous cases. The non-monotonic CFL strategies, SER and RDM, reduce the CFL in an attempt to resolve the transients, a process which takes many iterations due to the stiffness of this problem.

Both PTC and CPTC are able to obtain converged solutions with the update methods based on the line-search. However, the constrained solver with the greedy line-search (run 2.3.4) reaches the maximum number of iterations before convergence while the unconstrained counterpart succeeds in converging the residual within the iteration limit.

It is worth emphasizing the challenging nature of this case. The blunt flap cove

Table 3.8: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$, $p = 1$: success assessment of all runs

	PTC			CPTC		
	MPC	LS	LS+G	MPC	LS	LS+G
EXP	🟨 run 1.1.1	🟨 run 1.2.1	🟩 run 1.3.1	🟨 run 2.1.1	🟨 run 2.2.1	🟩 run 2.3.1
SER	🟨 run 1.1.2	🟨 run 1.2.2	🟨 run 1.3.2	🟨 run 2.1.2	🟨 run 2.2.2	🟨 run 2.3.2
RDM	🟨 run 1.1.3	🟨 run 1.2.3	🟨 run 1.3.3	🟨 run 2.1.3	🟨 run 2.2.3	🟨 run 2.3.3
mRDM	🟨 run 1.1.4	🟨 run 1.2.4	🟩 run 1.3.4	🟨 run 2.1.4	🟩 run 2.2.4	🟨 run 2.3.4

and the sharp turn behind the slat are geometric features that cause flow separation and hence the flow exhibits strong initial transients when initialized with uniform free-stream conditions. Clearly, such a flow initialization strategy is not ideal. However, we believe it is a useful exercise to test the solver under such demanding conditions.

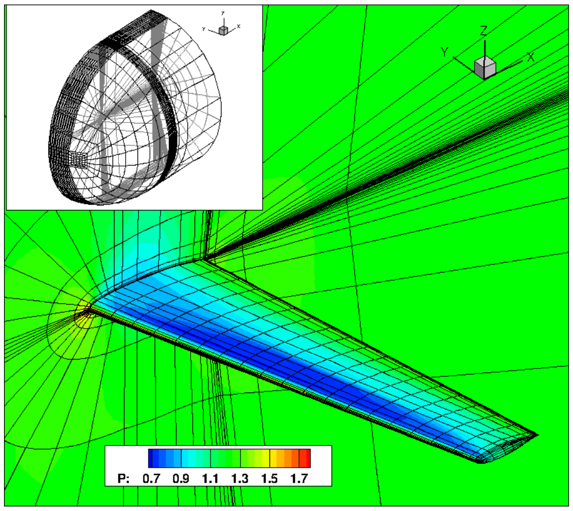
Amongst the converged runs (Table 3.9), the constrained solver with line-search and mRDM (run 2.2.4) is the most efficient in this case. Note that the wall-time is strongly related to the number of GMRES iterations. This is expected, as most of the computational time in an implicit solver is spent computing matrix-vector products involved in the linear solves.

Table 3.9: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$, $p = 1$: metrics for converged runs normalized by run 1.3.1 – absolute values in parentheses.

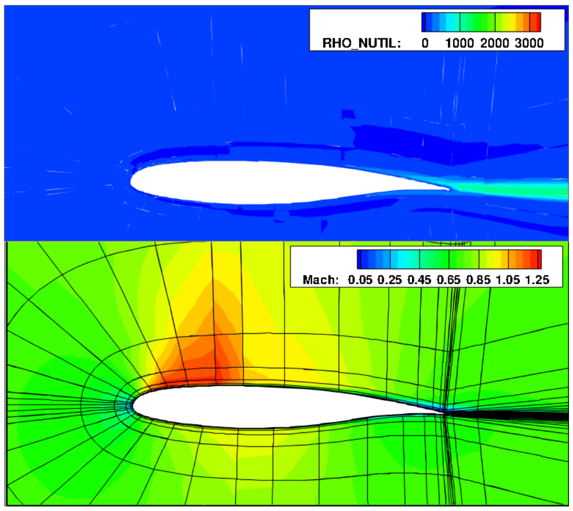
Run ID	Nonlinear iterations	GMRES iterations	Wall time (seconds)
1.3.1	1.000 (1412)	1.000 (608770)	1.000 (5.085×10^3)
1.3.4	4.750	0.935	1.005
2.2.4	5.135	1.346	1.059
2.3.1	1.161	0.881	0.920

3.4.4 DPW 3 Wing 1 – $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5°$

This case is transonic, turbulent flow over the baseline wing from the Third Drag Prediction Workshop. A linear multi-block (C-topology) mesh is generated following the workshop's guidelines [26]. The linear elements were agglomerated to generate a cubic ($q = 3$) mesh with 29310 elements (Figure 3.6(a)). The spacing of the linear mesh is such that the agglomerated mesh presents $y^+ \approx 1$, based on a friction coefficient correlation for a flat plate. Figure 3.6(b) shows the contours of Mach number and SA working variable at a mid-span slice of the flow domain.



(a) Mesh and static pressure contours. Upper left corner shows a view of the full computational domain (29310 elements).



(b) Mach number and $\rho\tilde{\nu}$ contours.

Figure 3.6: DPW 3 Wing 1 – $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5°$, $p = 1$: mesh and Mach number contours.

We show in Chapter 2 that higher values of κ_{SA} improve the performance of the solver, however the results presented in this section were obtained prior to that realization, and, for this reason, the SA discrete equation is not rescaled ($\kappa_{\text{SA}} = 1$). The free-stream level of turbulence is 0.1%.

The scheme's approximation order is $p = 1$, $\kappa_{\text{BR2}} = 2$, and the shock-capturing term is included in the residual operator. Each run uses 804 processors for a maximum of 5 hours and the maximum number of iterations is 4000. Under these limits, only the constrained solver converges as shown in Table 3.10. Note that the greedy line-search fails in most of the runs. This is due to excessively large updates in the beginning of the calculation that eventually lead to non-physical states at the trailing edge of the wing. The greedy algorithm is prematurely triggered because the inner-product in Eqn. 3.23 is dominated by negative contributions. In the case of the constrained solver, the local penalization is not enough to make the projection, θ, globally positive.

Table 3.10: DPW 3 Wing 1 – $M_\infty = 0.76,\ Re = 5 \times 10^6,\ \alpha = 0.5°,\ p = 1$: success assessment of all runs

	PTC			CPTC		
	MPC	LS	LS+G	MPC	LS	LS+G
EXP	🟥 run 1.1.1	🟨 run 1.2.1	🟥 run 1.3.1	🟩 run 2.1.1	🟩 run 2.2.1	🟥 run 2.3.1
SER	🟥 run 1.1.2	🟥 run 1.2.2	🟥 run 1.3.2	🟨 run 2.1.2	🟨 run 2.2.2	🟥 run 2.3.2
RDM	🟨 run 1.1.3	🟨 run 1.2.3	🟥 run 1.3.3	🟥 run 2.1.3	🟥 run 2.2.3	🟥 run 2.3.3
mRDM	🟨 run 1.1.4	🟨 run 1.2.4	🟨 run 1.3.4	🟩 run 2.1.4	🟩 run 2.2.4	🟥 run 2.3.4

The constrained solver using MPC and mRDM (run 2.1.4) takes the shortest time and fewest number of GMRES iterations to converge (Table 3.11). As discussed in the previous cases, the number of GMRES iterations strongly affects the run time. However, other aspects can significantly affect the total run time. For example, runs 2.1.1 and 2.2.4 have significantly different run times even though they take virtually the same number of GMRES iterations. The extra time in run 2.2.4 is mostly due to the additional residual evaluations involved in the line-search algorithm that compensate for the fewer nonlinear iterations.

Table 3.11: DPW 3 Wing 1 – $M_\infty = 0.76,\ Re = 5 \times 10^6,\ \alpha = 0.5°,\ p = 1$: metrics for converged runs normalized by run 2.1.1 – absolute values in parentheses.

Run ID	Nonlinear iterations	GMRES iterations	Wall time (seconds)
2.1.1	1.000 (1255)	1.000 (68253)	1.000 ($5.694 \times 10^3 s$)
2.1.4	0.897	0.935	0.845
2.2.1	0.880	0.944	1.021
2.2.4	0.751	1.002	1.127

Figure 3.7 shows the residual and penalization histories for the runs listed in Table 3.11. Note that runs 2.1.1 and 2.2.1 track similar penalization paths with the

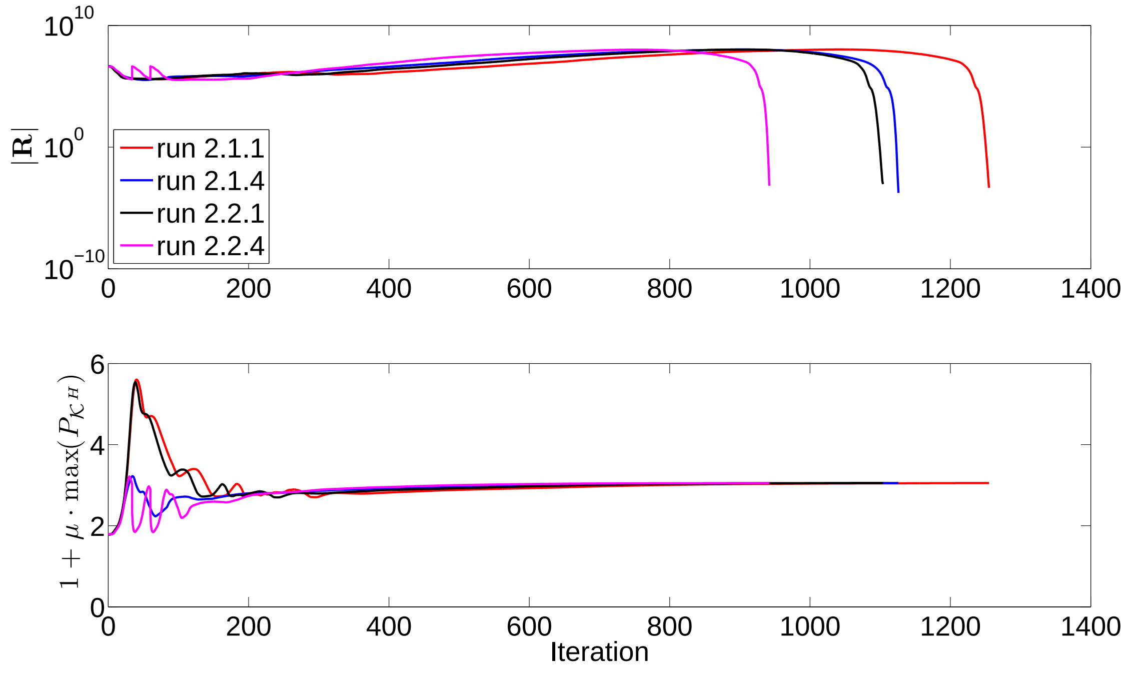
exception that the run with the line-search (2.2.1) takes fewer non-linear iterations. The runs using mRDM also follow similar paths in the first ~ 40 iterations but the line-search (2.2.4) returns a few "no-updates" (marked by the sudden drops in CFL Figure 3.7(b)), that is, when $\omega^k < \omega_{\min}$ and the state is reset to the last safe update.

3.4.5 Remarks about the test-suite results

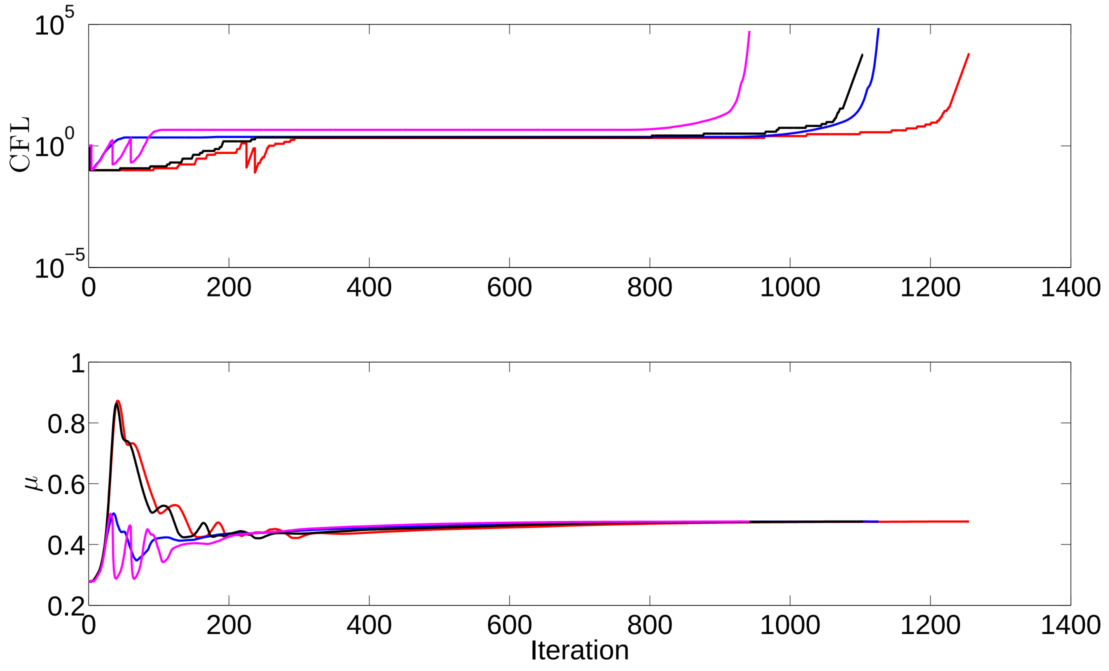
The results presented here are sensitive to various parameters in the discretization, in the linear solver, and, more importantly, in the solution update methods. Specifically, the β parameter in the CFL strategies significantly affects the performance of the solver. The value of this parameter is somewhat heuristic and the rationale for deciding on its value is that β closer to 1 is less aggressive and it tends to be more robust. However, this is not always the case, as less aggressive CFL strategies are more likely to trap the solver in transients that may not be physical. An example of such a problem occurs when the flow is initialized with uniform free-stream flow next to a wall and, if a flow expansion, *e.g.* at a blunt trailing edge, is solved time-accurately, cavitation may occur. The penalization approach reduces the sensitivity to these non-physical transients but it is not a *bullet-proof* approach and better flow initialization methods are certainly helpful.

The combination of the constrained solver with line-search and the mRDM CFL strategy converges all the cases presented here and it also helps in the even more challenging case of transonic, turbulent flow over NASA's CRM[2] wing-body geometry (presented in Chapter 4). Also, the line-search algorithms make use of the penalization idea and using them with PTC is very effective in obtaining solutions for most of the cases presented in this thesis.

[2]Common Research Model



(a) Residual norm and penalization histories.



(b) CFL and penalty factor histories.

Figure 3.7: DPW 3 Wing 1 – $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5°$, $p = 1$: residual norm and penalization histories for converged runs.

CHAPTER 4

Error Estimation and Mesh Adaptation

This chapter presents a method for concurrent mesh and polynomial-order adaptation with the objective of direct minimization of output error using a selection process for choosing the optimal refinement option from a discrete set of choices that includes directional spatial resolution and approximation order increment. The scheme is geared towards compressible viscous aerodynamic flows, in which various solution features make certain refinement options more efficient compared to others. No attempt is made, however, to measure the solution anisotropy or smoothness directly or to incorporate it into the scheme. Rather, mesh anisotropy and approximation order distribution arise naturally from the optimization of a merit function that incorporates both an output sensitivity and a measure of the computational cost of solving on the new mesh. The method is applied to output-based adaptive simulations of the Euler equations and of the laminar and Reynolds-averaged compressible Navier-Stokes equations on body-fitted meshes in two and three dimensions.

4.1 Output error estimation

Output-based error estimation techniques identify all areas of the domain that are important for the accurate prediction of an engineering output. The resulting estimates properly account for error propagation effects that are inherent to hyperbolic problems, and they can be used to ascribe confidence levels to outputs or to drive adaptation. A key component of output error estimation is the solution of an adjoint equation for the output of interest. In a continuous setting, an adjoint, $\boldsymbol{\psi} \in \mathcal{V}$, is a Green's function that relates residual source perturbations to a scalar output of interest, $J(\mathbf{u})$, where $\mathbf{u} \in \mathcal{V}$ denotes the state, and where $\mathcal{V}$ is an appropriate function space. Specifically, given a variational formulation of a partial differential equation: determine $\mathbf{u} \in \mathcal{V}$ such that

$$\mathbb{R}(\mathbf{u}, \mathbf{w}) = 0, \qquad \forall \mathbf{w} \in \mathcal{V}, \tag{4.1}$$

the adjoint $\boldsymbol{\psi} \in \mathcal{V}$ is the sensitivity of J to an infinitesimal source term added to the left-hand side of the original PDE. $\boldsymbol{\psi}$ satisfies a linear equation,

$$\mathbb{R}'[\mathbf{u}](\mathbf{w}, \boldsymbol{\psi}) + J'[\mathbf{u}](\mathbf{w}) = 0, \qquad \forall \mathbf{w} \in \mathcal{V}, \tag{4.2}$$

where the primes denote Fréchét linearization with respect to the arguments in square brackets. Details on the derivation of the adjoint equation can be found in many sources, including the review in Ref. [22]. Specifically, in the present work we employ the discrete adjoint method, in which the system is derived systematically from the discretized primal system [31, 51].

An adjoint solution can be used to estimate the numerical error in the corresponding output of interest. The resulting adjoint-weighted residual method is based on the observation that a solution $\mathbf{u}^{H,p}$ in a finite-dimensional approximation space $\mathcal{V}^{H,p}$, polynomials of order p on a subdivision T^H of the domain into elements κ_H, will generally not satisfy the original PDE. The adjoint $\boldsymbol{\psi} \in \mathcal{V}$ translates the residual perturbation to an output perturbation via,

$$\delta J = J(\mathbf{u}^{H,p}) - J(\mathbf{u}) \approx -\mathbb{R}(\mathbf{u}^{H,p}, \boldsymbol{\psi}). \tag{4.3}$$

This expression is based on a linear analysis, and hence for nonlinear problems and finite-size perturbations, the result is approximate.

Although the continuous solution $\mathbf{u}$ is not required directly, the continuous adjoint $\boldsymbol{\psi}$ must be approximated to make the error estimate in Eqn. 4.3 computable. In practice, $\boldsymbol{\psi}^{h,p^+}$ is solved approximately or exactly on a finer finite-dimensional space $\mathcal{V}^{h,p^+} \supset \mathcal{V}^{H,p}$ [66, 5, 72]. This finer space can be obtained either through mesh subdivision or approximation order increase [50, 34, 59] – denoted here by changes in the superscript H and p, respectively.

The adjoint-weighted residual evaluation in Eqn. 4.3 can be localized to yield an adaptive indicator consisting of the relative contribution of each element to the total output error. In this work, the finer space is obtained by approximation order increment, $\mathcal{V}^{H,p+1} \supset \mathcal{V}^{H,p}$, and $\boldsymbol{\psi}^{H,p+1}$ is approximated by injecting $\boldsymbol{\psi}^{H,p}$ into $\mathcal{V}^{H,p+1}$ and applying 5 – unless otherwise noted – element block-Jacobi smoothing iterations. The output perturbation in Eqn. 4.3 is approximated as

$$\delta J \approx - \sum_{\kappa^H \in T^H} \mathbb{R}_{\kappa^H}(\mathbf{I}_{H,p}^{H,p+1}(\mathbf{u}^{H,p}), \boldsymbol{\psi}^{H,p+1} - \mathbf{I}_{H,p}^{H,p+1}(\boldsymbol{\psi}^{H,p})), \tag{4.4}$$

where $\mathbf{I}_{H,p}^{H,p+1}(\cdot)$ is an injection operator from p to $p+1$ in the coarse mesh T^H, and $\mathbb{R}_{\kappa^H}$ corresponds to the elemental residual as defined in Eqn. 2.22. Note, the difference between the coarse-space and fine-space adjoints is not strictly necessary due to Galerkin orthogonality [22]. However, when the primal residual is not fully-converged to machine precision levels the use of the adjoint perturbation gives better error estimates. Equation 4.4 expresses the output error in terms of contributions from each coarse element. A common approach for obtaining an adaptive indicator is to take the absolute value of the elemental contribution in Eqn. 4.4 [81, 7, 35, 30, 5, 14],

$$\eta_{\kappa^H} = \left| \mathbb{R}_{\kappa^H}(\mathbf{I}_{H,p}^{H,p+1}(\mathbf{u}^{H,p}), \boldsymbol{\psi}^{H,p+1} - \mathbf{I}_{H,p}^{H,p+1}(\boldsymbol{\psi}^{H,p})) \right|. \tag{4.5}$$

With systems of equations, indicators are computed separately for each equation and summed together. Due to the absolute values, the sum of the indicators, $\sum_{\kappa^H} \eta_{\kappa^H}$, is greater or equal to the original output error estimate. However, it is not a bound on the actual error because of the approximations made in the derivation.

4.2 Mesh adaptation mechanics

The elemental adaptive indicator, η_{κ^H}, drives a fixed-fraction hanging-node adaptation strategy. In this strategy, which was chosen for simplicity and predictability of the adaptive algorithm, a certain fraction, f^{adapt}, of the elements with the largest values of η_{κ^H} is marked for refinement. Marked elements are refined according to discrete options which correspond to subdividing the element in different directions or increasing the approximation order. For quadrilaterals, the discrete options are: x-refinement, y-refinement, xy-refinement, and p-increment, as depicted in Figure 4.1. Note, x and y refer to reference-space coordinates of elements that can be arbitrarily oriented and curved in physical space. Also, the subelements created through refinement inherit the approximation order from the original element. In three dimensions a hexahedron can be refined in eight ways: three single-plane cuts, three double-plane cuts, isotropic refinement, and p increment.

h-refinement is performed in an element's reference space by employing the coarse element's reference-to-global coordinate mapping in calculating the refined element's geometry node coordinates. The refined elements inherit the same geometry approximation order and quadrature rules as the parent coarse element. As a result, there is no loss of element quality when a nonlinear mapping is used to fit the element to

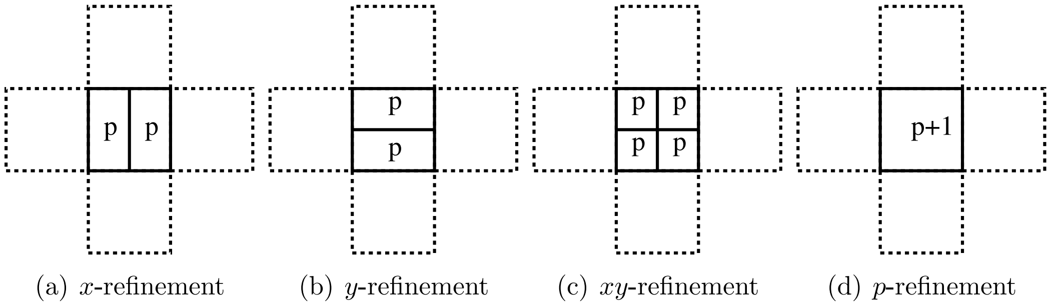


Figure 4.1: Quadrilateral refinement options. The dashed lines indicate the neighbors of the refined element.

a curved geometry. Therefore, curved elements near a boundary can be efficiently refined to capture boundary layers in viscous flow. For simplicity of implementation, the initial mesh is assumed to capture the geometry sufficiently well, through a high enough order of geometry interpolation on curved boundaries, such that no additional geometry information is used throughout the refinements. That is, refinement of elements on the geometry boundary does not change the geometry. We note that for highly-anisotropic meshes, curved elements may be required away from the boundary, and for simplicity we use meshes with curved elements throughout the domain.

Note that elements created in a hanging-node refinement can be marked for h-refinement again in subsequent adaptation iterations. In this case, neighbors will be cut to keep one level of refinement difference between adjacent cells. This is illustrated in Figure 4.2.

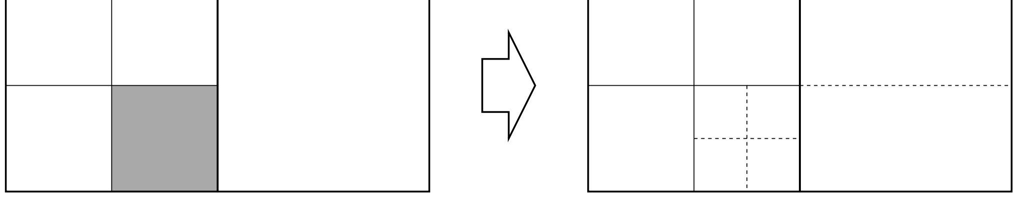


Figure 4.2: Hanging-node adaptation for a quadrilateral mesh, with a maximum of one level of refinement separating two elements. The shaded element on the left is marked for isotropic refinement, and the dashed lines on the right indicate the additional new edges formed.

4.3 Merit function

The choice of a particular refinement option is made locally in each element flagged for refinement. This choice is made by defining a merit function $m(i)$ that ranks each available refinement option i. This function is defined as

$$m(i) = \frac{b(i)}{c(i)} \ , \tag{4.6}$$

where b and c respectively correspond to measures of the benefit and the computational cost of the refinement option indexed by i. These measures depend on the method used for solving the flow equations and they should be tailored for each specific solver. We define them further in this section in the context of the applications presented in this thesis.

