# AUTOMATIC MESH ADAPTATION USING THE CONTINUOUS ADJOINT APPROACH AND THE SPECTRAL DIFFERENCE METHOD

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yi Li
March 2013

© 2013 by Yi Li. All Rights Reserved.

Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: http://purl.stanford.edu/zh986nw2266

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Antony Jameson, Primary Adviser**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Robert MacCormack**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Peter Pinsky**

Approved for the Stanford University Committee on Graduate Studies.

**Patricia J. Gumport, Vice Provost Graduate Education**

*This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.*

# Abstract

In computational fluid dynamics (CFD), mesh quality has a direct impact on solution accuracy. Traditional mesh generation relies heavily on 'rules of thumbs' and experience and is time-consuming. Mesh adaptation aims to produce meshes that offer maximum solution accuracy for a given number of degrees of freedom (DOF), by estimating and equi-distributing solution error. This thesis is motivated to develop adaptation methods that are fast, effective and robust.

Among various strategies, adjoint-based error estimation has shown promising results and is widely generalizable to complex problems. The goal is to estimate the error in a functional output of interest, such as lift or drag, by formulating and solving the adjoint problem. There are two approaches to develop the adjoint equations: continuous and discrete. The discrete adjoint equation is derived from discretized flow equations, and this approach has been largely applied and tested to give good performance in mesh adaptation. However, the implementation complexity and the added cost to obtain the adjoint solutions largely depend on the code structure of the flow solver.

This work believes that the continuous adjoint approach overcomes the difficulties in implementation and cost of solving the adjoint equations. To date, continuous adjoint has not been applied to mesh adaptation problems. Since the continuous adjoint equation is derived from the original flow equations, the adjoint solver is independent from the flow solver. It is also shown in this work that the adjoint solution accuracy is not crucial for mesh adaptation. This means that potentially, a fast and universal adjoint solver can be designed and used with any flow solver.

While the usual choices for the targeted outputs are lift and drag coefficients

etc, Fidkowski and Roe proposed a set of entropy variables and proved that they automatically satisfy the adjoint equation developed using the net entropy flow out of the domain. In this entropy adjoint approach, no adjoint equation needs to be solved, leading to a further saving of computational cost.

In this thesis, a continuous adjoint solver is implemented using the high order spectral difference method for 2D Euler equations. The goal is to study the performance of continuous adjoint method in mesh adaptation, and in comparison with the discrete adjoint approach used in the existing literature. Comparison is also made between $h$- and $p$-adaptation, in search of an optimal combintation adaptation strategy. The entropy adjoint method is also studied and compared with the continuous drag adjoint. It is found that $h$-adaptation in general is more suitable for singularities in the flow while $p$-adaptation is more efficient for sufficiently smooth flows. Both provide adapted meshes with higher solution accuracy than uniformly refined meshes for a given number of DOFs. The entropy adjoint and the drag adjoint yield very similar results, too, suggesting that the entropy adjoint can be a cheap alternative to adjoints targeting other outputs. Overall, the continuous adjoint approach provides comparable performance to the discrete adjoint, but with the advantages of freedom in implementation and choices of trade-off between cost and accuracy.

# Acknowledgements

I am extremely grateful for those who have guided and helped me during my PhD study at Stanford, making my pursuit of the degree both possible and memorable. Foremost, I would like to thank my academic advisor, Professor Antony Jameson. During my first trip to Stanford on the departmental visit day in April 2008, Professor Jameson's extreme kindness and intelligence immediately made me decide to come to Stanford, and I was very lucky to join the Aerospace Computing Lab thereafter. In research, Professor Jameson offered me much freedom in exploring topics of interest and encouraged me to work on projects of my passion. He had always been very accessible and generous in sharing his inspiring ideas. I am truly grateful for his guidance and help throughout my PhD.

I would also like to thank my oral examination committee and my dissertation reading committee. Professor Robert MacCormack is the most humble professor I have met. Professor Peter Pinsky is an excellent lecturer and I enjoyed greatly his Finite Element Method class. Professor Thomas Pulliam works at NASA yet still kindly agreed to be a member of my oral exam committee. Last but not least, Professor Michael Saunders not only chaired my oral exam, but also gave me mental support in the hardest times before my exam. I sincerely thank all faculty members of my committee for the time they had taken and the suggestions they had made to improve my dissertation work.

I would also like to thank Stanford Graduate Fellowship for three years of financial support, in particular the Larry C.K. Yung Graduate Student Support Fund.

In addition, I wish to express my gratitude to all the lab-mates in the Aerospace Computing Lab. Charlie Liang and Sachin Premasuthan gave me tremendous help to

get my PhD project started. Qiqi Wang offered me very useful insights in the adjoint method. Yves Allaneau and I worked together on several projects and he gave me the most advice among all. I also enjoyed great times in the lab and at conferences with other lab members including Patrice Castonguay, Andre Chan, Edmond Chiu, Matthew Culbreth, Rui Hu, Jen-Der Lee, Guido Lodato, Manuel Lopez, Kui Ou, Peter Vincent and David Williams.