During calculation of the merit function, local mesh and data structures are created, one for each element, that include the flagged element and its first-level neighbors along with the corresponding primal and adjoint states. In these local structures, the central element is refined in turn according to each of the discrete options. On the refined local mesh, the merit function is computed and the refinement option with the largest value of $m(i)$ is chosen.

The method for selecting a refinement option presented in this thesis is similar to that presented by Houston *et al.* [38] for quadrilateral meshes. These authors employ a heuristic that consists of the sum of the subelement error indicators computed for each refinement option, and a ratio, θ, of the maximum to minimum sum to make the decision of adapting isotropically or in one direction. Anisotropy is only deemed important when θ is larger than a user-prescribed threshold, for which a value of 3 is found to work well. The method proposed in the present work is an extension to *hp*-adaptation of the method proposed in Ref. [14] and it differs from Houston's approach in that it employs the merit function in Eqn. 4.6 instead of a user-prescribed parameter. The cost and benefit functions that define $m(i)$ are described next.

4.3.1 Cost

We consider two measures of computational cost. The first measure is solution storage that is proportional to the number of degrees of freedom in the discrete state vector. For tractability, we consider only the degrees of freedom pertinent to the flagged element κ^H,

$$c_{\text{DOF}}(i) = \sum_{\kappa^h \in \kappa^H} (p_{\kappa^h}(i) + 1)^{\text{dim}}, \tag{4.7}$$

where $\kappa^h \in \kappa^H$ denotes the subelements embedded in the original element selected for refinement and $p_{\kappa^h}(i)$ is the element's approximation order after the refinement as depicted in Figure 4.1. Note that $p_{\kappa^h} = p_{\kappa^H}$ for h-refinement while the number of embedded elements changes. Conversely, $p_{\kappa^h} = p_{\kappa^H} + 1$ for p-refinement and there is only one embedded element, *i.e.* the original element. Also, we are not considering the rank of the conserved state vector, N_s, because it is a constant throughout the mesh. It is worth emphasizing that this measure of cost is insensitive to the type of time integration used to solve Eqn. 2.28, and therefore it is a generic measure of cost.

The second measure of computational cost incorporates information about the time integration method. In this work, most of the computational time is spent solving the linear system in Eqn. 3.2 using the GMRES algorithm. In a sparse structure such as in Eqn. 3.2, we approximate the number of floating point operations in applying GMRES by the number of non-zero entries in the Jacobian matrix. Based on this observation, we define the second measure of cost as:

$$c_{\text{NZ}}(i) = \sum_{\kappa^h \in \kappa^H} \left\{ (p_{\kappa^h}(i) + 1)^{2\cdot\text{dim}} + \sum_{\partial\kappa^h \setminus \partial D} \left[(p_{\kappa^h}(i) + 1) \cdot (p^-_{\kappa^h}(i) + 1) \right]^{\text{dim}} \right\}, \tag{4.8}$$

where $p^-_{\kappa^h}$ denotes the approximation order of the neighboring element across face $\partial\kappa^h$, which must not be part of the boundary of the domain, ∂D. The first term in Eqn. 4.8 accounts for the self-blocks of the residual Jacobian matrix corresponding to each of the subelements. The second term corresponds to the dependence of the subelements' residual on the neighboring states. The cost function does not take into account possible sparsity within the blocks of the Jacobian matrix, as such sparsity is not taken into account by the solver. Note that c_{NZ} is more sensitive to the number of spatial dimensions than c_{DOF}.

4.3.2 Benefit

The benefit $b(i)$ is a measure of how much improvement in the prediction of an output results from refinement option i. Evidently, the definition of benefit is not unique and it may be tailored for different applications and solution methods. However, it is desirable that such a definition is tractable and computationally inexpensive.

In an output-based mesh adaptation cycle, the steady-state residual is driven to zero at each step. Therefore, mesh modification on the element level can be interpreted as a local residual perturbation. Since an adjoint solution represents the sensitivity of an output with respect to a residual perturbation, we define our benefit function as:

$$b(i) = \sum_{\kappa^h \in \kappa^H} |\mathrm{R}_{\kappa^h}(\mathrm{U}_{sb}\mathrm{T}_{bd}(i))_j||\Psi_{sb}\mathrm{T}_{bd}(i)\mathrm{V}_{sdj}|, \tag{4.9}$$

where $\mathrm{R}_{\kappa^h}(\cdot)_j$ is a discrete residual component in the embedded element, d indexes the basis functions, $\mathrm{T}_{bd}(i)$ is a matrix that transfers the discrete solution U_{sb} to the local meshes for each refinement i, Ψ_{sb} is the discrete adjoint solution and V_{sdj} is an unrolling tensor as defined in Eqn. 2.29. Note that the adjoint variables act as positive weights for each of the perturbations.

The definition in Eqn. 4.9 relies on the following observations:

- At each step of the adaptation cycle, a discrete primal solution is found so that the residual vector is machine-zero. Therefore, the benefit as defined in Eqn. 4.9 is also machine-zero if computed before refining the central element.
- In the limit of the discrete solution representing the exact solution to machine precision, the result of Eqn. 4.9 will be of the order of machine precision for any refinement option.
- The refinement option with the largest $b(i)$ is expected to be the option that produces the largest change in the output of interest.

Note that Eqn. 4.9 is inexpensive to compute since only a residual calculation in the local mesh and data structures is required for each refinement option. Also, this framework is different than a residual-based decision because the values of the discrete adjoint provide information on the distribution of output sensitivity.

4.4 *hp*-adaptation results

In this section, we assess the performance of our *hp*-adaptation framework using the cost measures c_{DOF} and c_{NZ}. The performance is measured in terms of number of degrees of freedom and CPU time. In the output-based adaptation methods, the time stamps include the solution of both the primal and adjoint solves, while for the uniform refinements only the primal solve time is included.

We limit the maximum approximation order to $p_{\max} = 3$ for the two-dimensional cases to improve the performance and robustness of the adaptive method. Specifically, c_{DOF} under-estimates the computational expense of p-increment so that the adaptive algorithm prioritizes increasing p over anisotropically h-refining. Even though c_{NZ} more accurately estimates the computational expense of the different refinement options, $p_{\max}$ is the same for both cost measures to establish a basis of comparison between the cost measures. In addition, the artificial viscosity shock capturing approach used in some of the cases is not perfect and suffers from increased dissipation at high p that pollutes the error estimates and interferes with adaptation; limiting p is a simple and effective fix to this problem. In three dimensions, the effect of spatial dimensionality on c_{DOF} reduces the impact of its under-estimation of cost on the adaptive process, resulting in fewer elements targeted for p-refinement. For this reason, the p-orders are not limited in the three-dimensional problems.

4.4.1 NACA 0012

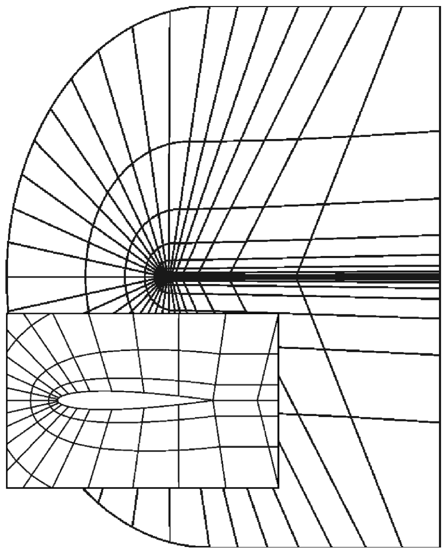
The first set of results we present consists of the NACA 0012 airfoil under three flow conditions:

1. $M_\infty = 0.5, \alpha = 2.0^o$, inviscid;
2. $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$;
3. $M_\infty = 0.8, \alpha = 1.25^o$, inviscid.

The airfoil was modified to have zero-thickness at the trailing edge and its geometry is given by the following formula:

$$y = \pm 0.6(0.2969\sqrt{x} - 0.1260 - 0.3516x^2 + 0.2843x^3 - 0.1036x^4), \tag{4.10}$$

where x and y are coordinates normalized by the chord length. The initial meshes (Figure 4.3) for these cases are composed of quartic ($q = 4$) quadrilaterals and the outer boundary is located 50 chord-lengths away from the airfoil. Our output of interest is the near-field drag. We compare the performance of the hp-adaptation routine using c_{DOF} and c_{NZ} against uniform h and p refinements. At each step of the adaptive procedure, $f^{\mathrm{adapt}} = 10\%$ of elements with the largest error indicators, η_{κ^H}, is selected for refinement. All runs used 4 Nehalem 8-core nodes from the Nyx cluster at the University of Michigan.


(a) Initial mesh for inviscid flows over the NACA 0012.

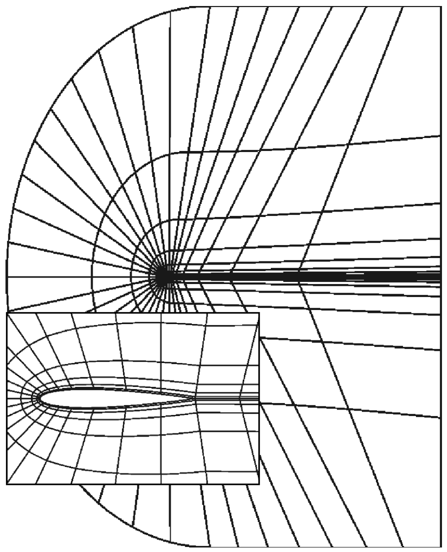
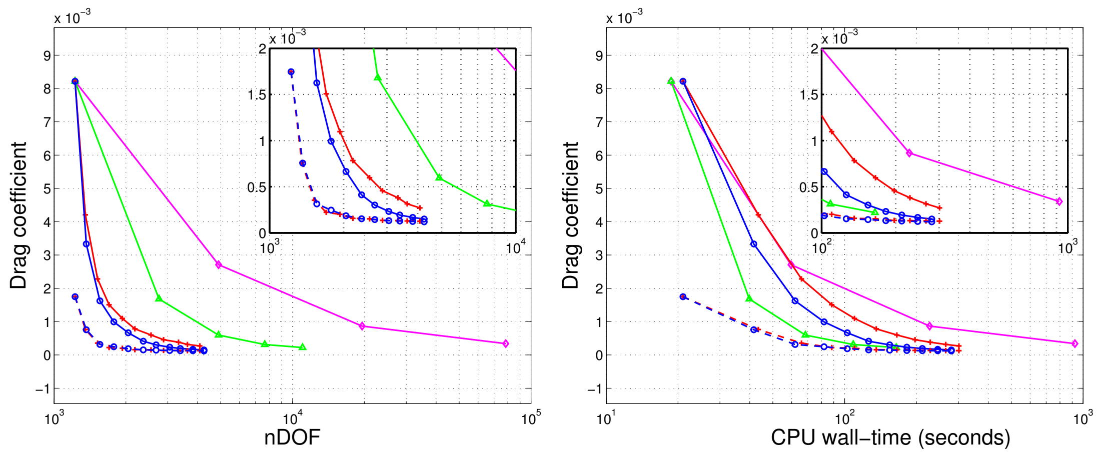

(b) Initial mesh for viscous flow over the NACA 0012.

Figure 4.3: Initial quartic ($q = 4$) meshes for the NACA 0012 cases.

4.4.1.1 $M_\infty = 0.5, \alpha = 2.0^o$, inviscid

The first test case is inviscid flow at $M_\infty = 0.5$ and $\alpha = 2.0^o$. In the hypothetical case of the outer boundary of the computational domain being located infinitely far, the drag measured on the surface of the airfoil should converge to zero since there are neither viscous effects nor shocks in the flow. However, when the far-field is located at a finite distance the near-field drag converges to a finite value [17] as plotted in Figure 4.4. When corrected by the error estimates, the performance of the *hp*-adaptation routine is not very sensitive, in this case, to the different cost measures. However, the uncorrected drag values (solid lines) converge faster when the *hp*-adaptation uses c_{DOF}, both in degrees of freedom (Figure 4.4(a)) and in CPU time (Figure 4.4(b). Note that uniform *p*-refinement performs very well in this case since the flow is smooth and the cost of uniformly high-order solutions is moderate in two dimensions. Also, the computational load imbalance in parallel runs adversely affects the *hp*-runs due to non-uniformity of *p*-orders.



(a) Drag coefficient evolution with respect to degrees of freedom

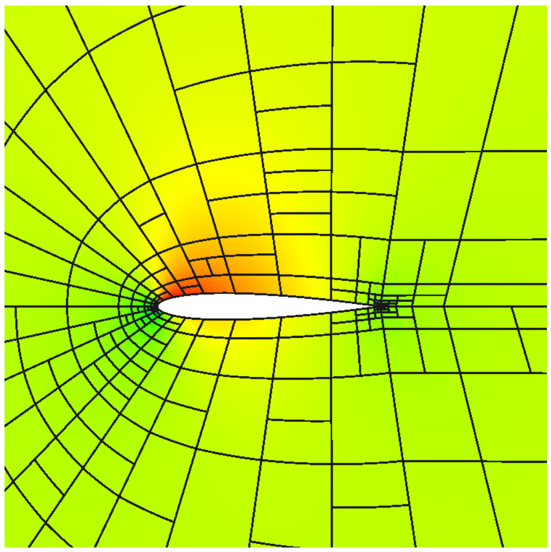
(b) Drag coefficient evolution with respect to CPU time

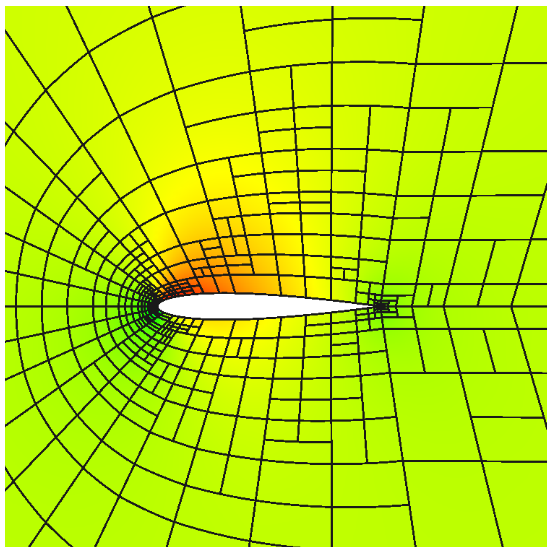
Figure 4.4: NACA 0012, $M_\infty = 0.5, \alpha = 2.0^o$, inviscid: drag coefficient convergence; $\diamond$: uniform h-refinement; $\triangle$: uniform p-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}. The dashed lines correspond to the drag values corrected with the error estimate.

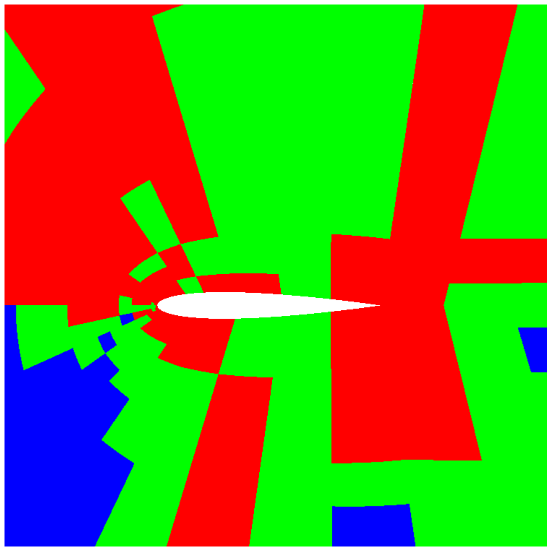
Figure 4.5 shows the final *hp*-adapted meshes. In both c_{DOF} and c_{NZ} cases, isotropic *h*-refinement and order increment are active at the trailing edge in order to accurately capture the total-pressure recovery that is important to elimination of drag through spurious entropy generation [60, 17]. Note that the *hp*-mesh obtained with c_{DOF} has a larger area at higher approximation order ($p > 1$), while the adaptive method with c_{NZ} shows significantly more mesh subdivisions.

4.4.1.2 $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$

The second two-dimensional test case is subsonic viscous flow at $M_\infty = 0.5, \alpha = 1.0^o$ and $Re = 5 \times 10^3$. Similarly to the previous case, we compare the *hp*-adaptation framework using both cost measures against uniform *h* and *p* refinements. Figure 4.6(a) shows the drag coefficient convergence in terms of number of degrees of freedom. While both *hp*-adaptation runs present similar convergence histories for the corrected


(a) 10th Mesh with Mach contours for c_{DOF}.


(b) 10th Mesh with Mach contours for c_{NZ}.


(c) 10th p-order distribution for c_{DOF}; blue indicates $p = 1$; red indicates $p = 3$.

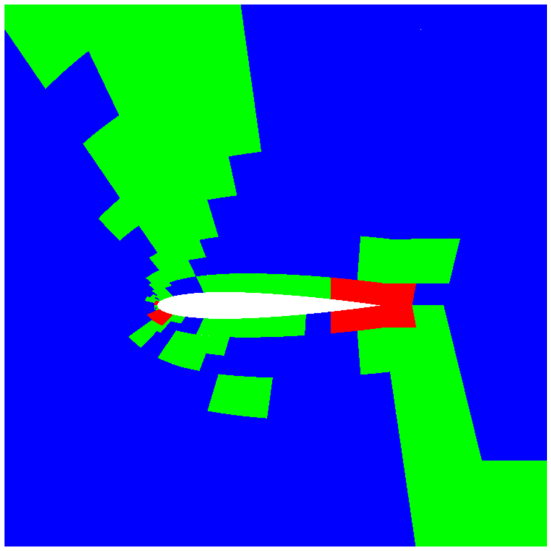
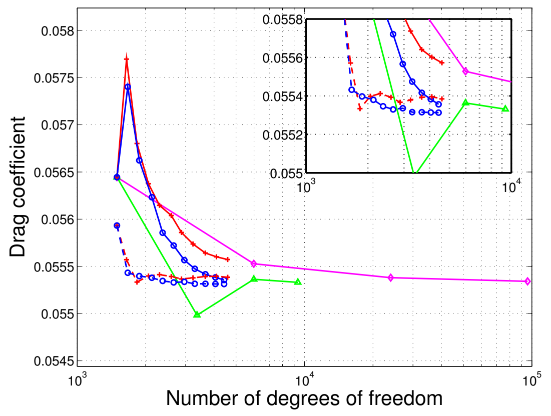
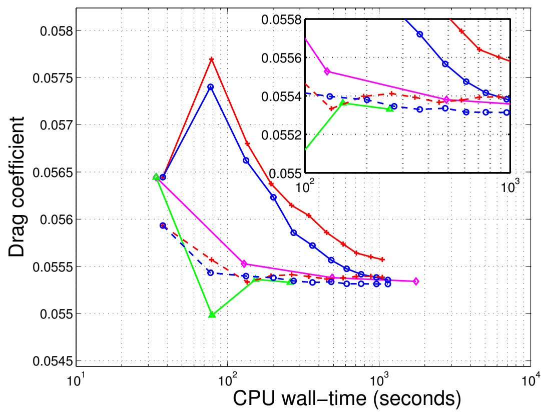

(d) 10th p-order distribution for c_{NZ}; blue indicates $p = 1$; red indicates $p = 3$.

Figure 4.5: NACA 0012, $M_\infty = 0.5, \alpha = 2.0^o$, inviscid: hp-adapted meshes for drag.

output (dashed lines), the uncorrected drag values (solid lines) converge faster with c_{DOF} than when c_{NZ} is employed. Additionally, the hp-adaptation runs converge the corrected output with significantly fewer degrees of freedom than the uniform refinements although uniform p-refinement performs reasonably well for this smooth problem. This observation is also valid in terms of CPU time (Figure 4.6(b)), however the savings are smaller.



(a) Drag coefficient evolution with respect to degrees of freedom



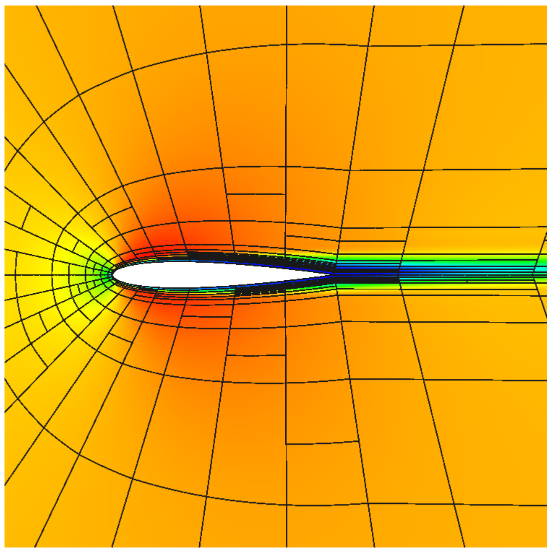
(b) Drag coefficient evolution with respect to CPU time

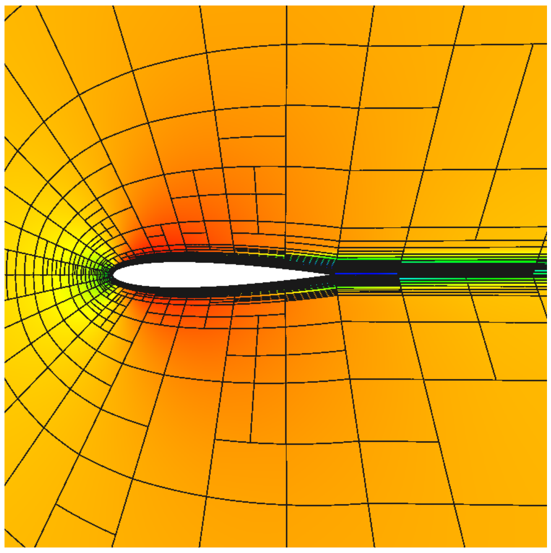
Figure 4.6: NACA 0012, $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$: drag coefficient convergence; $\diamond$: uniform h-refinement; $\triangle$: uniform p-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}. The dashed lines correspond to the drag values corrected with the error estimate.

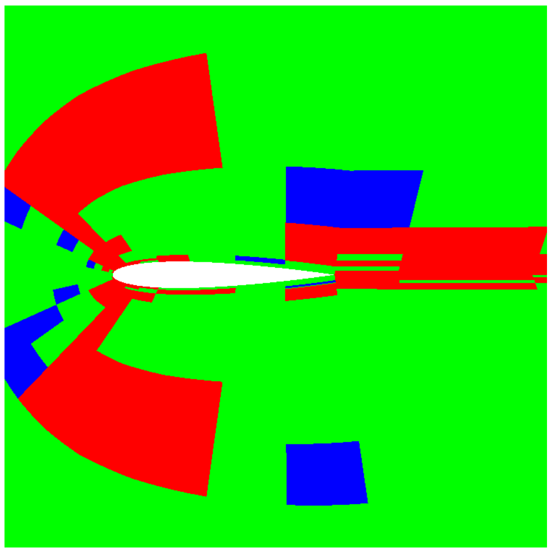
As observed in the previous case, the adaptive scheme produces a larger area of the domain with higher order cells when c_{DOF} is used to measure the cost of the refinement options (Figure 4.7(c)). In contrast, the adaptive algorithm, when using c_{NZ}, chooses p-increment mostly in the wake region combined with anisotropic h-refinement as seen in Figures 4.7(d) and 4.7(b) respectively.

4.4.1.3 $M_\infty = 0.8, \alpha = 1.25^o$, inviscid

The final two-dimensional case we present is inviscid transonic flow over the NACA 0012 geometry. For shock-capturing, we use an element-wise constant artificial viscosity approach [64]. The ability to robustly resolve discontinuities like shocks in high-order discretizations is a current challenge in CFD. Due to robustness problems, the error estimates can suffer and their convergence may not be reliable. Specifically, noise in the error estimates arises from dual-inconsistency of the shock indicator and the use of hierarchical mesh refinement on a fixed background topology. For this reason, we consider in this case only the pure drag output, that is, the output without the correction by the error estimate.


(a) 10th Mesh with Mach contours for c_{DOF}.


(b) 10th Mesh with Mach contours for c_{NZ}.


(c) 10th p-order distribution for c_{DOF}; blue indicates $p = 1$; red indicates $p = 3$.

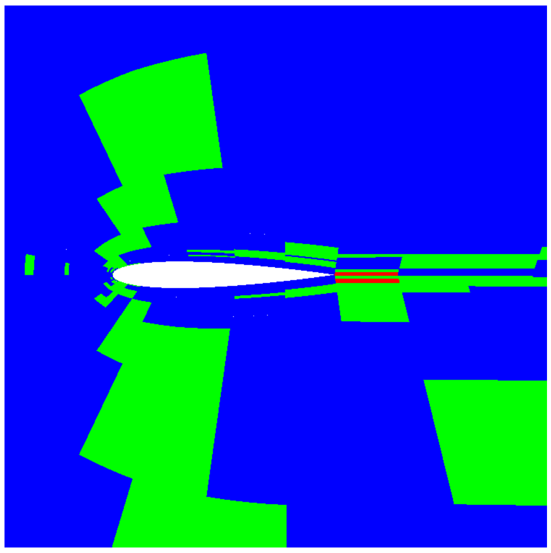
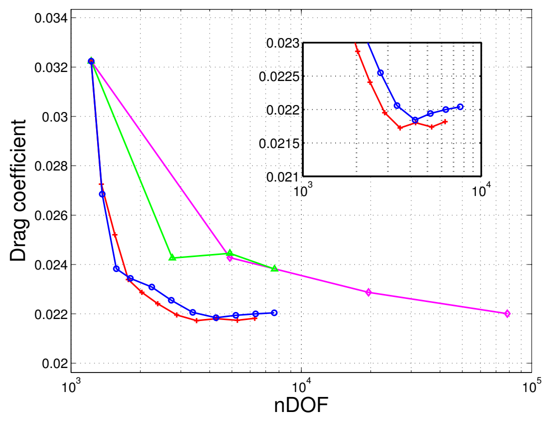

(d) 10th p-order distribution for c_{NZ}; blue indicates $p = 1$; red indicates $p = 3$.

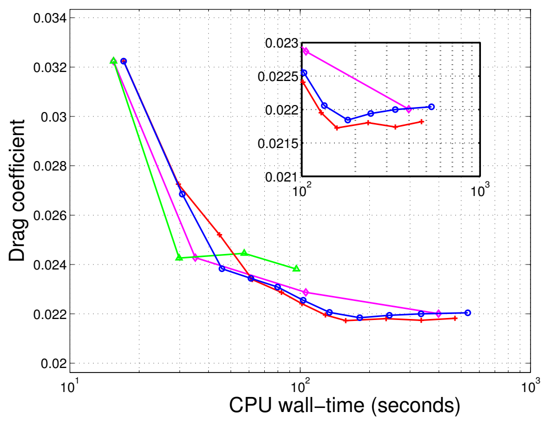
Figure 4.7: NACA 0012, $M_\infty = 0.5, \alpha = 1.0^o, Re = 5 \times 10^3$: hp-adapted meshes for drag.

We limit the maximum approximation order to $p_{\max} = 3$ for the reason described in the beginning of this section. Also, we found that the piecewise-constant artificial viscosity combined with a resolution-based discontinuity indicator used for shock capturing adds excessive dissipation for $p \geq 4$. The excessive dissipation leads to benefit-function values that do not necessarily favor the best refinement direction.

Figure 4.8(a) shows the drag convergence with respect to degrees of freedom. Note that the adaptation converges slightly faster with c_{NZ} than with c_{DOF} and in comparison with uniform h-refinement, the hp methods use approximately an order of magnitude fewer degrees of freedom. The savings in CPU time are smaller but still significant, since the hp methods took about half the time to achieve drag convergence to within 2 counts of drag (Figure 4.8(b)).



(a) Drag coefficient evolution with respect to degrees of freedom.



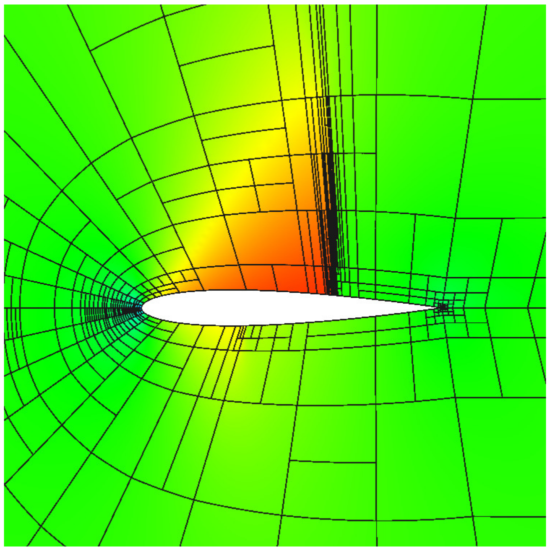
(b) Drag coefficient evolution with respect to CPU time.

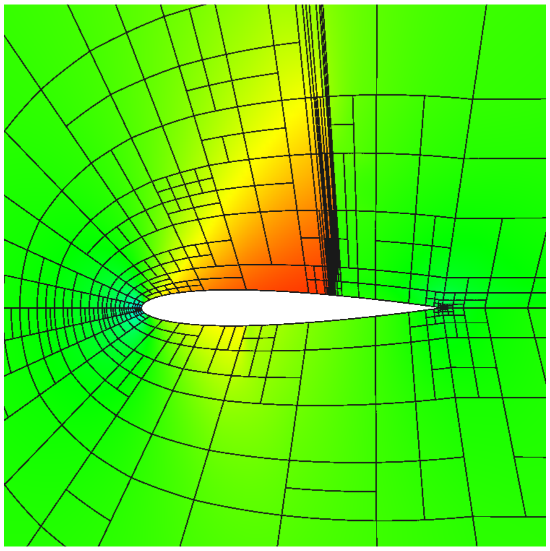
Figure 4.8: NACA 0012, $M_\infty = 0.8, \alpha = 1.25^o$, inviscid: drag coefficient convergence; $\diamond$: uniform h-refinement; $\triangle$: uniform p-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}.

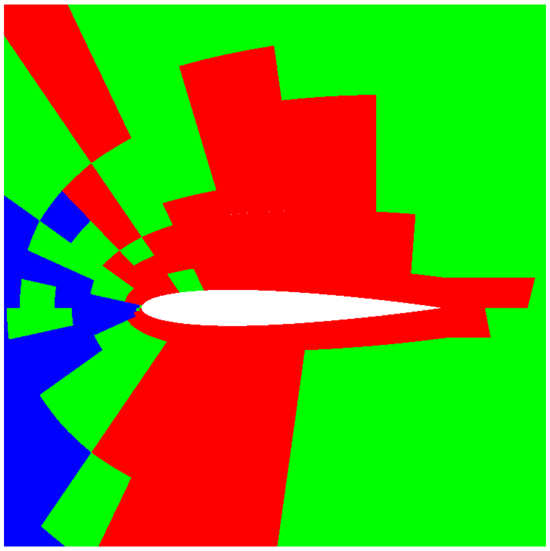
The meshes for c_{DOF} and c_{NZ} are shown in Figure 4.9. Note that both methods choose anisotropic h-refinement in combination with $p-$increment in the vicinity of the strong shock on the upper surface and the weaker shock on bottom surface of the airfoil. It is also notable that both isotropic h-refinement and higher-order cells are present at the trailing edge in order to accurately represent the pressure recovery.

4.4.2 NLR Delta wing, $M_\infty = 0.3, \alpha = 12.5^o, Re = 4 \times 10^3$

The second case we present is laminar flow over the NLR delta wing at a high angle of attack [67, 45]. The vortical structure of this flow presents both sharp and smooth features that can benefit from hp-adaptation. We consider the hp-adaptation


(a) 10th Mesh with Mach contours for c_{DOF}.


(b) 10th Mesh with Mach contours for c_{NZ}.


(c) 10th p-order distribution for c_{DOF}; blue indicates $p = 1$; red indicates $p = 3$.

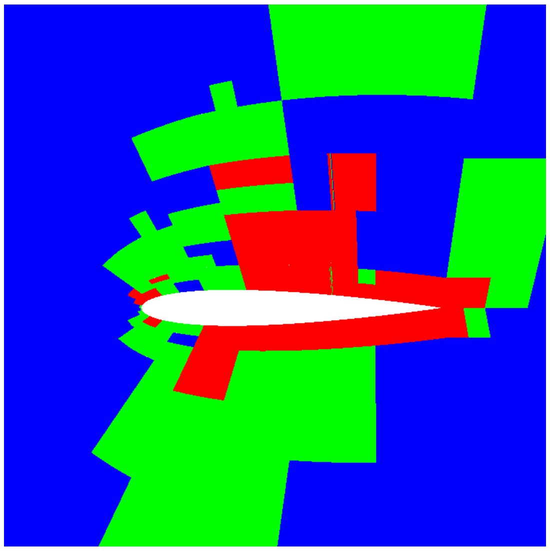
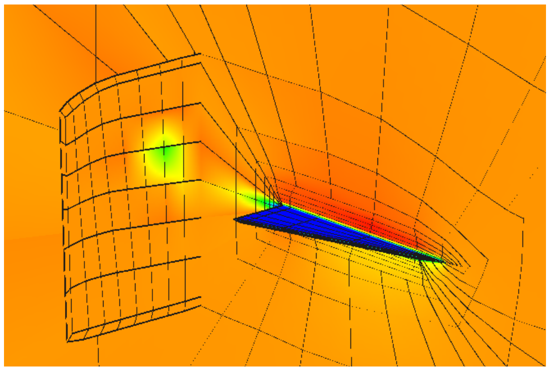
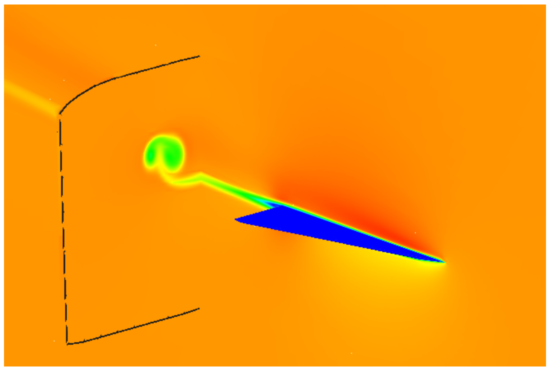

(d) 10th p-order distribution for c_{NZ}; blue indicates $p = 1$; red indicates $p = 3$.

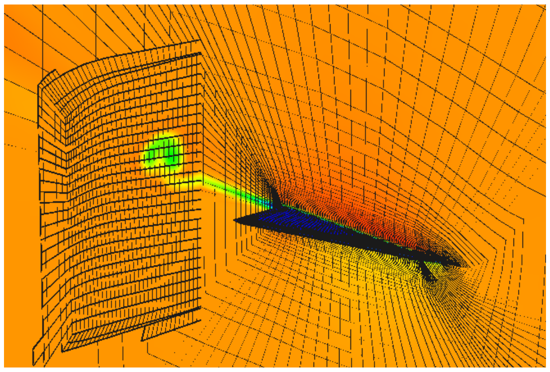
Figure 4.9: NACA 0012, $M_\infty = 0.8, \alpha = 1.25^o$, inviscid: hp-adapted meshes for drag.

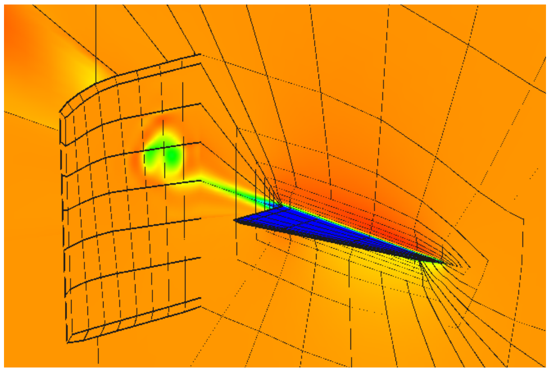
framework using both c_{DOF} and c_{NZ} cost measures. At each step of these strategies, $f^{\text{adapt}} = 10\%$ of the elements in the mesh was adapted starting from a $p = 1$ solution (Figure 4.10(a)) on the second-level linear mesh generated by NLR as part of the ADIGMA [1] project. The output of interest is drag and we present uniform h-refinement and uniform p-refinement along with the output-adapted meshes.

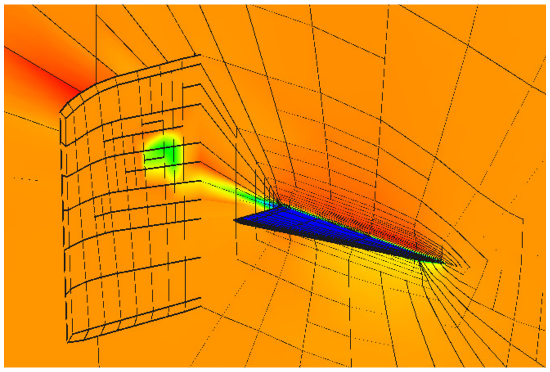
All calculations used 50 Harpertown 8-core nodes from NASA's Pleaides super-


(a) Initial mesh with Mach number contours computed with $p = 1$ (3264 linear elements).


(b) Mach contours on a finer mesh with $p = 2$.


(c) Two levels of uniform h-refinement with $p = 1$ (208896 linear elements).


(d) Two levels of uniform p-refinement $p = 1 \rightarrow 3$ (3264 linear elements).


(e) 4th drag-adapted mesh using c_{DOF} (5778 elements).

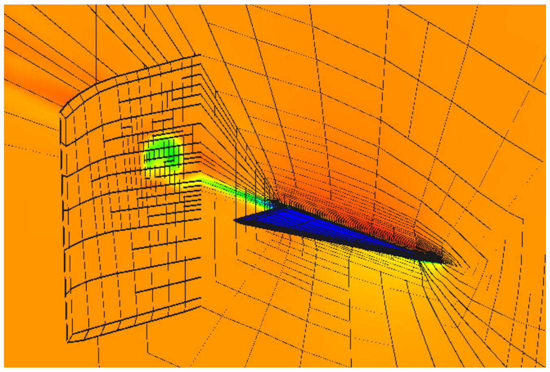
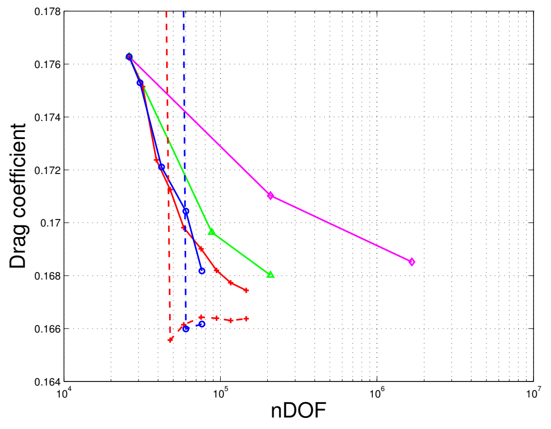
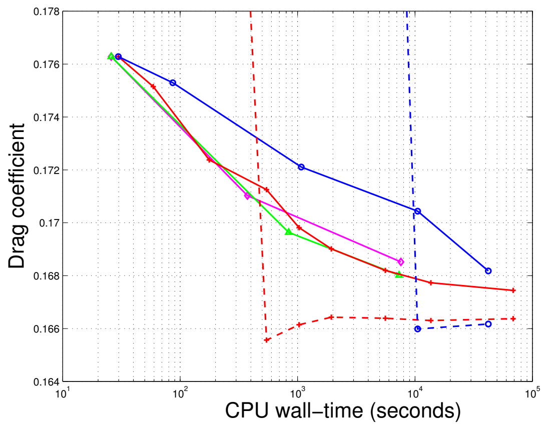

(f) 8th drag-adapted mesh using c_{NZ} (16260 elements).

Figure 4.10: NLR Delta wing, $M_\infty = 0.3, \alpha = 12.5^o, Re = 4\times10^3$: Initial and adapted meshes with Mach number contours.

computer and we fixed the maximum CPU time for each of the mesh-improvement strategies. The reason for fixing a CPU budget is to simulate a condition in which a practitioner has a certain amount of time to provide an answer to an engineering problem. We then assess the quality of the answer that each of the mesh-improvement strategies obtained within that CPU-time budget. The last converged solutions of all strategies are shown in Figure 4.10 along with the initial and reference solutions in Figure 4.10(a) and Figure 4.10(b) respectively.



(a) Drag coefficient evolution with respect to degrees of freedom



(b) Drag coefficient evolution with respect to CPU time

Figure 4.11: NLR Delta, $M_\infty = 0.3, \alpha = 12.5^o, Re = 4 \times 10^3$: drag coefficient convergence; $\diamond$: uniform h-refinement; $\triangle$: uniform p-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}. The drag values with the error estimates for the first three adaptive steps are out of the range of the vertical axis. The dashed lines correspond to the drag values corrected with the error estimate.

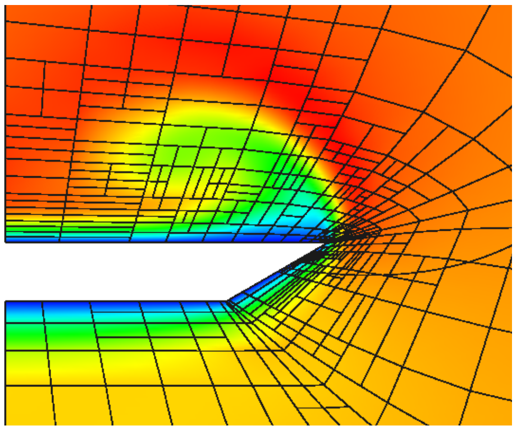
Figure 4.11 shows the evolution of the drag coefficient in terms of degrees of freedom and CPU-time. The dashed lines in that figure are the outputs of the adjoint-based adaptation methods corrected by their corresponding error estimates. Note that in the initial adaptation steps, the error estimates are very large. However, they converge rapidly after 2 to 3 adaptation steps. Even though the performance of c_{NZ} and of c_{DOF} are similar in terms of degrees of freedom (Figure 4.11(a)), in terms of computational time (Figure 4.11(b)), c_{NZ} shows a clear advantage over c_{DOF} since higher p-orders are more expensive than c_{DOF} estimates. Also, the uniform refinement strategies perform remarkably well in CPU-time. This is due to an adequate off-wall spacing and overall high-quality of the initial mesh which makes the error close to equally-distributed and, in this case, the large increments in number of degrees of freedom provides an advantage since no time is spent solving on intermediate meshes and the homogeneity of p-order helps the balance of computational load on the processors.

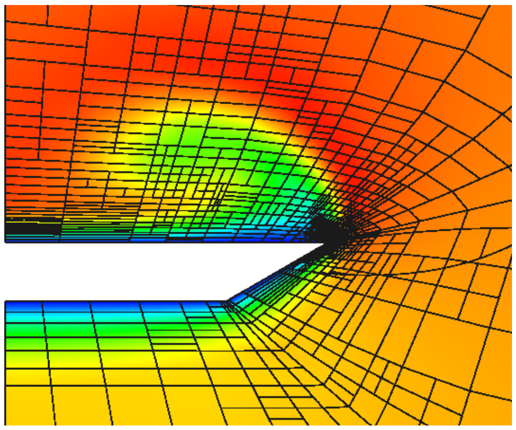
Table 4.1 lists the frequency of choice of each of the refinement options for the different cost measures. When using c_{DOF}, the adaptation method chooses p-refinement significantly more often compared to when c_{NZ} is employed. This larger frequency of p-refinement with c_{DOF} is due to under-estimation of the computational cost of solving higher-order discretizations and it causes the slower output convergence in terms of CPU time shown in Figure 4.11(b). Note that both methods tend to choose p-refinement more frequently in the later adaptation steps.

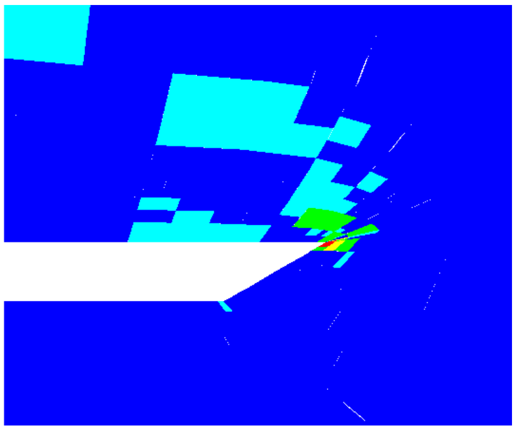
Table 4.1: NLR Delta wing, $M_\infty = 0.3, \alpha = 12.5^o, Re = 4 \times 10^3$, drag-driven adaptation: percentage of choice for each refinement option; iso-h: isotropic h-refinement; sc-h: single-cut h-refinements; dc-h: double-cut h-refinements; iso-p: isotropic p-refinement.

	c_{DOF}				c_{NZ}			
Adaptation step	iso-h	sc-h	dc-h	iso-p	iso-h	sc-h	dc-h	iso-p
1	0.0	84.9	0.0	15.0	0.0	100.0	0.0	0.0
2	0.0	75.1	0.8	24.0	0.0	98.0	1.5	0.5
3	0.0	66.8	1.4	31.8	0.0	95.4	2.5	2.1
4	0.0	70.4	0.0	29.6	0.0	96.2	0.4	3.4
5	–	–	–	–	0.0	95.2	1.2	3.5
6	–	–	–	–	0.0	96.0	0.9	3.0
7	–	–	–	–	0.0	95.7	0.6	3.9
8	–	–	–	–	0.0	95.9	0.3	3.8

The larger frequency of p-refinement when using c_{DOF} is illustrated in Figure 4.12. This figure compares the mesh and approximation order distribution for both measures of cost at a mid-chord cut of the delta wing. Note that both methods choose h-refinement on the upper sharp corner where shear effects are prominent. When using c_{NZ}, the adaptation routine reserves p-refinement mostly for regions where the flow field is smooth, while with c_{DOF} the mesh shows a combination of h and p refinements in sharp and smooth areas.


(a) 4th Mesh with Mach contours for c_{DOF}.


(b) 8th Mesh with Mach contours for c_{NZ}.


(c) 4th p-order distribution for c_{DOF}; the range is $p = 1 \rightarrow 5$.

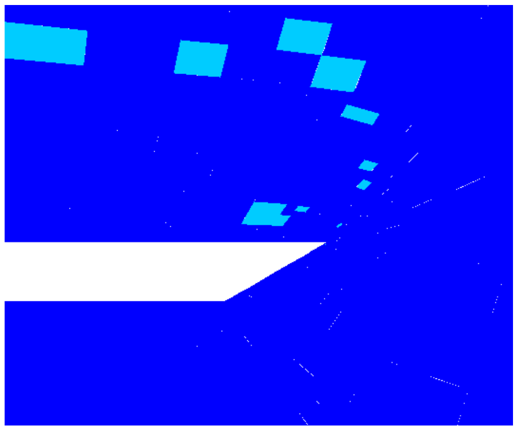

(d) 8th p-order distribution for c_{NZ}; the range is $p = 1 \rightarrow 6$.

Figure 4.12: NLR Delta wing, $M_\infty = 0.3, \alpha = 12.5^o, Re = 4 \times 10^3$: half-chord cut of the drag-adapted meshes.

4.4.3 DPW III - W1 geometry, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$

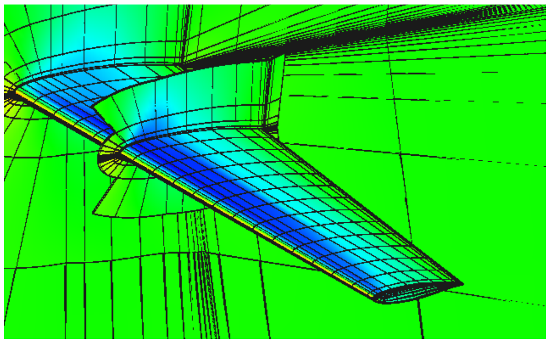
In this case study, we consider the baseline wing geometry (DPW-W1) from the third AIAA Drag Prediction Workshop [27]. This case consists of turbulent, transonic flow over a tapered wing and the mesh adaptation routine is driven by the drag output. The initial curved mesh, shown in Figure 4.13(a), was obtained through agglomeration of cells from a finer structured linear C-grid generated specifically for this purpose. In the agglomeration, each curved hexahedral element was obtained by merging twenty seven linear elements using a distance-based Lagrange interpolation of the nodal coordinates, resulting in cubic ($q = 3$) geometry interpolation. Also, the spacing of the linear mesh is such that the agglomerated mesh presents $y^+ \approx 1$ for the first element off the wall as recommended in the workshop [26] and the outer boundary is located at 100 mean-aerodynamic-chord-lengths away from the wing.

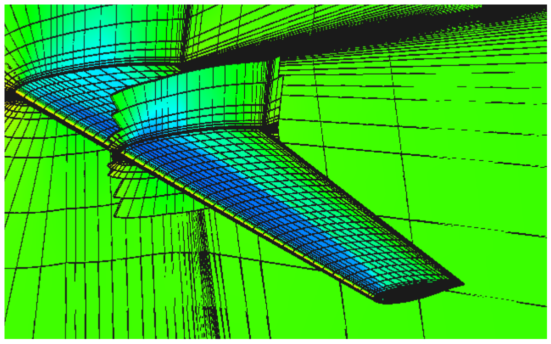
As described in Section 2.2, we use the Spalart-Allmaras turbulence model without trip terms and we assume fully-turbulent flow. Also, Persson and Peraire's [64] shock-capturing method is used to improve stability. The baseline flow solution is obtained with linear ($p = 1$) approximation order. As a basis of comparison, all the adaptive schemes start from the same initial solution so that all methods are compared against the same initial time-stamp. For the adjoint-based adaptation methods, the CPU time taken for the initial adjoint solve is also included in the initial starting time.

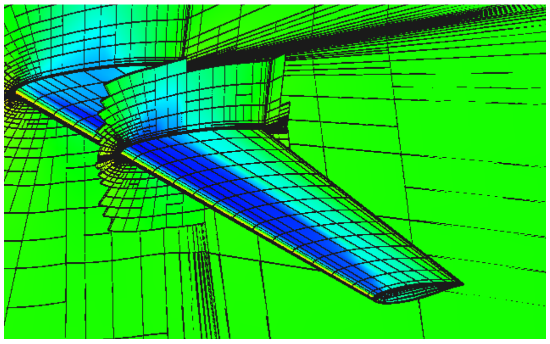
All of the calculations for this case were executed on 180 Harpertown 8-core nodes from NASA's Pleiades supercomputer. Due to the computational expense of these runs, we did not perform a statistical study to account for machine performance variability in the CPU-time measurements.

We compare three mesh improvement strategies starting from the initial $p = 1$ solution shown in Figure 4.13(a). As a reference, one of the strategies is uniform h-refinement (Figure 4.13(b)) in which all hexahedra are divided into 8 elements. The two cost measures described earlier are compared for hp-adaptation in which a fraction $f^{\text{adapt}} = 10\%$ of the elements is selected for refinement at each adaptation step. Additionally, we fix the overall budget of CPU wall-time for each of the three runs and the last converged solutions obtained within that budget are shown in Figure 4.13.

Figure 4.14 shows the drag coefficient convergence for the mesh refinement strategies. The solid lines in Figures 4.14(a) and 4.14(b) are the computed drag values and


(a) Initial pressure contours (29310 cubic elements, $p = 1$).


(b) Pressure contours on the 1st level of uniform h-refinement (234480 cubic elements, $p = 1$).


(c) Pressure contours on the 5th drag-adapted mesh using c_{DOF} (59503 cubic elements).

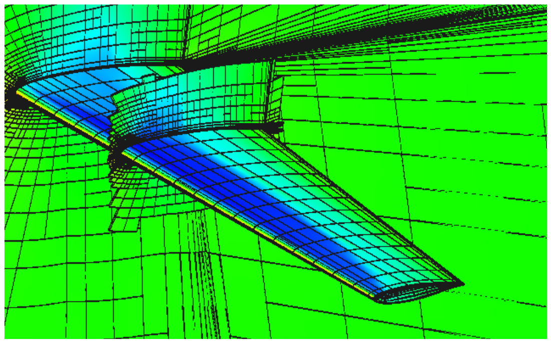
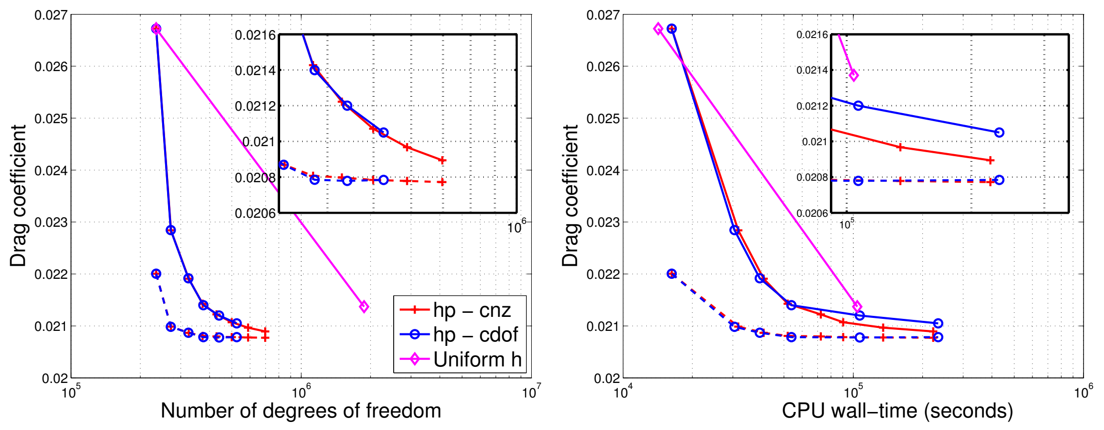

(d) Pressure contours on the 7th drag-adapted mesh using c_{NZ} (85377 cubic elements).

Figure 4.13: DPW III - Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: Initial and drag-adapted meshes with pressure contours.

the dashed lines correspond to the output corrected with the error estimate. The difference between these corrected values for the last two adaptation steps of the output-based strategies is within 0.15 counts of drag. Note that the performance in terms of degrees of freedom of the output-based strategies is very similar. However, in terms of CPU time, the use of c_{NZ} leads to faster output convergence. This difference is due to the more representative measure of solution cost by c_{NZ}. This effect is illustrated in Table 4.2 where we show the frequency at which the refinement options are chosen for each cost measure at each adaptation step. Note that for c_{NZ}, p-refinement is chosen significantly less often than for c_{DOF} and both methods have a propensity to choose p-refinement more often in the later stages of adaptation. Moreover, the large increase in CPU time between the third and fourth adaptation steps for c_{DOF} (Figure



(a) Drag coefficient evolution with respect to number of degrees of freedom

(b) Drag coefficient evolution with respect to CPU time

Figure 4.14: DPW III - Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: drag coefficient convergence; $\diamond$: uniform h-refinement; $\circ$: hp-adaptation with c_{DOF}; $+$: hp-adaptation with c_{NZ}. The dashed lines correspond to the drag values corrected with the error estimate.

4.14(b)) is an effect of p-increment being chosen more often for c_{DOF} (Table 4.2) which makes the primal and adjoint solves more expensive.

Table 4.2: DPW III - Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: percentage of choice for each refinement option; iso-h: isotropic h-refinement; sc-h: single-cut h-refinements; dc-h: double-cut h-refinements; iso-p: isotropic p-refinement.

	c_{DOF}				c_{NZ}			
Adaptation step	iso-h	sc-h	dc-h	iso-p	iso-h	sc-h	dc-h	iso-p
1	0.0	99.3	0.0	0.7	0.0	100.0	0.0	0.0
2	0.0	97.3	0.0	2.7	0.0	99.9	0.0	0.1
3	0.0	94.9	0.0	5.1	0.0	99.8	0.0	0.2
4	0.0	91.8	0.4	7.8	0.0	99.1	0.3	0.6
5	0.0	90.6	0.3	9.1	0.0	98.7	0.5	0.8
6	–	–	–	–	0.0	98.6	0.5	0.9
7	–	–	–	–	0.0	98.6	0.4	1.0

In flows with high Reynolds number, highly stretched cells in regions such as boundary layers and wakes are key to an efficient calculation. To assess the levels of anisotropy in our meshes, we define an aspect ratio measure for one element as follows:

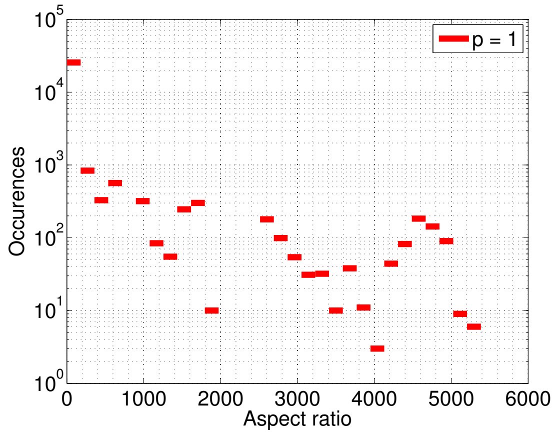
$$\Lambda = \frac{\left(\frac{\mathbb{S}}{2\cdot\dim}\right)^{\frac{\dim}{(\dim-1)}}}{\mathbb{V}}, \tag{4.11}$$

where $\mathbb{S}$ and $\mathbb{V}$ are the cell surface area and volume respectively. Note that $\Lambda = 1$ for a square in two dimensions and a cube in three dimensions, and that $\Lambda > 1$ indicates anisotropic elements. Since the refinement is performed in the elements' reference space, isotropic refinement does not necessarily preserve Λ on curved elements.

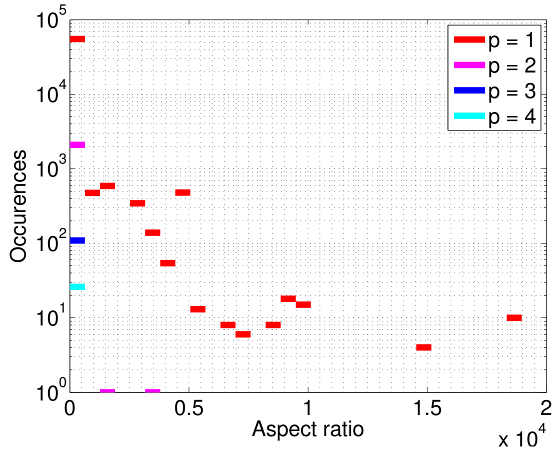
Figure 4.15 shows histograms of the aspect ratios of the cells in the initial and adapted meshes. Note that the aspect ratios in the adapted meshes are in the range of tens of thousands and the higher-order cells generally have lower aspect ratios (in the hundreds range).

Figures 4.16 and 4.17 show two cuts at representative span-wise positions. For comparison purposes, the contours are scaled to the same range for both c_{DOF} and c_{NZ}. The Mach-number contours are similar for both strategies, however c_{NZ} presents a larger number of anisotropic cells along the shock and on the boundary layer. The larger percentage of p-refinement observed for c_{DOF} is illustrated in the order distribution figures. Note that both methods have mostly $p = 1$ cells at the shock and higher-order cells on each side of the shock.

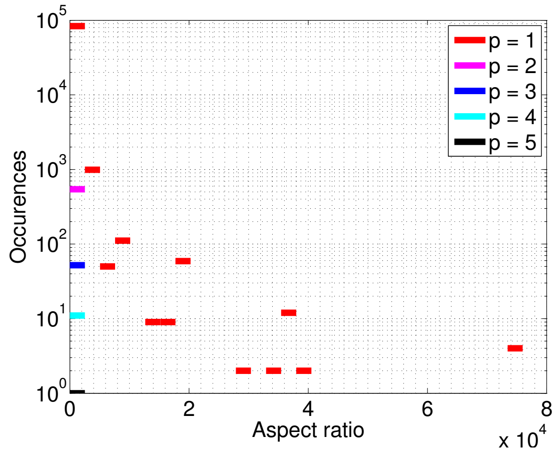
Design optimization methods offer insight on improving vehicle configurations. Similarly, an optimization-based mesh adaptation algorithm offers insight on improving gridding guidelines. We notice that several regions of the flow are frequently targeted for refinement. One of these regions is near the leading edge where the flow accelerates through the sonic condition. This acceleration causes strong variations in the adjoint solution which are responsible for large error indicators. In fact, the adjoint solution is C^1-discontinuous through the sonic condition in inviscid quasi-1D flows [84]. Another region is the edge of the boundary layer, where the tur-



(a) Initial mesh.



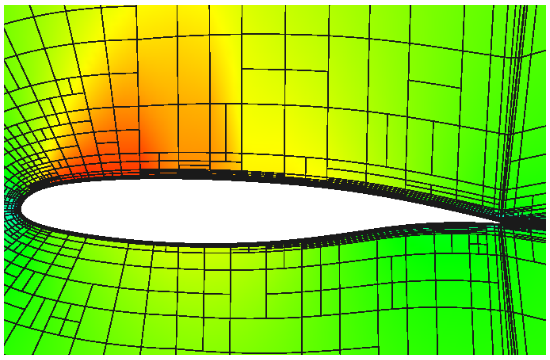
(b) Final (5th) adapted mesh with c_{DOF}.

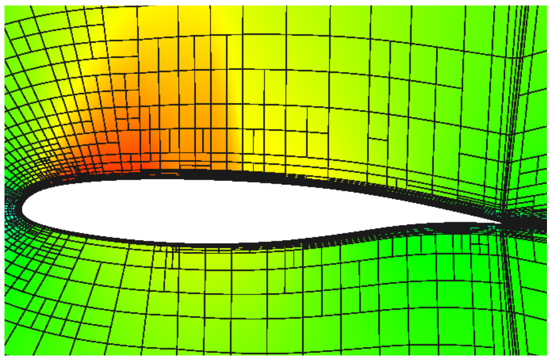


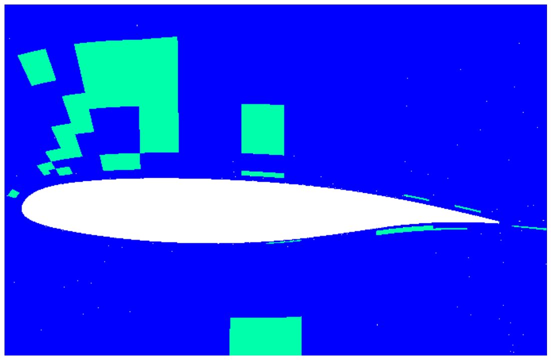
(c) Final (7th) adapted mesh with c_{NZ}.

Figure 4.15: DPW III - Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: aspect ratio histograms for the initial and adapted meshes.

bulent working variable, $\tilde{\nu}$, transitions to zero rapidly. The other two regions are the shock-boundary-layer interaction and the trailing edge. These regions exhibit strong gradients of $\tilde{\nu}$ that contribute to the drag output. Figure 4.18 shows the interaction between the shock and the boundary layer. Note the concentration of cells in the boundary layer and the sharp variation of $\tilde{\nu}$. Further downstream, in the trailing edge region (Figure 4.19), the beginning of turbulent wake is also adapted.


(a) 5th Mesh with Mach contours for c_{DOF}.


(b) 7th Mesh with Mach contours for c_{NZ}.


(c) 5th p-order distribution for c_{DOF}; the range is $p = 1 \rightarrow 5$.

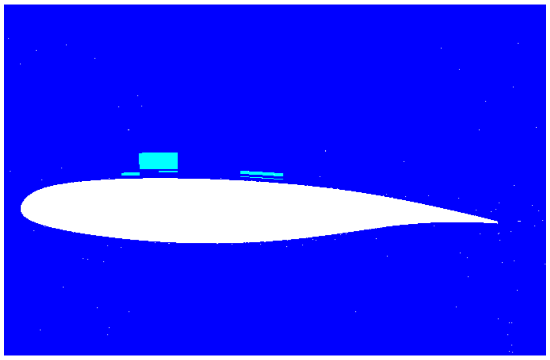
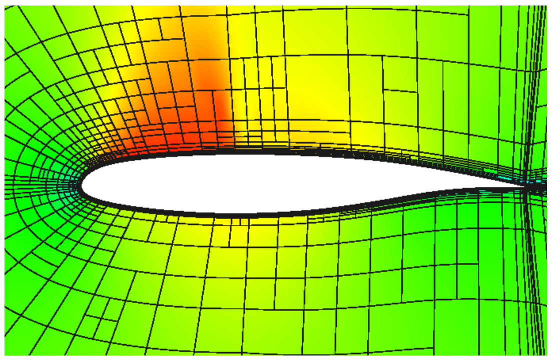

(d) 7th p-order distribution for c_{NZ}; the range is $p = 1 \rightarrow 5$.

Figure 4.16: DPW III - Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: cut at $y =$ 220mm of the drag-adapted meshes.

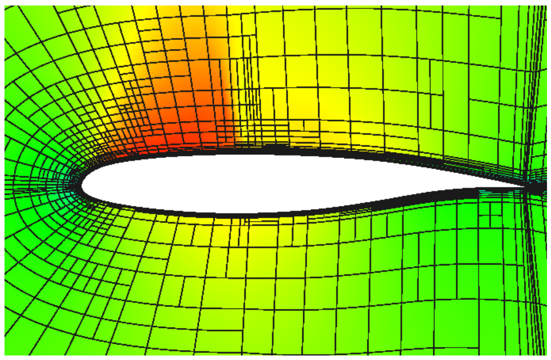
4.4.4 MDA 30p30n – $M_\infty = 0.2,\ Re = 9 \times 10^6,\ \alpha = 16°$

This test case is subsonic, turbulent flow over the MDA 30p30n multi-element airfoil. The purpose of this test case is to assess the effect of limiting the maximum p-order in the hp-adaptation cycle. We consider lift-based hp-adaptation using the c_{NZ} measure with unlimited $p_{\max}$ and with $p_{\max} = 3$. The fraction of elements selected for adaptation at each step is $f^{\mathrm{adapt}} = 10\%$. Uniform h and uniform p refinements are presented to establish a reference for computational cost.

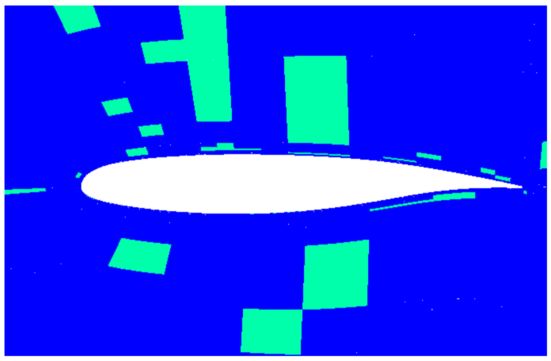
The initial $p = 1$ flow solution and the baseline mesh for this case are presented in Section 3.4.3. CPTC with line-search and exponential CFL evolution is used to solve the discretized equations at each adaptive step. Sixty four processors are used for each run and a maximum of 10 hours is allotted for each adaptation strategy. Figure



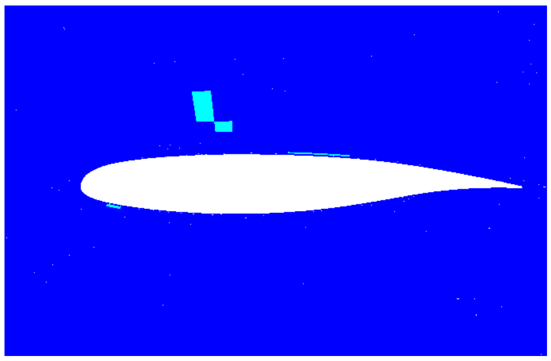
(a) 5th Mesh with Mach contours for c_{DOF}.



(b) 7th Mesh with Mach contours for c_{NZ}.



(c) 5th p-order distribution for c_{DOF}; the range is $p = 1 \rightarrow 4$.



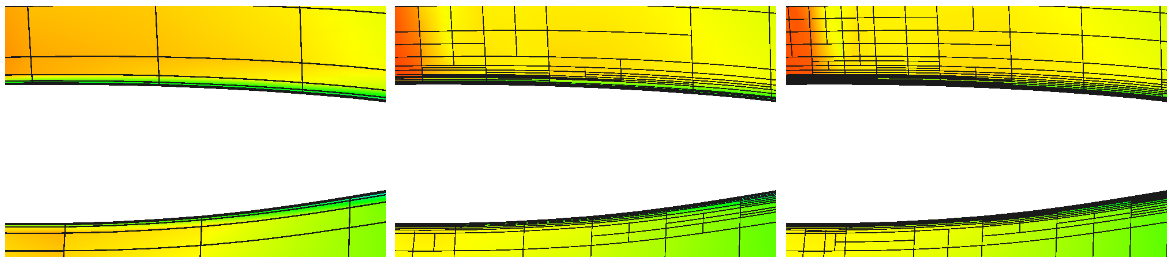
(d) 7th p-order distribution for c_{NZ}; the range is $p = 1 \rightarrow 5$.

Figure 4.17: DPW III Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: cut at $y = 620$mm of the drag-adapted meshes.

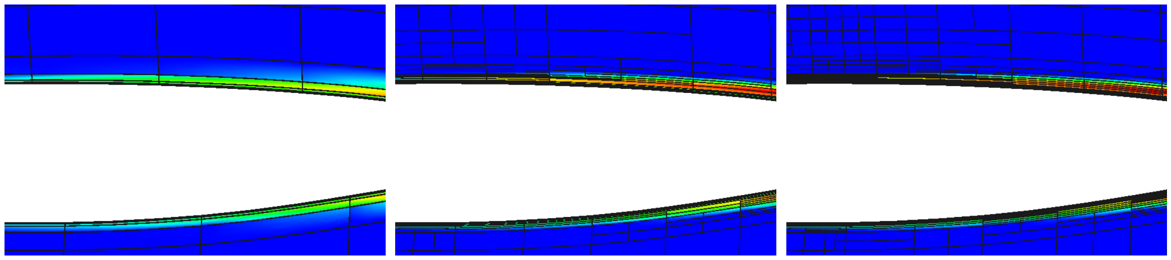
4.20(b) shows the solutions obtained within this time limit. The initial time stamp is set to zero as all strategies start with the same converged initial solution. The time stamps for the output-based strategies include the time consumed by the primal and adjoint solves, error estimation and mesh adaptation.

Similarly to the previous cases, hp-adaptation is significantly more efficient than the uniform refinements (h and p) which can only provide one refined solution within the computational resources allocated. This advantage is also observed in terms of degrees of freedom (Figure 4.20(a)). An additional advantage is that the output-based strategies provide error estimates that can be used to improve the rate of convergence of the output of interest.

Limiting the maximum p-order slightly affects the lift convergence for this case.

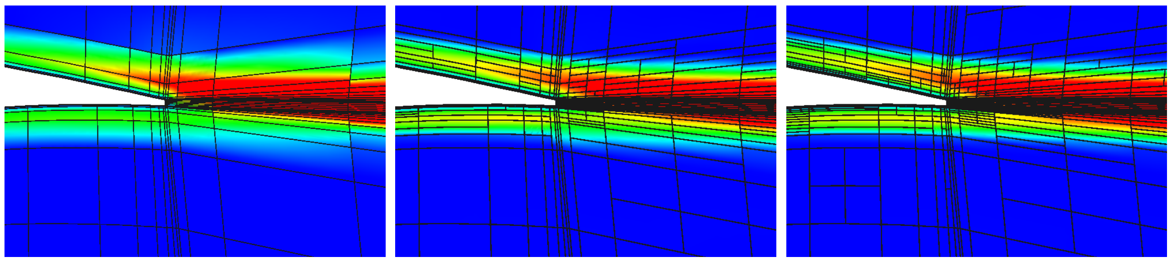


(a) Mach contours for initial mesh. (b) Mach contours for the 5th mesh with c_{DOF}. (c) Mach contours for the 7th mesh with c_{NZ}.



(d) $\tilde{\nu}$ contours for initial mesh. (e) $\tilde{\nu}$ contours for the 5th mesh with c_{DOF}. (f) $\tilde{\nu}$ contours for the 7th mesh with c_{NZ}.

Figure 4.18: DPW III - Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: interaction between shock and boundary-layer at $y = 620$mm.

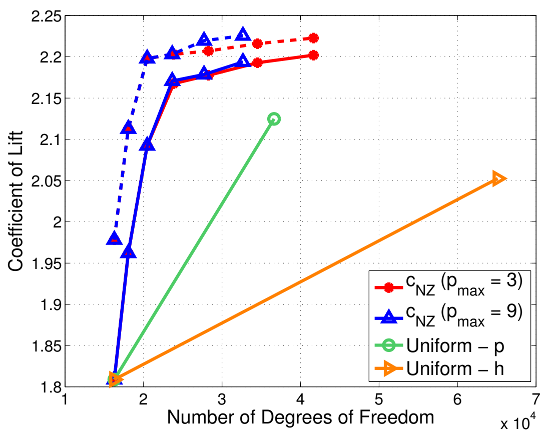


(a) $\tilde{\nu}$ contours for initial mesh. (b) $\tilde{\nu}$ contours for the 5th mesh with c_{DOF}. (c) $\tilde{\nu}$ contours for the 7th mesh with c_{NZ}.

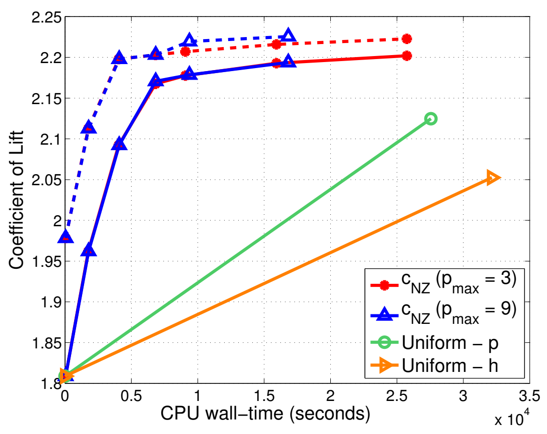
Figure 4.19: DPW 3 Wing 1, $M_\infty = 0.76, \alpha = 0.5^o, Re = 5 \times 10^6$: trailing edge at $y = 620$mm.

The error estimates, on the other hand, are more sensitive to this limitation as they use a locally finer space ($\mathcal{V}^{H,p+1}|_{\kappa^H}$) as surrogate for the continuum . Specifically, the corrected lift for the unlimited-p case in Figure 4.20 becomes "flatter" earlier in the adaptive process. Conversely, the reduced computational cost resultant from limiting p allows for the limited-p run to obtain an additional solution within the time limit.

We now analyze how the refinement choices are affected by limiting $p_{\max}$. Table



(a) Lift coefficient evolution with respect to number of degrees of freedom.



(b) Lift coefficient evolution with respect to CPU time.

Figure 4.20: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$: lift coefficient convergence: the dashed lines correspond to the lift values corrected with the error estimate.

4.3 shows the frequency at which the adaptation algorithm chooses each refinement option at each adaptation step. Note that the percentages are identical for both runs in the first two adaptation steps as the constraint of $p \leq 3$ only becomes active at the third adaptation step. At this point, the adaptive algorithm in the limited-p run stops considering p-refinement as an option for elements with $p = p_{\max}$ and we observe an increase in isotropic and anisotropic h-refinements.

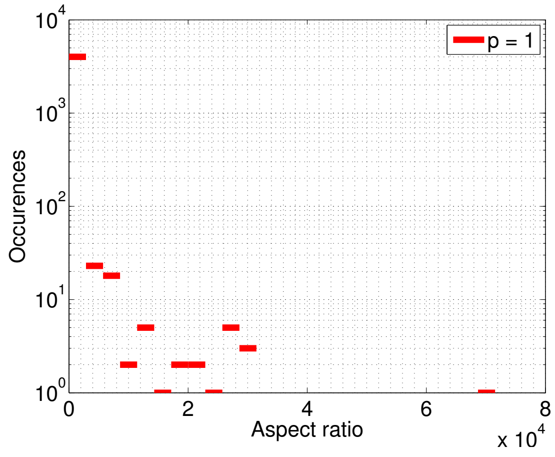
Table 4.3: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$: percentage of choice for each refinement option; iso-h: isotropic h-refinement; sc-h: single-cut h-refinements; iso-p: isotropic p-refinement.

	c_{NZ} ($p_{\max} = 3$)			c_{NZ} (unlimited $p_{\max}$)		
Adaptation step	iso-h	sc-h	iso-p	iso-h	sc-h	iso-p
1	0.0	62.9	37.1	0.0	62.9	37.1
2	0.0	54.2	45.8	0.0	54.2	45.8
3	1.5	64.7	33.8	0.9	57.4	41.7
4	2.0	68.2	29.8	1.2	53.7	45.1
5	5.5	66.3	28.2	1.7	51.0	47.3
6	3.7	66.8	29.5	3.5	47.1	49.4
7	4.8	68.0	27.2	–	–	–

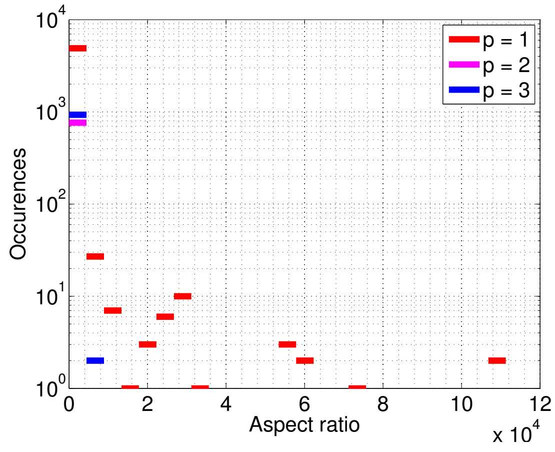
Figure 4.21 shows the histograms of elemental aspect ratio (Eqn. 4.11) for the

initial and adapted meshes. Note that elements with $p > 1$ are generally more isotropic than elements with $p = 1$.

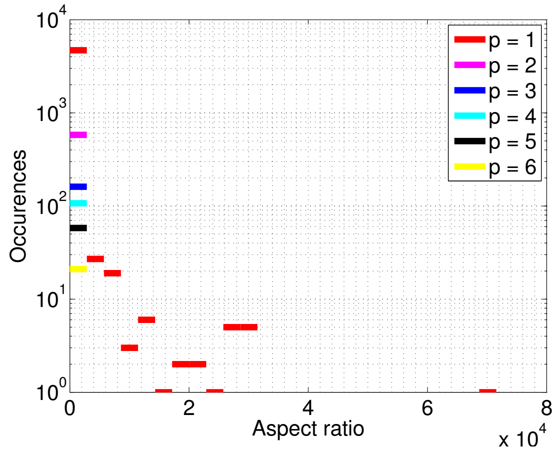
Define an element's *length* as the length of its largest edge. In this work, elemental aspect ratio only changes via directional cuts. Then, we can conclude that elements that are marked for p-refinement are not frequently marked for *length-wise* (parallel to the largest edge) cuts in subsequent adaptation steps as only *length-wise* cuts increase aspect ratio. This observation relies on the fact that we do not allow p-coarsening is this work and the elements created with h-refinement inherits the p-order.



(a) Initial mesh.



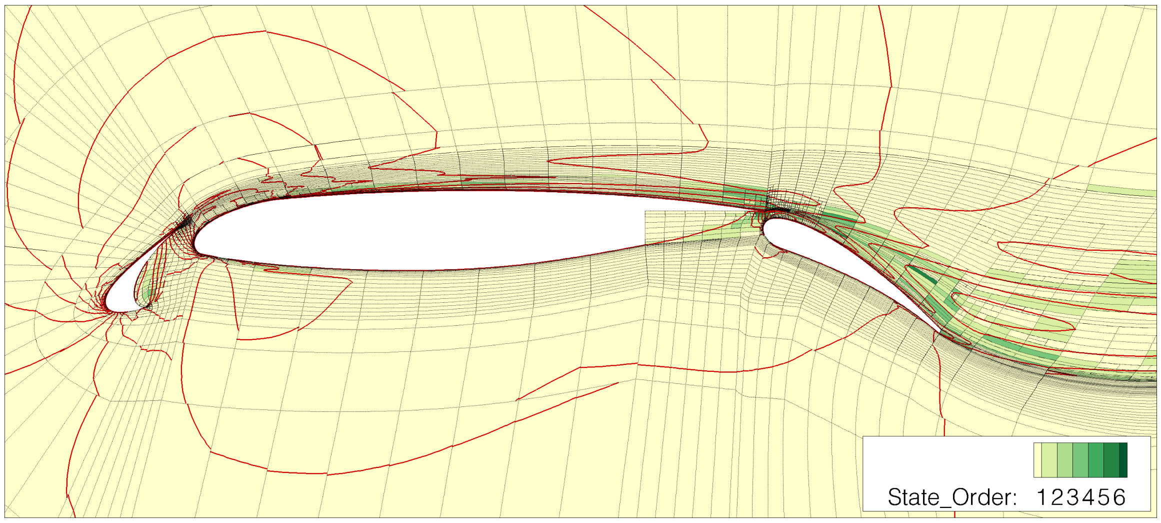
(b) Final (7th) adapted mesh with c_{NZ} and $p_{\max} = 3$.



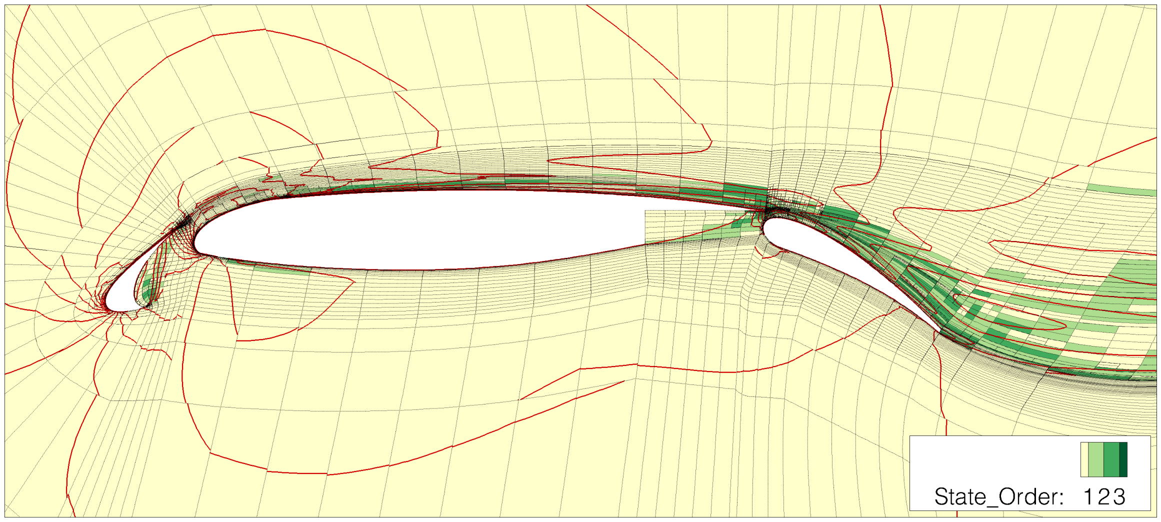
(c) Final (6th) adapted mesh with c_{NZ} and unlimited $p_{\max}$.

Figure 4.21: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$: aspect ratio histograms for the initial and adapted meshes.

It is also interesting to analyze how the various refinement options are distributed in the domain in correspondence with certain flow features. Figure 4.22 shows the mesh and p-order distribution with Mach contour lines for both runs. Note the h-refinement on the trailing edges of the slat and the main airfoil where strong shear is present. Higher-order elements are mostly distributed along the slat's wake and the main airfoil's upper boundary layer as well as the wake.



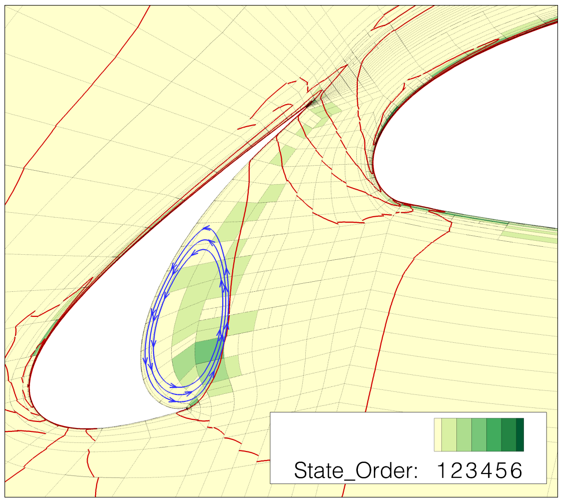
(a) 6th adapted mesh and p-order distribution for unlimited $p_{\max}$.



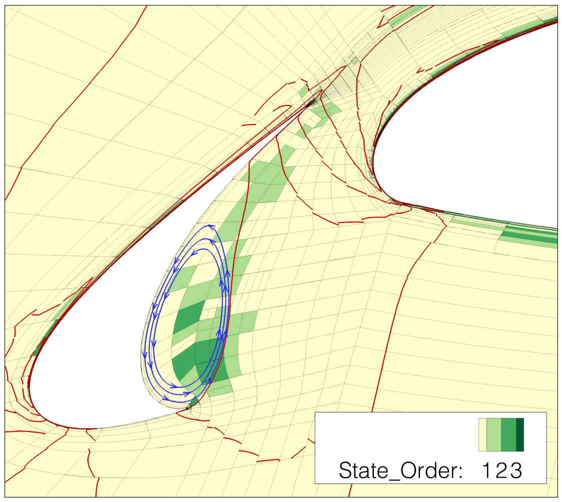
(b) 7th adapted mesh and p-order distribution for $p_{\max} = 3$.

Figure 4.22: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$: final lift-adapted mesh and p-order distribution for $c_{NZ}$with unlimited $p_{\max}$ and $p_{\max} = 3$; red lines: Mach number contours.

Figure 4.23 shows the gap between the slat and the main airfoil. In this region, the fluid experiences a strong acceleration and accurately predicting the shape of the recirculation bubble on the concave side of the slat is important for predicting lift as this bubble affects the mass flow rate through the gap. Note that the adaptive algorithm adequately recognizes that p-refinement is the most efficient option in this region where the flow features are smooth. Furthermore, both runs present similar p-orders in the bubble which indicates that p-refinement is not frequently limited in this region.



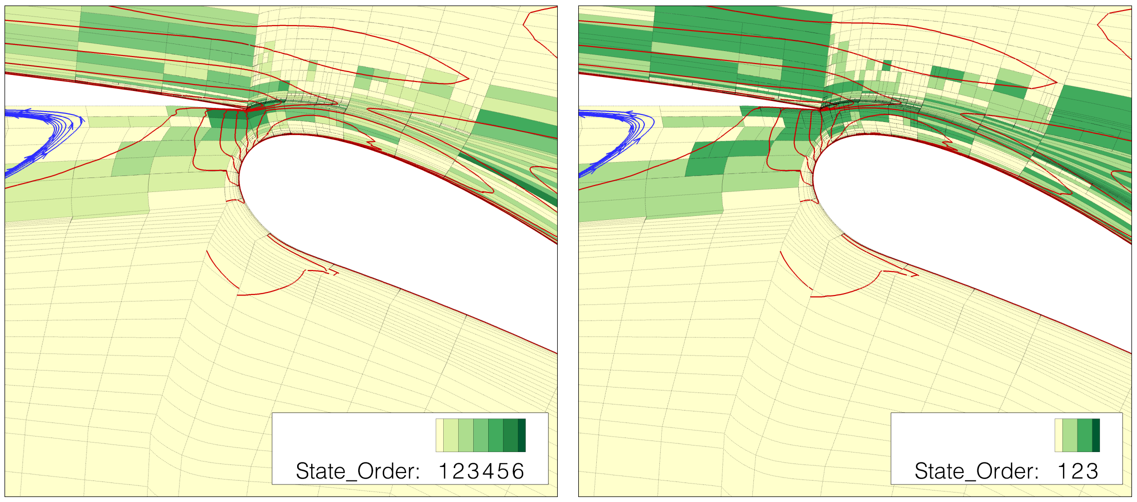
(a) 6th adapted mesh and p-order distribution for unlimited $p_{\max}$.



(b) 7th adapted mesh and p-order distribution for $p_{\max} = 3$.

Figure 4.23: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$ - slat zoom: final lift-adapted mesh and p-order distribution for c_{NZ} with unlimited $p_{\max}$ and $p_{\max} = 3$; red lines: Mach number contours; blue arrow-lines: streamlines in circulation region.

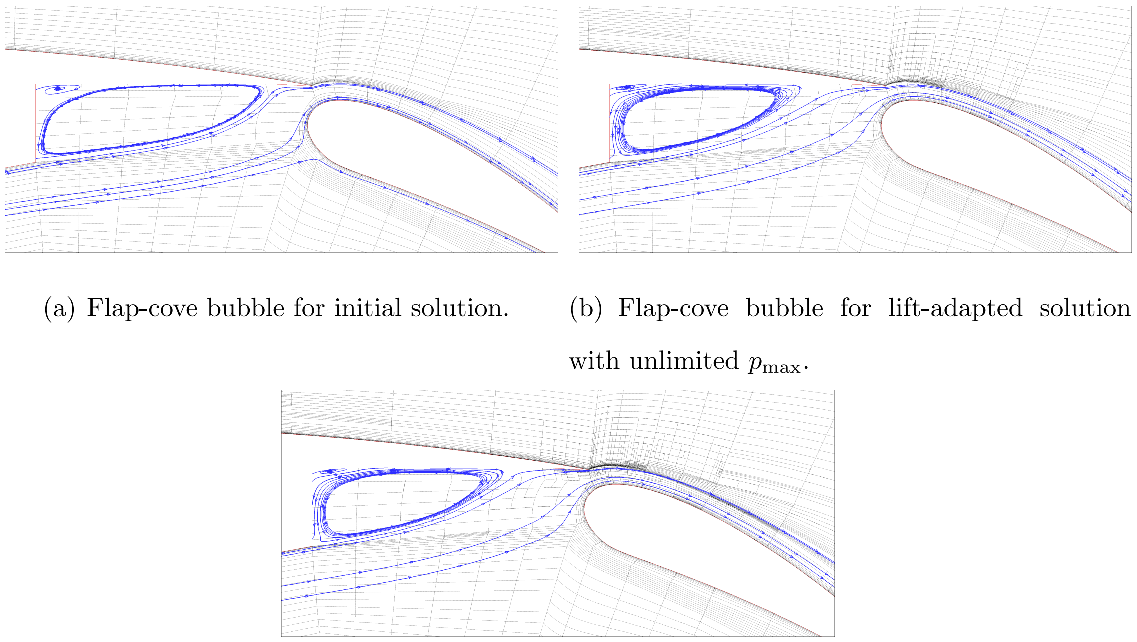
Figure 4.24 shows the flap cove and the gap between the flap and the main airfoil. This region also presents a recirculation bubble that strongly affects the lift force as it dictates the mass flow rate over the upper surface of the flap as shown in Figure 4.25. Note that the path that the fluid takes from the lower side of the main airfoil to the upper side of the flap is paved with higher-order elements. Also, the thin shear layer originating from the trailing edge of the main airfoil is refined in both h and p.



(a) 6th adapted mesh and p-order distribution for unlimited $p_{\max}$.

(b) 7th adapted mesh and p-order distribution for $p_{\max} = 3$.

Figure 4.24: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$ - flap zoom: final lift-adapted mesh and p-order distribution for $c_{\mathrm{NZ}}$with unlimited $p_{\max}$ and $p_{\max} = 3$; red lines: Mach number contours; blue arrow-lines: streamlines in circulation region.



(a) Flap-cove bubble for initial solution.

(b) Flap-cove bubble for lift-adapted solution with unlimited $p_{\max}$.

(c) Flap-cove bubble for lift-adapted solution with $p_{\max} = 3$.

Figure 4.25: MDA 30p30n – $M_\infty = 0.2$, $Re = 9 \times 10^6$, $\alpha = 16°$ - flap zoom: recirculation bubble and streamlines for the initial and lift-adapted solutions.

4.4.5 NACA 0012, $M_\infty = 0.15,\ Re = 6 \times 10^6$, drag polar

This case is one of the NASA's Turbulence Modeling Resource cases [56]. The purpose of this case is to validate the modifications made to the SA model. As suggested by NASA's Turbulence Modeling Resource, the domain's outer boundary is located 500 chord-lengths away from the airfoil. We consider eight angles of attack in the drag polar: $\alpha = 0°$, $2°$, $4°$, $6°$, $8°$, $10°$, $12°$, and $15°$. For each angle of attack, an initial quartic mesh is generated by agglomerating 16 quadrilaterals from a linear mesh. The linear meshes are generated so as the cells downstream from the airfoil are approximately aligned with the wake. Figure 4.26 shows an example of an initial quartic mesh.

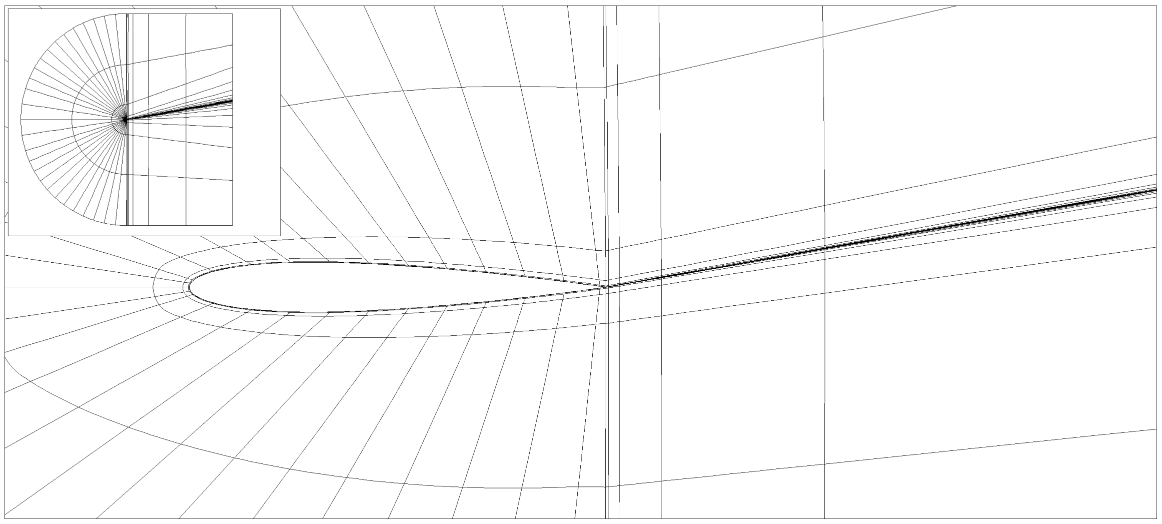
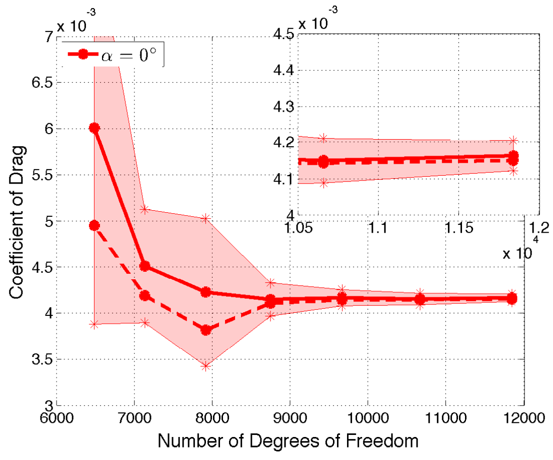


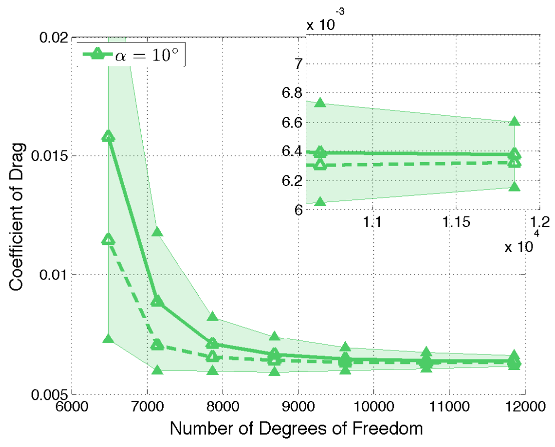
Figure 4.26: NACA 0012, $M_\infty = 0.15,\ Re = 6 \times 10^6$, drag polar: Initial mesh for $\alpha = 10°$ (720 quartic elements).

In this case, we fix the polynomial order at $p = 2$ and consider only the h-refinement options in the adaptive method. The discretized SA equation is scaled by $\kappa_{\mathrm{SA}} = 1000$ and $\kappa_{\mathrm{BR2}} = 10$ for the viscous discretization. The adaptation is driven by drag error and $f^{\mathrm{adapt}} = 10\%$. The physicality-constrained solver is used with line-search and exponential CFL progression for the primal solves.

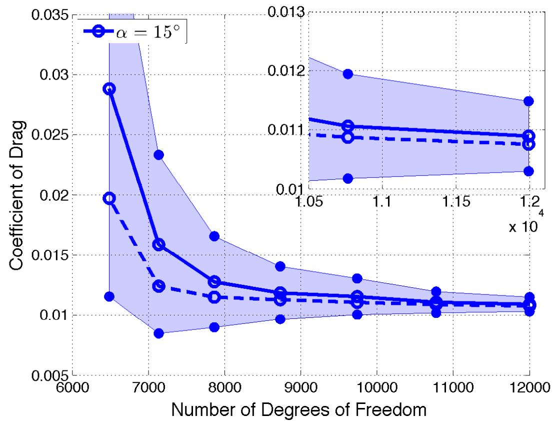
To simplify our analyses, we limit the number of the adaptive steps to 6 for all the angles and measure the error level of the final result. Figure 4.27 shows the drag convergence for three representative angles of attack. The largest final error estimate over all the angles of attack is approximately 3 drag counts ($\sim 3\%$) in the $\alpha = 15°$ case.



(a) $\alpha = 0°$.

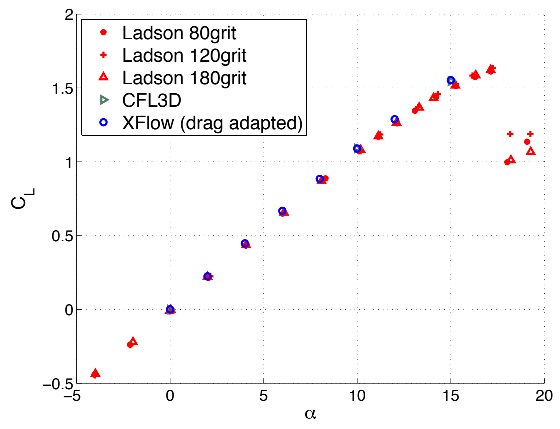


(b) $\alpha = 10°$.

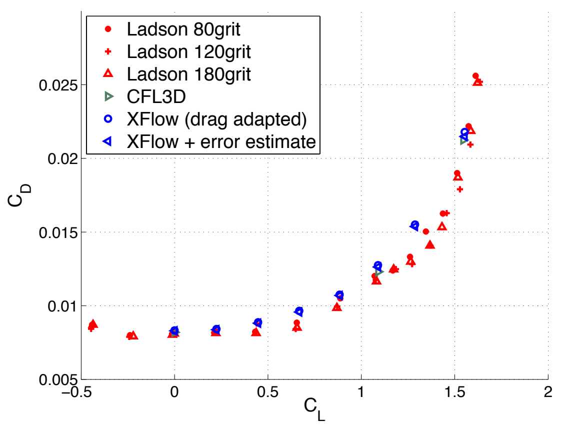


(c) $\alpha = 15°$.

Figure 4.27: NACA 0012, $M_\infty = 0.15$, $Re = 6 \times 10^6$, drag polar: drag convergence for three angles of attack; solid lines: drag values; dashed lines: drag corrected by its error estimate; shading: magnitude of the sum of error indicators.



(a) Lift coefficient versus angle of attack.

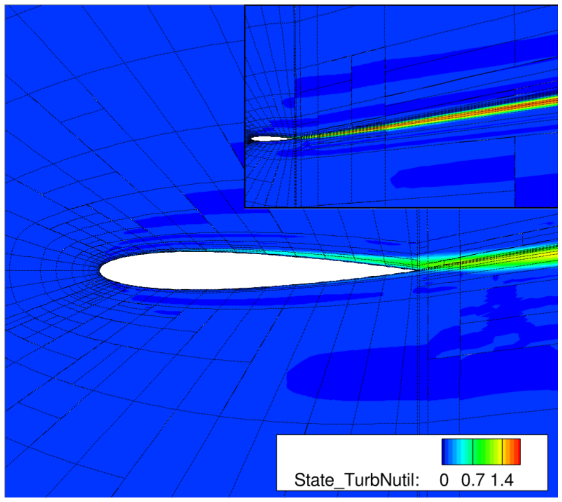


(b) Drag polar

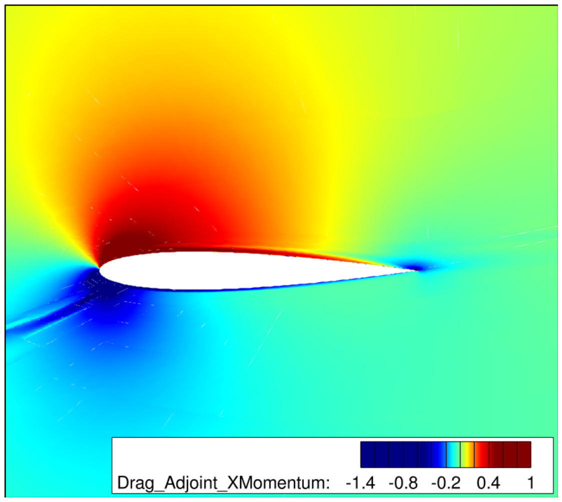
Figure 4.28: NACA 0012, $M_\infty = 0.15$, $Re = 6 \times 10^6$, drag polar: comparison with experimental data.

Figure 4.28 compares our adaptive results with Ladson's experimental data [47] and with computational results computed with CFL3D on a fine, 897×257 element, structured grid [56]. The experimental data consists of three sets of wind tunnel runs with varying roughness of carborundum strips to force transition to turbulence at the 5% position along the chord. This reduces transition effects and allows for a more adequate comparison with fully turbulent simulations.

In spite of the adaptation being driven by drag error, the lift values in Figure 4.28(a) are in close agreement with the experimental data. Our computed drag values are within 3% difference with respect to CFL3D's results which is within the spread of 4% in the CFD results with the SA model presented in Ref. [56]. With respect to the experimental values in Figure 4.28(b), the simulations show slightly larger drag values. We attribute these differences to the turbulence model and to experimental errors as the adjoint-based error estimation and adaptation only targets discretization error.



(a) 6th drag-adapted mesh and SA-working variable contours.



(b) 6th drag-adapted x-momentum adjoint solution for drag.

Figure 4.29: NACA 0012, $M_\infty = 0.15$, $Re = 6 \times 10^6$, drag polar: final mesh, $\rho\tilde{\nu}$ contours, and drag adjoint for $\alpha = 10°$.

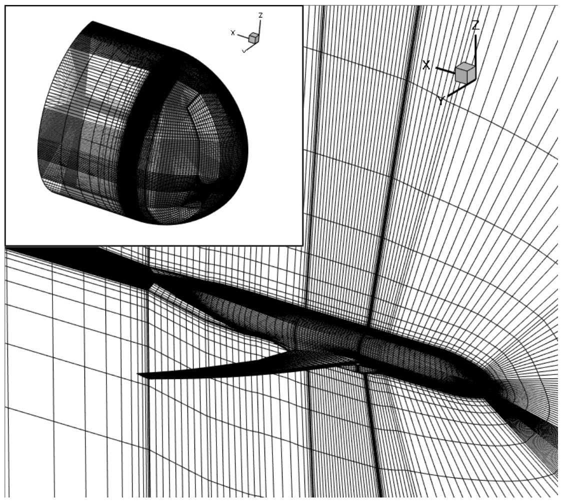
The adjoint solution offers insight on regions of the computational domain where discretization errors affect the output of interest. Figure 4.29(b) shows the x-momentum drag-adjoint solution for the $\alpha = 10°$ case. The most notable feature of this adjoint solution is the stagnation streamline which, in the inviscid limit, is a weak inverse-square-root singularity [29]. This sharp variation of the adjoint is reflected in the adapted mesh in Figure 4.29(a).

Other features that are important for accurate prediction of drag are the boundary layer, the upper flow acceleration region, the trailing edge, and the wake. These regions are also frequently targeted for refinement as they present large magnitudes of the adjoint variables.

4.4.6 CRM - wing-body geometry, $M_\infty = 0.85,\ C_L = 0.5,\ Re_{\mathbf{MAC}} = 5 \times 10^6$

This case is part of the Fifth Drag Prediction Workshop and it consists of transonic, turbulent flow over NASA's Common Research Model [76]. This wing-body geometry mimics a modern passenger aircraft and its purpose is to establish a reference for testing computational tools for simulation and design. This case is the most challenging case of this thesis and certain aspects of the results presented here could lead to additional investigations in the future. These aspects are discussed later in this section.

The cubic mesh used in this case was generated via agglomeration of linear cells. The initial linear mesh was generated with the tradeoff of being coarse to use in our adaptation routine but fine-enough to represent the geometry adequately. Figure 4.30 shows the linear and the agglomerated meshes. The off-wall spacing in the agglomerated mesh is such that $y^+ \approx 100$, based on a flat-plate correlation for the coefficient of friction and with the Reynolds number based on the mean aerodynamic chord ($Re_{\text{MAC}} = 5 \times 10^6$).


(a) Linear mesh used for agglomeration (1218375 elements).

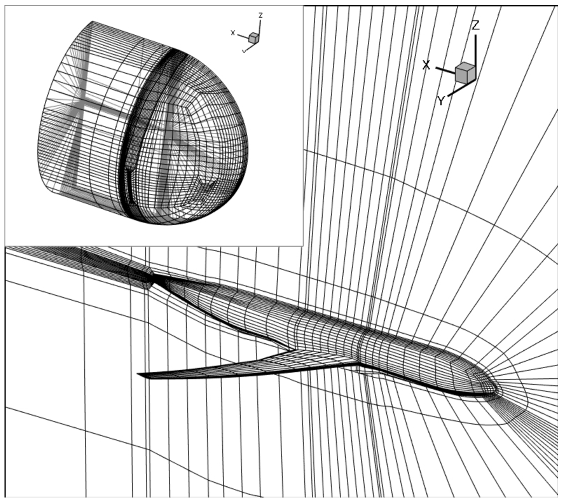

(b) Cubic mesh generated via agglomeration (45125 elements).

Figure 4.30: CRM - wing-body geometry, $M_\infty = 0.85,\ C_L = 0.5,\ Re_{\text{MAC}} = 5 \times 10^6$: linear and agglomerated cubic meshes.

The discretized SA equation is scaled by $\kappa_{\text{SA}} = 100$ and the stabilization constant in the viscous discretization is $\kappa_{\text{BR2}} = 15$. Also, the shock-capturing term described in 2.3.2 is added to the residual operator.

We consider anisotropic h-adaptation at fixed $p = 1$. Converging the initial solution for this problem is difficult. The physicality-constrained solver with the greedy line-search and mRDM is used for the first primal solve. In addition, one step of mesh adaptation based on the penalty projection (Eqn. 3.23) is taken to help the solver to converge. This adaptation method is presented in Ref. [15]. In subsequent solves, converging the residual is significantly easier. The greedy algorithm is then turned off and the mRDM strategy is substituted by exponential CFL progression.

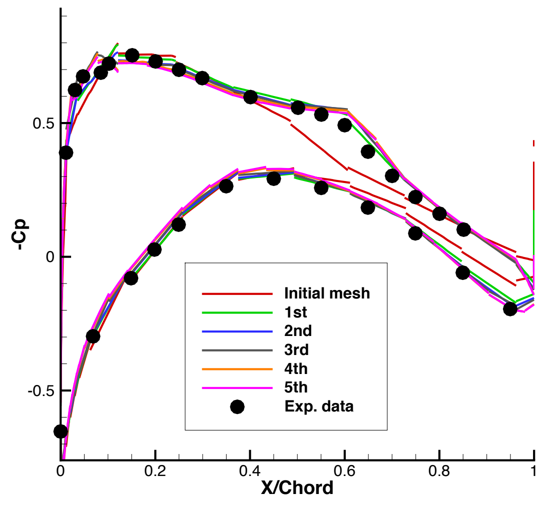
The output used for adaptation is the total drag at a fixed lift. That is, at each primal solve, the angle of attack is trimmed so that the coefficient of lift is $C_{L_{\text{target}}} = 0.5 \pm 0.001$. The method for trimming α is described in Section 5.2.

Due to lack of spatial resolution in the initial mesh, the flow separates (Figure 4.32(a)) before the lift requirement is achieved. The solution in the initial mesh is nearly unsteady which makes the adjoint problem very ill-conditioned and, consequently, causes the error-estimates to be very large as shown in Figures 4.33(a) and 4.33(b). In this situation, the lift requirement is relaxed and the adaptive process proceeds. This decision is not yet automated and is one of the aspects of this problem that could benefit from further research.

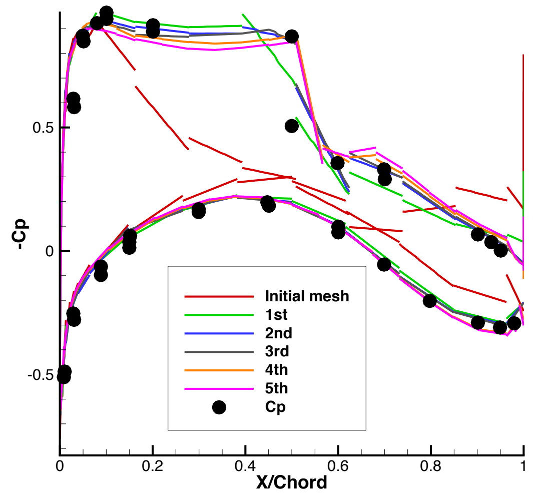
After the first drag-based adaptation step, the flow field is significantly different (Figure 4.32(b)). The supersonic region is larger and no visible flow separation is present. The lift requirement is now satisfied and the error estimates for lift and drag are significant smaller (Figure 4.33).

The Mach number contours shown in Figure 4.32 do not present large differences after the second adaptation step. Also, the areas targeted for adaptation are similar to the regions observed in the DPW III - W1 case. These regions are: the stagnation streamline, the sonic transition, the shock-boundary-layer interaction, and the wake.

Figure 4.31 compares the pressure coefficient at two span locations with the corresponding experimental data[1] [74]. Note that the initial result is very far from the experiments, however, after one adaptation step the pressure distribution is much closer to the experimental data and as the adaptation progresses, the shock profile becomes sharper and the changes in pressure distribution become smaller.



(a) Pressure coefficient at 13.06% of the reference span.



(b) Pressure coefficient at 50.24% of the reference span.

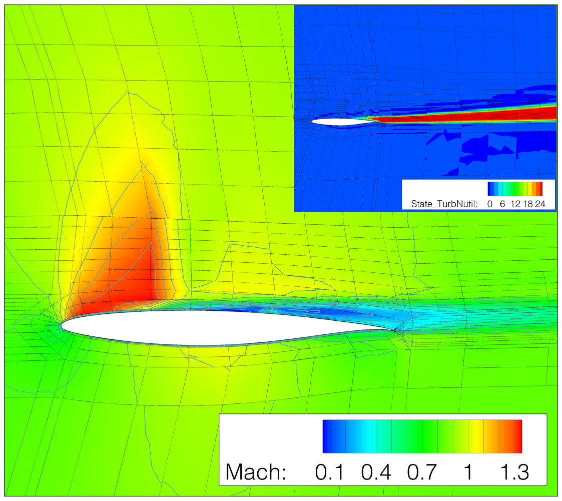
Figure 4.31: CRM - wing-body geometry, $M_\infty = 0.85,\ C_L = 0.5,\ Re_{\text{MAC}} = 5 \times 10^6$: comparison of pressure coefficient with experimental data.

Figure 4.33 shows the convergence history for drag, lift, and pitching moment. Note that our results for pitching moment are within the range of data submitted to the workshop, while the drag values are above the range of results from the workshop. However, it is worth emphasizing that the finest solution presented here has a factor of 5 to 10 fewer degrees of freedom than the mid-range meshes used in the uniform refinement studies in the workshop.

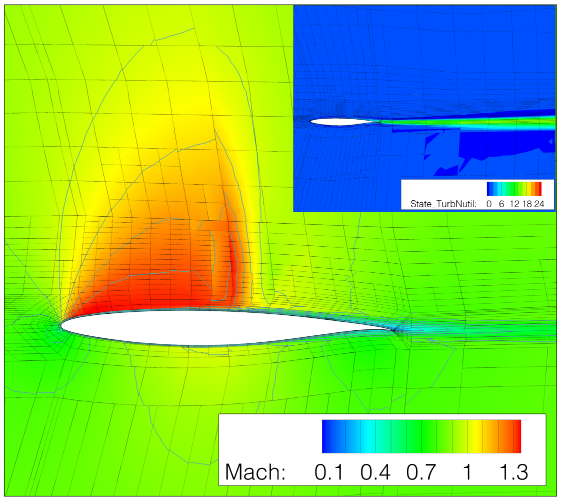
Drag error estimation with fixed lift

In a fixed-lift run, we directly solve the discrete residual equations and indirectly, via a feedback loop (Section 5.2), solve for α to satisfy the lift constraint. This can

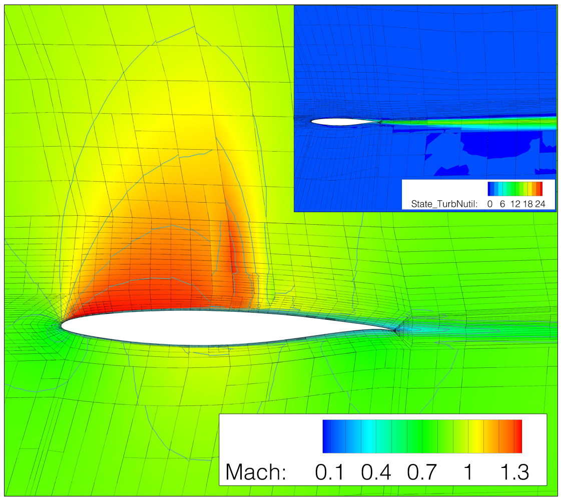
[1]Experimental data was digitized from the 5th Drag Prediction Workshop summary presentation.



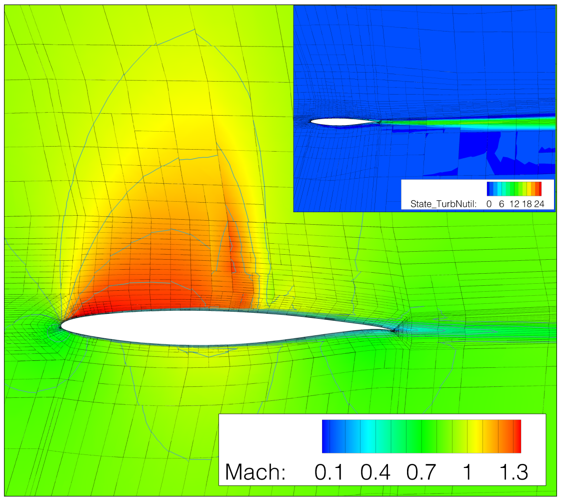
(a) Initial mesh ($\alpha = 2.8°$).



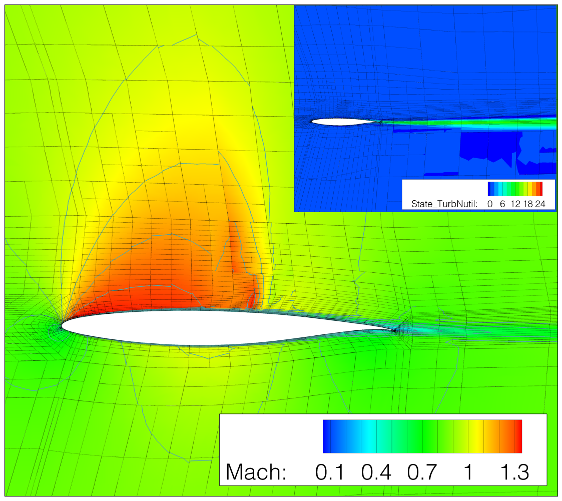
(b) 1st drag-adapted mesh ($\alpha = 2.675°$).



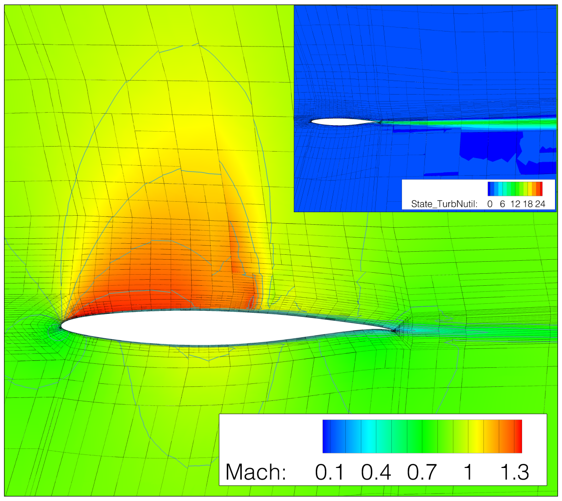
(c) 2nd drag-adapted mesh ($\alpha = 2.465°$).



(d) 3rd drag-adapted mesh ($\alpha = 2.37°$).

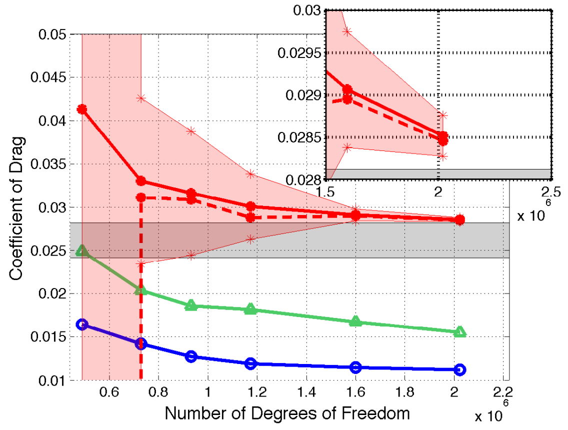


(e) 4th drag-adapted mesh ($\alpha = 2.2665°$).

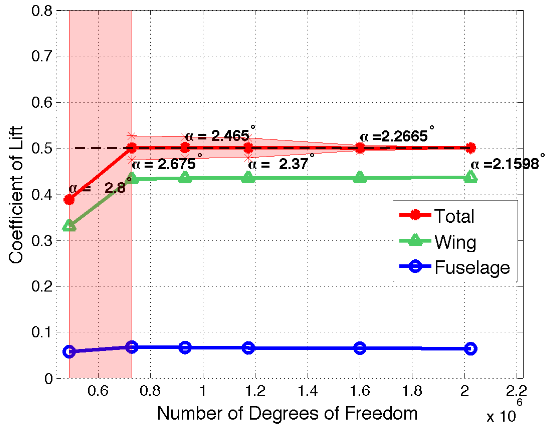


(f) 5th drag-adapted mesh ($\alpha = 2.1598°$).

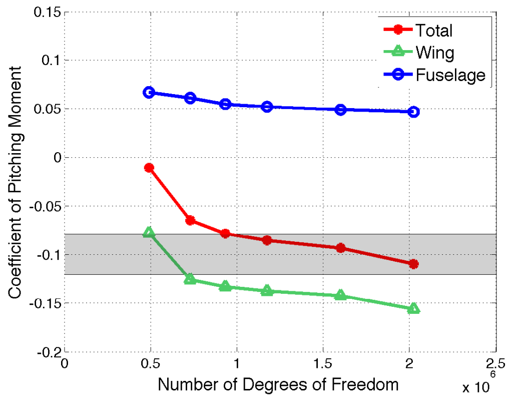
Figure 4.32: CRM - wing-body geometry, $M_\infty = 0.85,\ C_L = 0.5,\ Re_{\text{MAC}} = 5 \times 10^6$: slice at 37% of the span (428 inches).



(a) Drag convergence; dashed line: drag corrected by its error estimate; red shaded region is delimited by sum of drag error indicator over the elements.



(b) Lift history; red shaded region is delimited by sum of the lift error indicator over the elements.



(c) Pitching moment history.

Figure 4.33: CRM - wing-body geometry, $M_\infty = 0.85$, $C_L = 0.5$, $Re_{\text{MAC}} = 5 \times 10^6$: drag, lift, and pitching moment for the sequence of adapted meshes; gray shaded region: range of data submitted to DPW-V.

be written as the following system:

$$\begin{cases} \mathbf{R}(\alpha, \mathbf{U}) & = 0 \\ L(\alpha, \mathbf{U}) & = 0, \end{cases} \tag{4.12}$$

where $L(\alpha, \mathbf{U}) = C_L(\alpha, \mathbf{U}) - C_{L_{\text{target}}}$. Here, we introduce α as an argument in the residual operator to explicitly denote the parameterization of the free-stream boundary condition.

As we are interested in computing drag with a solution that satisfies the constraints in Eqn. 4.12, we form a Lagrangian by introducing adjoint variables:

$$\mathcal{L}(\alpha, \mathbf{U}, \boldsymbol{\Psi}_R, \Psi_L) = D(\alpha, \mathbf{U}) + \boldsymbol{\Psi}_R^T \mathbf{R}(\alpha, \mathbf{U}) + \Psi_C L(\alpha, \mathbf{U}), \tag{4.13}$$

where $D(\alpha, \mathbf{U})$ is the drag function, Ψ_R is the drag adjoint as defined previously and Ψ_C represents the discrete sensitivity of drag with respect to perturbations in the lift constraint. In the continuum limit, this would represent the slope of a tangent to the drag polar curve.

We seek variations of the drag function, *i.e.* a drag error estimate, that satisfy the constraints in Eqn. 4.12. This corresponds to setting $\delta\mathcal{L} = 0$ for general perturbation in the input parameters. Taking the variation of the Lagrangian yields:

$$\delta\mathcal{L} = \delta D + \underbrace{\boldsymbol{\Psi}_R^T \delta\mathbf{R}}_{\text{(a)}} + \underbrace{\Psi_C \delta L}_{\text{(b)}} = 0, \tag{4.14}$$

where "a" is the drag error estimate for fixed α described in Section 5.2,"b" is the influence of the lift error in the drag error due to the lift constraint and δL is the lift error estimate.

We compute Ψ_C according to:

$$\Psi_C = \frac{\partial D/\partial\alpha}{\partial L/\partial\alpha}, \tag{4.15}$$

where the sensitivities with respect to α are approximated via an inner product between the drag and lift adjoints and a residual perturbation due to a perturbation in α. This procedure is described in Section 5.2.

CHAPTER 5

Implementation aspects

In this chapter we describe three important implementation aspects of this work. First, we describe the parallel algorithm for the optimization-based adaptation method from Chapter 4. Then, we present the adjoint-based boundary condition correction algorithm used for the fixed-lift runs. The chapter ends with a weighted mesh partitioning algorithm that makes use of preconditioner information to improve the parallel efficiency of the solver.

5.1 Parallel adaptation algorithm

The methods for solving the flow equations, estimating output error, and selecting the elements for adaptation are implemented in a distributed-memory architecture. That is, the processors only have direct access to mesh and corresponding data that are subsets of the global domain and they exchange information using a routed network. Due to the compact stencil of DG methods, the inter-processor communication is only through one layer of elements, resulting in a easily-parallelizable algorithm with a large ratio of computation per communication. The hanging-node refinement, however, creates meshes with communication patterns that are not easily predictable due to possible differences of level of refinement between adjacent elements (Section 4.2). For this reason, the refinement process is performed serially by merging the subdomains from all the processors into one processor. Fortunately, the refinement overhead is not significant because the operations involved in this process are not computationally intensive compared to the primal and adjoint solves. On the other hand, the operations involved in the discrete mesh optimization problems can be expensive if performed serially, especially in three-dimensional problems. Our trade-off is to merge the subdomains from all of the processors and to distribute copies of the global domain to all processors prior to solving the local optimization problems. This way, the full mesh already resides in one processor for the refinement using the computed optimum options. This process is illustrated in Figure 5.1.

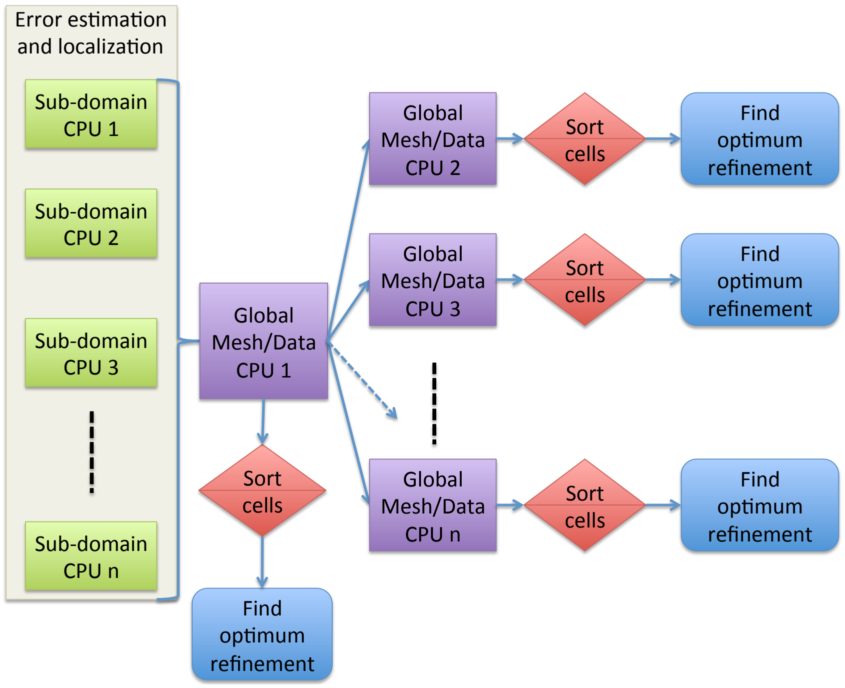


Figure 5.1: Parallel algorithm for distributing the local discrete optimization problems.

Once each processor has a copy of the full domain, the elements are sorted (Algorithm 5.1) such that each processor solves its share of local mesh optimization problems. The resulting optimum refinement decisions – arrays of integers – are then merged into the main processor and the mesh is refined.

ALGORITHM 5.1 Algorithm for sorting elements to be optimally refined

```
1: Input: $\mathbf{Elem2Refine}[i = 1 \rightarrow N^{\text{ref}}_{\kappa^H}]$    ▷ Array of elements marked for refinement
2: for $j = 1 \rightarrow \left\lceil \frac{N^{\text{ref}}_{\kappa^H}}{N_{\text{CPU}}} \right\rceil$ do    ▷ where $\lceil\cdot\rceil$ is the ceiling function
3:     $k \leftarrow (j-1) \cdot N_{\text{CPU}} + i_{\text{CPU}}$    ▷ $i_{\text{CPU}}$ is the processor number
4:     if $k \leq N^{\text{ref}}_{\kappa^H}$ then
5:         $\kappa^{\text{ref}}_H \leftarrow \mathbf{Elem2Refine}[k]$
6:         Find optimal refinement for element $\kappa^{\text{ref}}_H$
7:     else
8:         Processor "$i_{\text{CPU}}$" finished its work.
9:     end if
10: end for
```

5.2 Adjoint-based boundary condition correction

Frequently in the aeronautical industry, CFD simulations are conducted under trimmed conditions, meaning, under fixed, user-defined values of certain outputs – typically lift or pitching moment. This means that certain boundary condition parameters, *e.g.* angle of attack, depend on outputs computed from the flow solution. Thus, a feedback loop must be used to correct those input parameters.

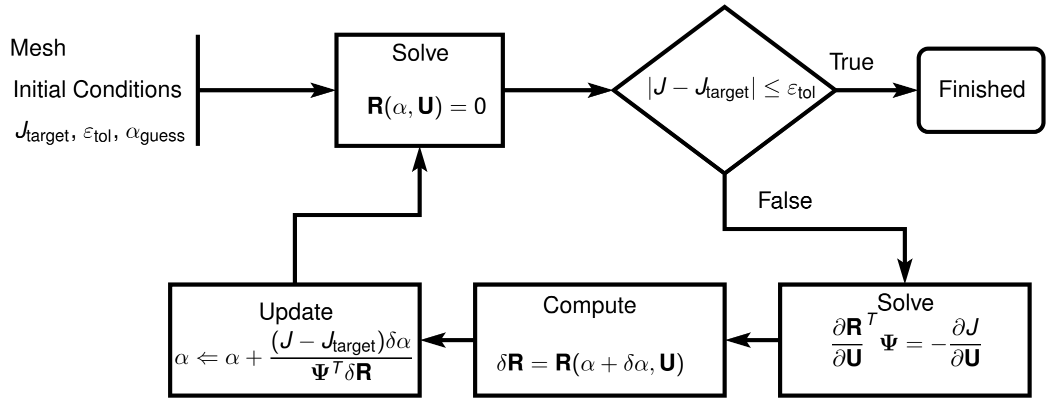

Mesh
Initial Conditions
$J_{\text{target}}, \varepsilon_{\text{tol}}, \alpha_{\text{guess}}$
Solve
$\mathbf{R}(\alpha, \mathbf{U}) = 0$
$|J - J_{\text{target}}| \leq \varepsilon_{\text{tol}}$
True
Finished
False
Solve
$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \mathbf{\Psi} = -\frac{\partial J}{\partial \mathbf{U}}$
Compute
$\delta\mathbf{R} = \mathbf{R}(\alpha + \delta\alpha, \mathbf{U})$
Update
$\alpha \Leftarrow \alpha + \frac{(J - J_{\text{target}})\delta\alpha}{\mathbf{\Psi}^T \delta\mathbf{R}}$

Figure 5.2: Adjoint-based boundary-condition parameter correction.

The feedback loop used in this work is illustrated in Figure 5.2, where J_{target} is the target value of the output for which the parameter, α, is trimmed. The cycle starts by solving the flow equations using an initial guess for α. Then, J is computed and checked against J_{target} under a trimming tolerance, ε_{tol}. Until this tolerance is met, α is corrected using Newton's method for which the sensitivity of J with respect to α is needed. This sensitivity is computed via an inner product between an adjoint for J and a residual perturbation δR resultant from a perturbation in α. This residual perturbation is computed by evaluating the residual with the boundary condition perturbed by $\delta\alpha$.

In cases where the target value for the output is not achievable or the initial guess is bad, the cycle in Figure 5.2 may not converge. In those cases, a contingency plan is needed, *e.g.*, a maximum number of iterations is assigned or the cycle is restarted with a better initial guess. In the output-based adaptation framework presented in this work, the boundary conditions are only trimmed if the error estimate for J is smaller then its trimming tolerance, ε_{tol}.

5.3 Weighted mesh partitioning

The time taken to solve the primal and dual problems increases with approximation order, p, and the non-homogeneity of p affects the distribution of computational work amongst the processors. Therefore, dynamic load-balancing for *hp*-adaptive methods is important for efficient use of computational resources. However, such balance is not trivial and, in fact, is a topic of research rarely explored. Two difficulties are that the computational effort required for evaluating the residual operator and its Jacobian is not constant amongst elements in the mesh and that the performance of the line-Jacobi preconditioner deteriorates when cells with strong coupling reside on different processors [19].

Typically in CFD, the mesh is represented as an irregular graph where each element κ^H is a node in the graph and the interior faces $\partial\kappa^H \setminus \partial D$ are edges in the graph (Figure 5.3). This graph is then partitioned using a multilevel algorithm in which sequences of smaller graphs are systematically generated and partitioned until the partitions are as close to equal size as possible.

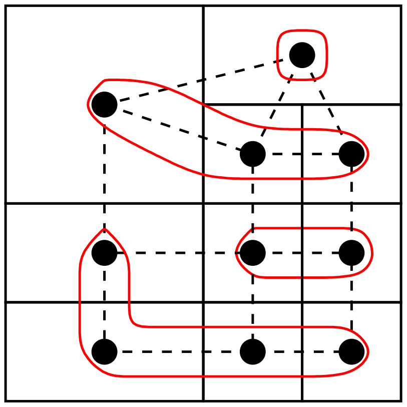


Figure 5.3: Example of mesh (continuous black lines) and corresponding graph (dashed lines); the sets of elements circled in red represent lines of the preconditioner.

In our work, we use the multilevel k-way graph partitioning algorithm implemented in the ParMETIS library [41] which permits the attribution of weights to nodes and edges of the graph. The node weights are used to represent the computational effort for each element due to non-uniformity of p-orders. The edge weights are used to make the partitioning algorithm avoid separating elements with strong coupling, thus improving the effectivity of the preconditioner.

The inter-domain communication stores the data in one layer of fictitious elements neighboring each inter-domain boundary. This information is enough for the residual calculation and the assembly of its Jacobian due to the compact stencil of DG discretizations.

Various choices are possible for the node weights. In this work, we choose the weights based on the number of non-zero entries in the self-blocks, *i.e.* the main-diagonal blocks, of the Jacobian matrix,

$$\omega_{\kappa^H} = (p_{\kappa^H} + 1)^{2\cdot\mathrm{dim}}, \tag{5.1}$$

where p_{κ^H} is the polynomial order of element κ^H. Equation 5.1 assumes a tensor-product approximation space, such as the case of this work. When a different approximation space is used, the expression for the node weights may be different but they should adequately represent the solution cost for the corresponding elements.

The edge weights are computed in the following sequence.

1. Loop through edges of the graph and compute:

   $$\omega_{\partial\kappa^H\setminus\partial D} = (p^+_{\kappa^H} + 1)^{\mathrm{dim}} + (p^-_{\kappa^H} + 1)^{\mathrm{dim}}, \tag{5.2}$$

   where $p^+_{\kappa^H}$ and $p^-_{\kappa^H}$ are the polynomial orders of the elements on both sides of the interior face.

2. Loop through edges of the graph that are part of lines of the preconditioner (red node groups in Figure 5.3) and augment $\omega_{\partial\kappa^H\setminus\partial D}$ with

   $$\omega_{\partial\kappa^H\setminus\partial D} \Leftarrow \omega_{\partial\kappa^H\setminus\partial D} \cdot \max(v^+_{\kappa^H}, v^-_{\kappa^H}), \tag{5.3}$$

   where $v^+_{\kappa^H}$ and $v^-_{\kappa^H}$ are the valencies of the nodes, in the connectivity graph, connected by the edge $\partial\kappa^H \setminus \partial D$.

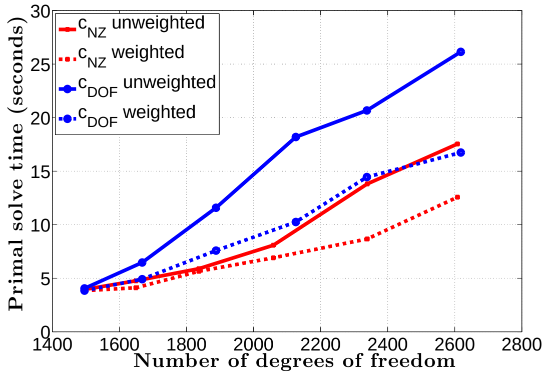
Equation 5.2 gives weights to the edges that are proportional to the amount of data transferred in each exchange of information between processors, that is, the number of degrees of freedom of the fictitious cells used to store the data to be transferred. The second step (Eqn. 5.3) makes the partitioner prefer to separate elements that are not strongly coupled.

The effect of the mesh partitioning algorithms is assessed in an output-adaptive context by comparing runs using the unweighted and weighted partitioning methods. The first case considered is *hp*-adaptation applied to the viscous flow over the NACA 0012 airfoil presented in Section 4.4.1.2. Eight processors are used for this case and Figures 5.4(a) and 5.4(b) show the time taken to solve the flow and adjoint equations in each adaptive step. In the case of weighted partitioning runs, the primal solve time includes the time taken to compute the weights and partition the mesh. Note that the runs with weighted partitioning are significantly faster and, more importantly, the savings increase as the adaptation evolves. This is not surprising, as the unweighted partitioning assumes equal computational cost for each element. Perhaps more interesting, are the savings in number of GMRES iterations shown in Figure 5.4(d). Those savings are mostly due to using the element lines of the preconditioner [19] to assign weights to the edges of the graph.

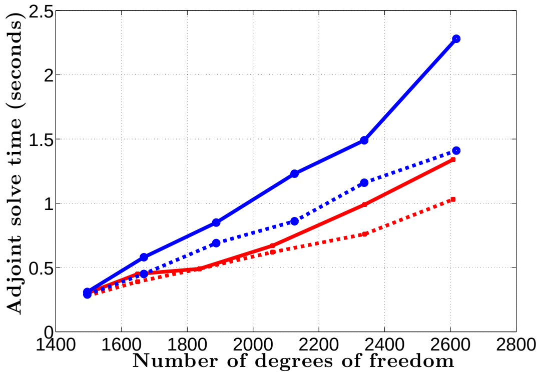
The adaptive process uses block-Jacobi smoothing for the fine-space approximations involved in error estimation, therefore, the savings shown in Figure 5.4(c) are due to a better distribution of computational work amongst the processors.

When the computational work is not well distributed amongst the processors, the overall time consumed by a parallel operation will be dictated by the processor that has the most work load. With an iterative solver, such as the flow solver, it is difficult to estimate *a priori* the total amount of computational work required to converge the solution. Hence, it is important to find adequate representations of computational work imbalance. For this purpose, we analyze the standard deviation of different characteristic quantities of each mesh partition. We then relate large standard deviations with imbalance of a specific quantity in the partition map.

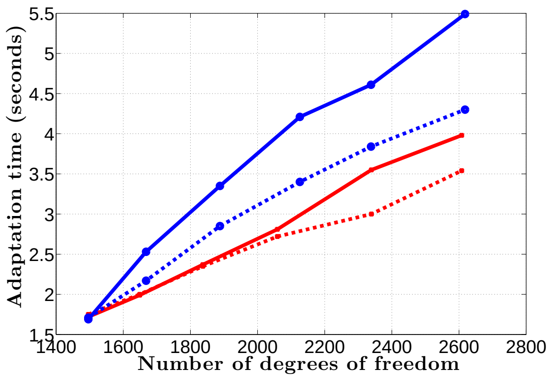
The most trivial characteristic quantity of a mesh partition is the number of elements and its standard deviation: this is shown for each adaptive step in Figure 5.5(b). Note that the unweighted method presents much less variation in the



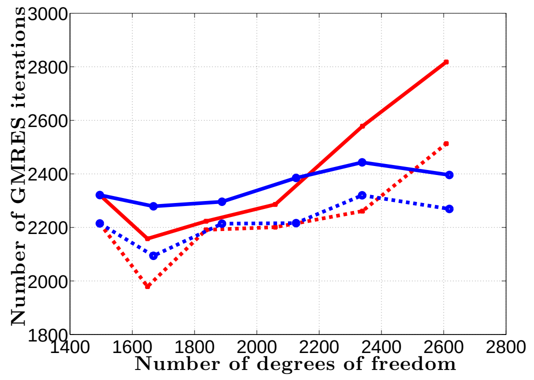
(a) Primal solve time.



(b) Adjoint solve time.



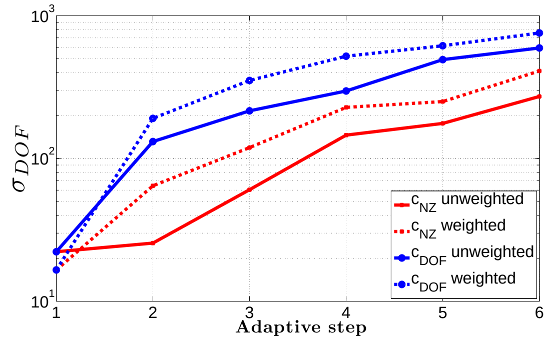
(c) Adaptation time.



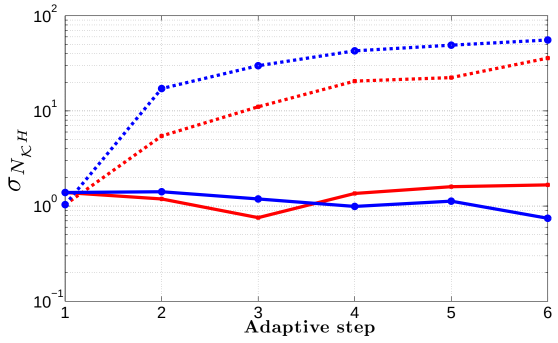
(d) Number of GMRES iterations for primal and adjoint solves.

Figure 5.4: NACA 0012 - $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 1°$, *hp*-adaptation: comparison of runs with unweighted and weighted mesh partitioning. The data points correspond to the adaptation steps

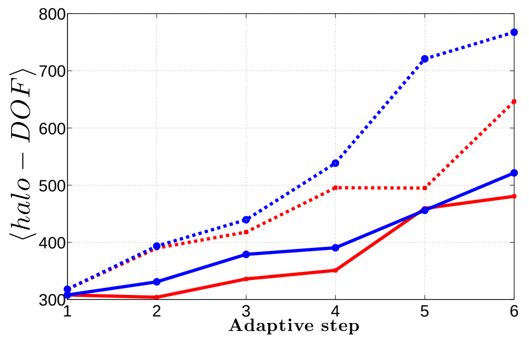
number of elements compared to the weighted method. This is clear evidence that number of elements is not a good measure of computational work in an *hp*-adaptive framework as Figure 5.5(b) does not reflect the savings observed in Figure 5.4.



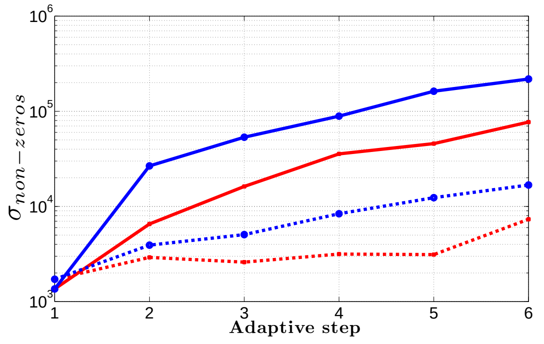
(a) Standard deviation of number of degrees of freedom in different processors.



(b) Standard deviation of number of elements in different processors.



(c) Average number of degrees of freedom used for inter-domain communication.



(d) Standard deviation of number of non-zero entries in the residual Jacobian in different processors.

Figure 5.5: NACA 0012 - $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 1°$, *hp*-adaptation: statistics for the partitioning methods.

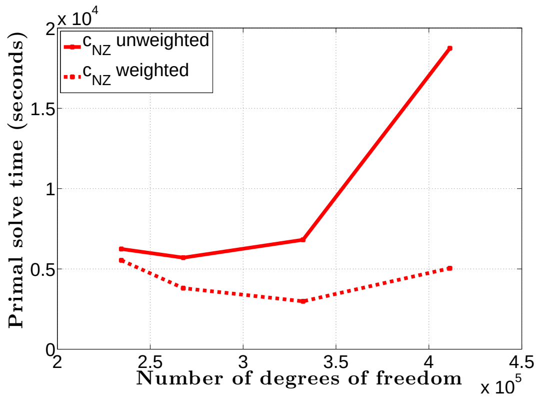
Another measure of computational work is the number of degrees of freedom for which the standard deviation is shown in Figure 5.5(a). Even though this quantity accounts for the elements' orders, it indicates that the unweighted partitions are better balanced than the weighted ones, which clearly disagrees with the performance measures in Figure 5.4.

As discussed previously in this work, the computational effort for solving the flow and adjoint equations is dominated by matrix-vector products in the GMRES algorithm which are dependent on the number of non-zero entries in the Jacobian matrix. The standard deviation for this quantity is shown in Figure 5.5(d). Note that this quantity reflects the time savings observed previously.

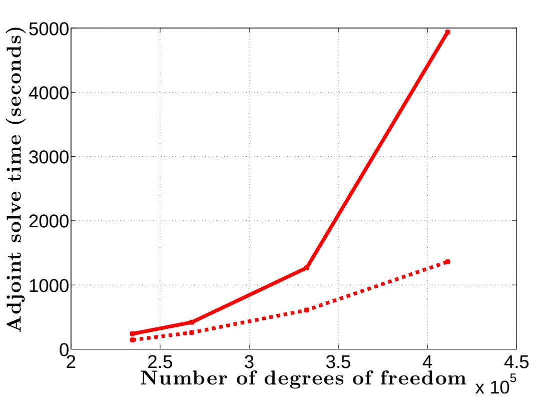
Another important aspect of parallel computations is data transfer between processors. Figure 5.5(c) shows the average number of degrees of freedom correspondent to the fictitious elements used for inter-processor communication. Note that the weighted partitioning leads to more inter-domain communication. This is a result of using preconditioner lines to assign edge weights. However, this does not impact significantly the parallel performance because of the large ratio of computation per communication in the implicit implementation of DG used in this work [19].

To support this analysis, we superimpose, in Figure 5.6, the subdomain boundaries on the mesh and the global preconditioner lines, *i.e.*, the lines of strongly-coupled elements in the ideal case where the entire mesh and data reside in one processor. Note that the subdomains for the weighted partitioning method are generally more anisotropic compared to the unweighted partitioning, especially on the wake region. This anisotropy is responsible for the larger average communication in the weighted-partitioned cases shown in Figure 5.5(c). It is such anisotropy, on the other hand, that makes the subdomains contain more strongly coupled elements resulting in the savings observed in Figure 5.4(d).

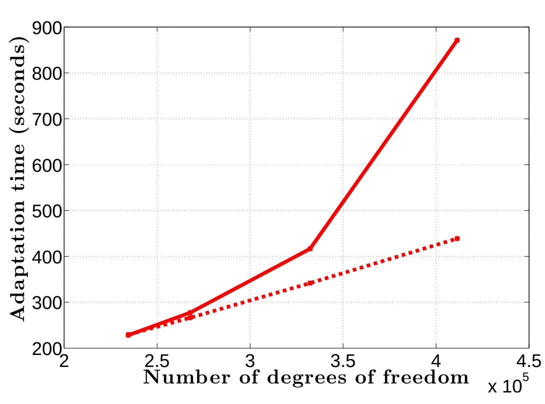
We now analyze the effect of the weighted partitioning algorithm in the three dimensional case of transonic, turbulent flow over the DPW III - Wing 1 presented in Section 4.4.3. At each adaptive step, the computational domain is distributed amongst 720 processors using both the weighted and the unweighted partitioning methods. Figures 5.7(a) and 5.7(b) show the time taken for the primal and adjoint solves respectively. Note that the weight-partitioned run is approximately 70% faster than its unweighted counterpart at the fourth adaptive step. This is because the computational work amongst the unweighted partitions quickly becomes poorly balanced as the *hp*-adaptation progresses.



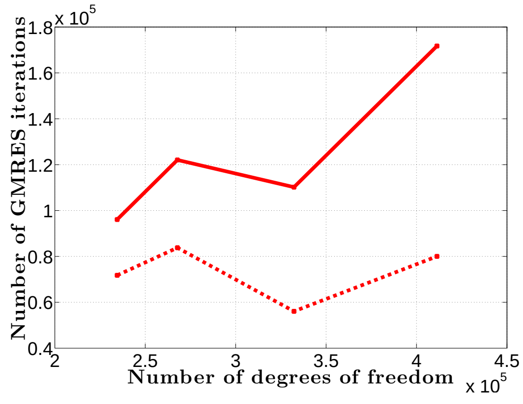
(a) Primal solve time.



(b) Adjoint solve time.



(c) Adaptation time.



(d) Number of GMRES iterations for primal and adjoint solves.

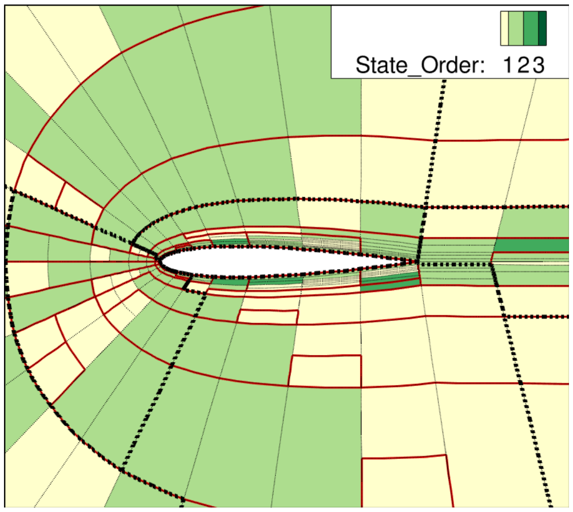
Figure 5.7: DPW III Wing 1 - $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5°$, *hp*-adaptation: comparison of runs with unweighted and weighted mesh partitioning. The data points correspond to the adaptation steps

Another interesting aspect of this test case is that the primal solve time for the second adaptive step is shorter than the initial solve despite the increase in number of degrees of freedom. This is because the flow is initialized with uniform free-stream conditions for the initial solve and, due to the difficulty of this case, the solver takes many nonlinear iterations to converge. In subsequent adaptive iterations, the most recent solution is used as the initial condition.

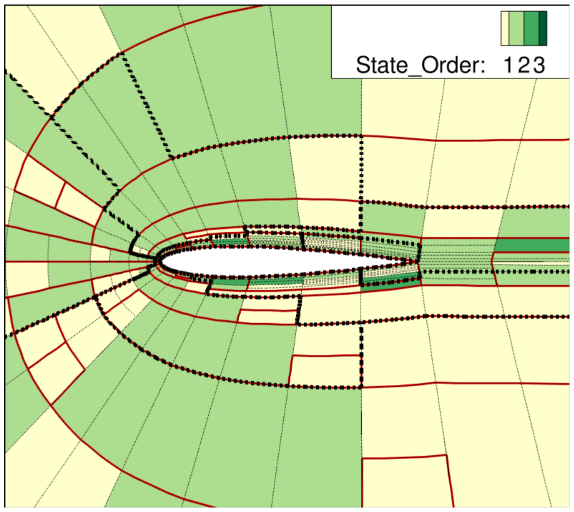
In the weighted-partitioning, the initial primal solve is the longest of all the solves. This is because the preconditioner lines used for the weighted partitioning are computed with the initial condition for each primal solve which, in turn, makes the subsequent weighte-partitioned solves more efficient as they are initialized by the converged state from the previous adaptive iteration.

As observed in the previous case, the adaptation time (Figure 5.7(c)) is also sensitive to the partitioning algorithm. This measure includes the time taken by the parallel adaptation algorithm described in Section 5.1, but it is dominated by the time consumed in the block-Jacobi smoothing iterations involved in error estimation. The computational effort of these iterations is dictated by the number of entries in the self-blocks of the Jacobian matrix (Eqn. 5.1) which explains the savings observed in Figure 5.7(c).

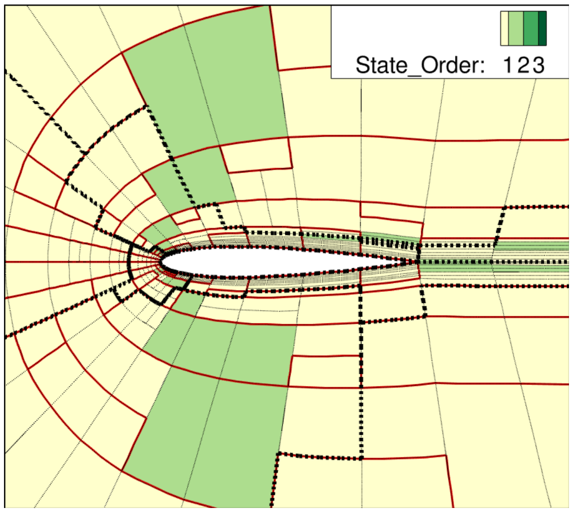
The statistical analysis of the computational work distribution is shown in Figure 5.8. This figure confirms what is observed in the previous case. Specifically, the anisotropic nature of the weighted partitions results in more inter-processor communication when compared to the unweighted partitioning (Figure 5.8(c)). Additionally, Figures 5.8(a) and 5.8(b) respectively confirm that number of degrees of freedom and number of elements are not good measures of computational effort in a *hp*-adaptation framework as uniform distribution of these quantities does not reflect the savings observed in Figure 5.7.



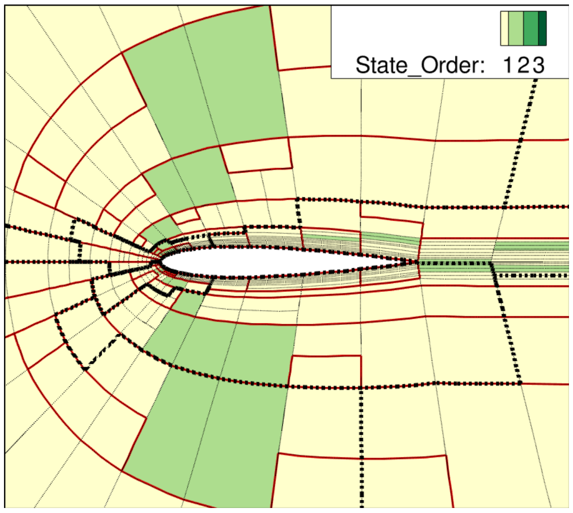
(a) Mesh, element lines, and subdomain boundaries for c_{DOF} with unweighted partitioning.



(b) Mesh, element lines, and subdomain boundaries for c_{DOF} with weighted partitioning.

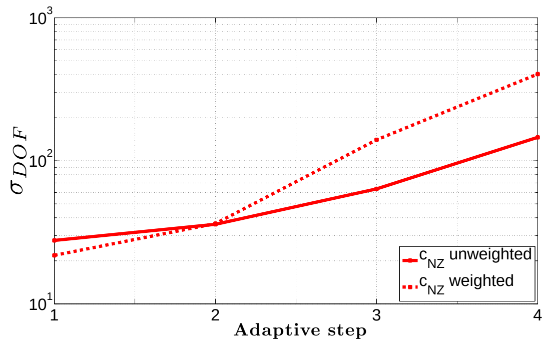


(c) Mesh, element lines, and subdomain boundaries for c_{NZ} with unweighted partitioning.

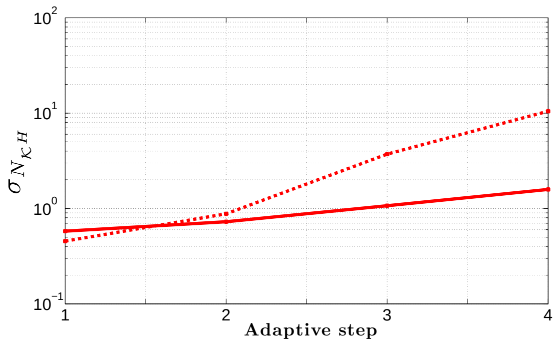


(d) Mesh, element lines, and subdomain boundaries for c_{NZ} with weighted partitioning.

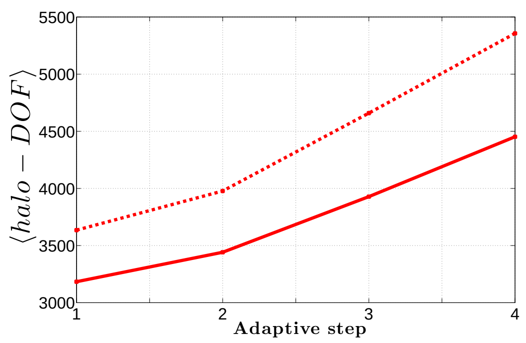
Figure 5.6: NACA 0012 - $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 1^\circ$, 6th adaptation step; mesh: fine dotted black lines; subdomain boundaries: thick dotted black lines; global preconditioner lines: red lines.



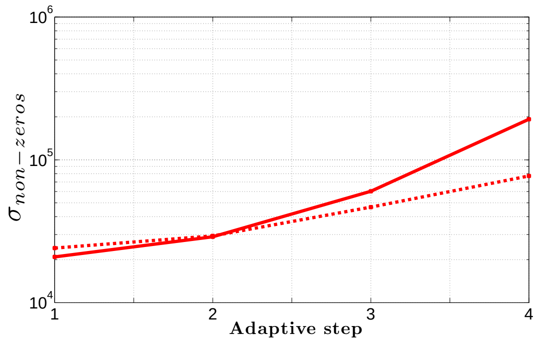
(a) Standard deviation of number of degrees of freedom in different processors.



(b) Standard deviation of number of elements in different processors.



(c) Average number of degrees of freedom used for inter-domain communication.



(d) Standard deviation of number of non-zero entries in the residual Jacobian in different processors.

Figure 5.8: DPW III Wing 1 - $M_\infty = 0.76$, $Re = 5 \times 10^6$, $\alpha = 0.5°$, hp-adaptation: statistics for the partitioning methods.

CHAPTER 6

Conclusions and Future Work

6.1 Summary

This thesis presents an optimization-based *hp*-adaptation method that considers *p*-refinement as a refinement option amongst directional cuts in a element. The refinement options are ranked based on a cost-benefit analysis in which the benefit is an output sensitivity with respect to the different ways of refining the solution space. The cost is estimated considering two measures of CPU work: number of degrees of freedom and number of floating point operations. The latter is correlated to the number of non-zero entries in the residual Jacobian.

The discontinuous Galerkin method is used for the spatial discretization of the flow equations and a physicality-constrained pseudo-transient continuation is proposed for advancing the solution to steady-state. This method includes information of physical realizability constraints in the solution path with the objective of improving robustness by trying to circumvent non-physical regions of the solution space. From an optimization argument that stems from the backward Euler time continuation scheme, the constraints are incorporated in the iterative solution path by a vector penalization technique that makes the prohibited regions of the solution space less attractive for the solver, and thus shifting the solution path away from non-physical states without affecting the final zero-residual solution. A robustness improvement is observed when the method is applied to challenging problems in the aeronautical industry. Furthermore, a line-search method is proposed with the objective of improving the global convergence properties of time continuation methods.

In the implementation realm, we propose a method for assigning weights to the nodes and edges of the connectivity graph used for partitioning the mesh for solution on distributed-memory architectures. The method accounts for the non-homogeneity of computational cost of the elements in the mesh and uses preconditioner information to improve parallel efficiency. A significant speedup is observed, despite the increase in inter-processor communication. This is accomplished by assigning edge weights that make the line-Jacobi preconditioner more globally effective and by using node weights that represent adequately the non-homogeneity of computational cost that arises in *hp*-adaptation routines.

In spite of DG's attractive qualities, its application to practical problems in the aeronautical industry has been somewhat limited due to robustness challenges and high computational cost. The main contribution of this work is to address some of these challenges by demonstrating the applicability of the methods presented here to industry-relevant problems.

6.2 Conclusions

The test-suite results in Chapter 3 show that the inclusion of physicality constraints in the solution path increases the robustness with which the solver can converge the residual in more challenging cases such as the DPW-III wing and the transonic NACA case without shock-capturing. In problems with higher free-stream Mach number, initializing the flow with uniform free-stream conditions produces strong expansions that can lead to cavitation. The constrained solver reduces the impact of such an aggressive flow initialization but it is not a *bullet-proof* method and better flow initialization strategies are certainly desirable.

The proposed optimization-based *hp*-adaptation method is effective in achieving output convergence. This is demonstrated in challenging cases of the aeronautical industry. The method's effectivity is due to the use of adjoint information to guide the refinement decision and the use of measures of computational cost. The cost measures presented here are not perfect as they do not account for stiffness effects of different refinements in the iterative solution method. These effects have a direct impact on the CPU time, but they are difficult and computationally intensive to estimate because they are global measures and their effect depends on the type of solver and preconditioner used. Since the merit function is computed multiple times in each refinement cycle, the local property of cost and benefit measures is attractive.

The two-dimensional adaptation results show savings of up to an order of magnitude in terms of degrees of freedom and up to a factor of two in terms of CPU time when compared to uniform refinement. In three dimensions, *hp*-adaptation using c_{NZ} saves degrees of freedom and CPU time require to achieve output convergence, especially when the output is corrected by its error estimate. Properly accounting for the effect of dimensionality in the computational expense is the reason for c_{NZ} outperforming c_{DOF} in three dimensions.

The weighted mesh-partitioning method accounts for the non-homogeneity of computational cost of the elements in the mesh and uses preconditioner information to improve parallel efficiency. The results show speedup of up to two with the weighted partitioning in three dimensions. Although not yet fully robust due to the possibility of empty partitions, the weighted-partitioning algorithm is an attractive option for the element-line preconditioner.

6.3 Future work

Certain topics for future work were identified during the course of this work. These topics are listed below:

1. **Better flow initialization strategies:** as mentioned previously, the physicality constrained solver reduces the impact of aggressive flow initialization strategies but it is not perfect. Better flow initialization methods can further improve the robustness of the solver.

2. h **and** p **coarsening:** the adaptation mechanics presented in this work increments the solution cost at each adaptive step. The use of coarsening allows for a reallocation of computation cost in the mesh, thus permitting mesh adaptation at approximately-fixed solution cost.

3. **Adaptive node movement:** the performance of the adaptation method presented here depends, to a certain extent, on the initial mesh topology. The ability of moving the nodes in the mesh would reduce the dependence on the initial mesh.

4. μ**-evolution in the constrained solver:** only one strategy for evolving the penalty factor was explored. Other strategies may further improve the robustness of the solver.

5. **Extension to other DG methods:** the methods presented in this work are not exclusively applicable to the nodal DG method used here and they can be extended to more cost-efficient discretizations like *hybridized*-DG.

6. **Weighted-partitioning:** the partitioning method presented in this work is not yet fully robust as empty partitions may occur during the adaptation cycle. In addition, the method for assigning edge weights can be extended to other types of preconditioners.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] ADIGMA – A European project on the development of adaptive higher-order variational methods for aerospace applications. In AIAA, editor, *47th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA 2009-176, 2009.

[2] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1), 1966.

[3] Harold L. Atkins and Alyssa Pampell. Robust and accurate shock capturing method for high-order discontinuos Galerkin methods. In *20th AIAA Computaional Fluid Dynamics Conference*, 2011.

[4] Garrett E. Barter. *Shock Capturing with PDE-Based Artificial Viscosity for an Adaptive Higher–Order Discontinuous Galerkin Finite Element Method*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[5] Timothy Barth and Mats Larson. A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes. In R. Herban and D. Kröner, editors, *Finite Volumes for Complex Applications III*, pages 41–63, London, 2002. Hermes Penton.

[6] F. Bassi and S. Rebay. GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations. In Karniadakis Cockburn and Shu, editors, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, pages 197–208. Springer, Berlin, 2000.

[7] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. In A. Iserles, editor, *Acta Numerica*, pages 1–102. Cambridge University Press, 2001.

[8] Kim S. Bey. *An hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws*. PhD dissertation, University of Texas at Austin, 1994.

[9] H. M. Bucker, B. Pollul, and A. Rasch. On CFL evolution strategies for implicit upwind methods in linearized Euler equations. *International Journal for Numerical Methods in Fluids*, 59:1–18, 2009.

[10] Nicholas K. Burgess and Dimitri J. Mavriplis. An *hp*-adaptive discontinuous Galerkin solver for aerodynamic flows on mixed-element meshes. In *49th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA 2011-490, 2011.

[11] Nicholas K. Burgess and Dimitri J. Mavriplis. Robust computation of turbulent flows using a discontinuous galerkin method. In *50th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA 2012-0457, 2012.

[12] Giuseppe Buttazzo, Aldo Frediani, Steven R. Allmaras, John E. Bussoletti, Craig L. Hilmes, Forrester T. Johnson, Robin G. Melvin, Edward N. Tinoco, Venkat Venkatakrishnan, Laurence B. Wigton, and David P. Young. Algorithm issues and challenges associated with the development of robust CFD codes. In *Variational Analysis and Aerospace Engineering*, volume 33 of *Springer Optimization and Its Applications*, pages 1–19. Springer New York, 2009.

[13] M. J. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaptation for flow simulations. *International Journal for Numerical Methods in Fluids*, 25:475–491, 1997.

[14] Marco Ceze and Krzysztof J. Fidkowski. Output-driven anisotropic mesh adaptation for viscous flows using discrete choice optimization. In *48th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA 2010-170, 2010.

[15] Marco Ceze and Krzysztof J. Fidkowski. A robust adaptive solution strategy for high-order implicit CFD solvers. In *20th AIAA Computaional Fluid Dynamics Conference*, number AIAA 2011-3696. AIAA, 2011.

[16] Bernardo Cockburn, San-Yih Lin, and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws iii: one-dimensional systems. *Journal of Computational Physics*, 84(90-113), 1989.

[17] K. J. Fidkowski, M. A. Ceze, and P. L. Roe. Entropy-based drag error estimation and mesh adaptation in two dimensions. *AIAA Journal of Aircraft*, 49(5):1485–1496, September-October 2012.

[18] K. J. Fidkowski and D. L. Darmofal. A triangular cut–cell adaptive method for high–order discretizations of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 225:1653–1672, 2007.

[19] Krzysztof J. Fidkowski. A high–order discontinuous Galerkin multigrid solver for aerodynamic applications. MS thesis, M.I.T., Department of Aeronautics and Astronautics, June 2004.

[20] Krzysztof J. Fidkowski. *A Simplex Cut-Cell Adaptive Method for High–order Discretizations of the Compressible Navier-Stokes Equations.* PhD dissertation, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2007.

[21] Krzysztof J. Fidkowski and David L. Darmofal. An adaptive simplex cut-cell method for discontinuous Galerkin discretizations of the Navier-Stokes equations. AIAA Paper 2007-3941, 2007.

[22] Krzysztof J. Fidkowski and David L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*, 49(4):673–694, 2011.

[23] L. Formaggia, S. Micheletti, and S. Perotto. Anisotropic mesh adaptation with applications to CFD problems. In H. A. Mang, F. G. Rammerstorfer, and J. Eberhardsteiner, editors, *Fifth World Congress on Computational Mechanics*, Vienna, Austria, July 7-12 2002.

[24] Luca Formaggia and Simona Perotto. New anisotropic a priori error estimates. *Numerische Mathematik*, 89:641–667, 2001.

[25] Luca Formaggia, Simona Perotto, and Paolo Zunino. An anisotropic a posteriori error estimate for a convection-diffusion problem. *Computing and Visualization in Science*, 4:99–104, 2001.

[26] Neal T. Frink. 3rd AIAA CFD drag prediction workshop gridding guidelines. NASA Langley, 2007. http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop3/gridding_guidelines.html.

[27] Neal T. Frink. Test case results from the 3rd AIAA drag prediction workshop. NASA Langley, 2007. http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop3/.

[28] Stefano Giani and Paul Houston. High–order *hp*–adaptive discontinuous Galerkin finite element methods for compressible fluid flows. In Norbert Kroll, Heribert Bieler, Herman Deconinck, Vincent Couaillier, Harmen van der Ven, and Kaare Sørensen, editors, *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, volume 113 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 399–411. Springer Berlin / Heidelberg, 2010.

[29] M. B. Giles and N. A. Pierce. Adjoint equations in CFD: duality, boundary conditions and solution behavior. AIAA Paper 97-1850, 1997.

[30] M. B. Giles and E. Süli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. In *Acta Numerica*, volume 11, pages 145–236, 2002.

[31] Michael B. Giles, Mihai C. Duta, Jens-Dominik Müller, and Niles A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA Journal*, 41(2):198–205, 2003.

[32] Wagdi G. Habashi, Julien Dompierre, Yves Bourgault, Djaffar Ait-Ali-Yahia, Michel Fortin, and Marie-Gabrielle Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles. *International Journal for Numerical Methods in Fluids*, 32:725–744, 2000.

[33] R. Hartmann and P. Houston. Goal-oriented a posteriori error estimation for multiple target functionals. In T.Y. Hou and E. Tadmor, editors, *Hyperbolic Problems: Theory, Numerics, Applications*, pages 579–588. Springer-Verlag, 2003.

[34] Ralf Hartmann. Adjoint consistency analysis of discontinuous Galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[35] Ralf Hartmann and Paul Houston. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. *Journal of Computational Physics*, 183(2):508–532, 2002.

[36] Joachim Hartung. A stable interior penalty method for convex extremal problems. *Numerische Mathematik*, 29(2), 1978.

[37] V. Heuveline and R. Rannacher. Duality-based adaptivity in the *hp*-finite element method. *Journal of Numerical Mathematics*, 11(2):95–113, 2003.

[38] Paul Houston, Emmanuil H. Georgoulis, and Edward Hall. Adaptivity and a posteriori error estimation for DG methods on anisotropic meshes. In *International Conference on Boundary and Interior Layers*, 2006.

[39] Paul Houston and Endre Süli. A note on the design of *hp*-adaptive finite element methods for elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194:229–243, 2005.

[40] A. Jameson, W. Schmidt, and E. Turkel. Numerical simulation of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. In *AIAA 5th Computational Fluid Dynamics Conference*, 1981.

[41] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal for Scientific Computing*, 20(1):359–392, 1998.

[42] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* SIAM, 1995.

[43] C. T. Kelley and David E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 1998.

[44] C. T. Kelley, Li-Zhi Liao, LIqun Qi, Moody T. Chu, J.P. Reese, and C. Winton. Projected pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 46(6):3071–3083, 2008.

[45] C. M. Klaij, J. J. W. van der Vegt, and H. van der Ven. Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 217:589–611, 2006.

[46] Dmitri Kuzmin. A vertex-based hierarchical slope limiter for p-adaptive discontinuous Galerkin methods. *Journal of Computational and Applied Mathematics*, (233), 2010.

[47] Charles L. Ladson. Effects of independent variation of Mach and Reynolds numbers on the low-speed aerodynamic characteristics of the NACA 0012 airfoil section. Technical Memorandum 4074, NASA, 1988.

[48] Kelly R. Laflin, Steven M. Klausmeyer, Thomas Zickuhr, John C. Vassberg, Richard A. Wahls, Joseph H. Morrison, Olaf P. Brodersen, Mark E. Rakowitz, Edward N. Tinoco, and Jean-Luc Godard. Data summary from the second AIAA computational fluid dynamics drag prediction workshop. *Journal of Aircraft*, 42(5), 2005.

[49] David W. Levy, Thomas Zickuhr, John Vassberg, Shreekant Agrawal, Richard A. Wahls, Shahyar Pirzadeh, and Michael J. Hemsch. Data summary from the first AIAA computational fluid dynamics drag prediction workshop. *Journal of Aircraft*, 40(5):875–882, 2003.

[50] James Lu. *An a Posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin Finite Element Method.* PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[51] Charles A. Mader, Joaquim R.R.A. Martins, Juan J. Alonso, and Edwin van der Weide. ADjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA Journal*, 46(4):863–873, 2008.

[52] D. J. Mavriplis. Adaptive mesh generation for viscous flows using Delaunay triangulation. *Journal of Computational Physics*, 90:271–291, 1990.

[53] D. J. Mavriplis and A. Jameson. Hermite-based mesh adaptation for functional outputs improvement in fluid flow simulation. *AIAA Journal*, 47(8):1965–1976, 2009.

[54] James M. Modisette. *An Automated Reliable Method for Two-Dimensional Reynolds-averaged Navier-Stokes Simulations.* PhD dissertation, Massachusetts Institute of Technology, 2011.

[55] Wim A. Mulder and Bram van Leer. Experiments with implicit upwind methods for the Euler equations. *Journal of Computational Physics*, 1985.

[56] National Aeronautics and Space Administration. Turbulence Modeling Resource. NASA Langley, 2012. http://turbmodels.larc.nasa.gov/.

[57] Marian Nemec, Michael J. Aftosmis, and Mathias Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. AIAA Paper 2008-0725, 2008.

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

[59] Todd A. Oliver. *A High–order, Adaptive, Discontinuous Galerkin Finite Elemenet Method for the Reynolds-Averaged Navier-Stokes Equations.* PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[60] K. Oswatitsch. *Gas Dynamics*. Academic Press, New York, 1956.

[61] M. A. Park. Adjoint-based, three-dimensional error prediction and grid adaptation. AIAA Paper 2002-3286, 2002.

[62] Michael A. Park. *Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows*. PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[63] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72:449–466, 1987.

[64] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, number 2006-112, 2006.

[65] W. Rachowicz, D. Pardo, and L. Demkowicz. Fully automatic *hp*-adaptivity in three dimensions. Technical Report 04-22, ICES, 2004.

[66] R. Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128:205–233, 2001.

[67] A. J. Riley and M. V. Lowson. Development of a three-dimensional free shear layer. *Journal of Fluid Mechanics*, 369:49–89, 1998.

[68] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

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

[70] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 7(3):856–869, 1986.

[71] R. Schneider and P. K. Jimack. Toward anisotropic mesh adaptation based upon sensitivity of a posteriori estimates. Technical Report 2005.03, University of Leeds, School of Computing, 2005.

[72] P. Solín and L. Demkowicz. Goal-oriented *hp*-adaptivity for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 193:449–468, 2004.

[73] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*, number AIAA-92-0439. AIAA, 1992.

[74] Edward N. Tinoco, David Levy, and Olaf P. Brodersen. DPW 5 summary of participant data, June 2012. http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/.

[75] Bram van Leer. Towards the ultimate conservative difference scheme. II - monotonicity and conservation combined in a second order scheme. *Journal of Computational Physics*, (14):361–370, 1974.

[76] John C. Vassberg, Mark A. DeHaan, Melissa Rivers, and Richard A. Wahls. Development of a common research model for applied CFD validation studies. In *26th AIAA Applied Aerodynamics Conference*, 2008.

[77] John C. Vassberg, Mark A. DeHaan, and Tony J. Sclafani. Grid generation requirements for accurate drag predictions based on OVERFLOW calculations. In *16th AIAA Computational Fluid Dynamics*, number 2003-4124, 2003.

[78] John C. Vassberg, Edward N. Tinoco, Mori Mani, Olaf P. Brodersen, Bernhard Eisfeld, Richard A. Wahls, Joseph H. Morrison, Thomas Zickuhr, Kelly R. Laflin, and Dimitri J. Mavriplis. Abridged summary of the third AIAA computational fluid dynamics drag prediction workshop. *Journal of Aircraft*, 45(3), 2008.

[79] John C. Vassberg, Edward N. Tinoco, Mori Mani, Ben Rider, Thomas Zickuhr, David W. Levy, Olaf P. Brodersen, Bernhard Eisfeld, Simone Crippa, Richard A. Wahls, Joseph H. Morrison, Dimitri J. Mavriplis, and Mitsuhiro Murayama. Summary of teh fourth AIAA CFD drag prediction workshop. In *28th AIAA Applied Aerodynamics Conference*, 2010.

[80] D. A. Venditti. *Grid Adaptation for Functional Outputs of Compressible Flow Simulations*. PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002.

[81] D. A. Venditti and D. L. Darmofal. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–39, 2002.

[82] D. A. Venditti and D. L. Darmofal. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22–46, 2003.

[83] Venkat Venkatakrishnan. Convergence to steady state solutions of the Euler equations on unstructured grid with limiters. *Journal of Computational Physics*, (118):120–130, 1995.

[84] E.V. Volpe and L.C. de C. Santos. Boundary and internal conditions for adjoint fluid flow problems. *Journal of Engineering Mathematics*, 65(1), 2009.

[85] Masayuki Yano. *An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes*. PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 2012.