During the past five years, I made some of the most important friends of my life. I want to thank Ting, Ming, Can, Frank, among others, for making my PhD life colorful and enjoyable.

None of these would have been possible without the support and love from my parents and the rest of my families throughout my life. In particular I am grateful for my late grandfather, whose persistence and passion in learning inspired me until today. Above all, I would like to thank Yang Xu, my husband, who has always been there to share my joy in the good times and enlighten me in the hard times despite the long-distance segregation.

# Contents

**Abstract** **iv**

**Acknowledgements** **vi**

**1 Introduction** **1**

1.1 Motivation . . . . . 1

1.1.1 Mesh adaptation and error estimation . . . . . 2

1.1.2 Continuous vs discrete adjoint . . . . . 4

1.2 Outline . . . . . 6

**I Mesh Adaptation and Error Estimation** **7**

**2 Mesh Adaptation** **8**

2.1 $h$, $p$, $hp$ Adaptation . . . . . 9

2.1.1 $h$-adaptation . . . . . 9

2.1.2 $p$-adaptation . . . . . 10

2.1.3 $hp$-adaptation . . . . . 10

2.2 Adaptation Indicator . . . . . 11

**3 Feature-Based Error Indicator** **12**

3.1 A Discontinuity Sensor . . . . . 12

**4 Adjoint-Based Error Estimation** **14**

4.1 A Posteriori Error Estimator . . . . . 14

4.2 The Continuous Adjoint Equation . . . . . 15
4.3 The Discrete Adjoint Approach . . . . . 17

**5 The Entropy Adjoint Approach** **20**
5.1 For Euler Equations . . . . . 22

**II Implementation** **23**

**6 Spectral Difference Method** **24**
6.1 Governing Equations . . . . . 25
6.2 The Transformed Equations . . . . . 26
6.3 The Basis Function and the Reconstructed Polynomials . . . . . 27
6.4 Implementation in 2D . . . . . 28
6.5 Boundary Conditions . . . . . 29
6.5.1 Far field . . . . . 29
6.5.2 Wall surface . . . . . 30

**7 Adjoint Solver** **31**
7.1 The Adjoint Equations for Euler Equations . . . . . 31
7.1.1 Flux Jacobian and functionals of interest . . . . . 31
7.1.2 Adjoint equation . . . . . 32
7.2 The SD Adjoint Solver . . . . . 34
7.2.1 The transformed equation . . . . . 34
7.2.2 Implementation in 2D . . . . . 35
7.2.3 Interface values of $\psi$ . . . . . 35
7.2.4 Boundary conditions in 2D . . . . . 36

**8 Mortar Elements** **37**
8.1 $p$-Adaptation . . . . . 38
8.1.1 Face to mortar . . . . . 38
8.1.2 Mortar to face . . . . . 39
8.1.3 Mortar procedure . . . . . 39

8.2 $h$-Adaptation . . . 40
8.2.1 Face to mortar . . . 40
8.2.2 Mortar to face . . . 41
8.2.3 Mortar procedure . . . 42

**9 Adaptation Procedure** **43**
9.1 Error Indicator for an Element . . . 43
9.2 $h$- and $p$-adaptation . . . 44
9.3 $hp$-adaptation . . . 44

**III Test Cases** **46**

**10 Test Cases** **47**
10.1 Subsonic Flow Past a Half-Cylinder with Entropy Adjoint Approach . 47
10.1.1 $h$-refinement . . . 48
10.1.2 $p$-refinement . . . 49
10.1.3 $p$-refinement/coarsening . . . 49
10.1.4 Comparison of $h$- and $p$-refinement . . . 56
10.2 Subsonic Flow Past an Airfoil . . . 57
10.2.1 $h$-Refinement . . . 57
10.2.2 $p$-Refinement . . . 64
10.2.3 Drag convergence plot . . . 64
10.3 Transonic Flow Past an Airfoil . . . 65
10.3.1 $h$-Refinement . . . 65
10.3.2 $hp$-Refinement . . . 70
10.3.3 Drag convergence plot . . . 73
10.4 Output Functionals . . . 73
10.5 Cost of the Adjoint Solver . . . 74

**11 Conclusions and Future Work** **80**
11.1 Summary and Conclusions . . . 80
11.2 Future Work . . . 83

**Bibliography** **84**

# List of Tables

# List of Figures


2.1 Isotropic and Anisotropic $h$-refinement on a quadrilateral grid. . . . . 10

6.1 Transformation to standard elements. . . . . 26
6.2 A SD standard element in 2D, with $N = 3$. . . . . 28

8.1 Mortar configuration for $p$-refinement. . . . . 39
8.2 Mortar configuration for $h$-refinement. . . . . 42

10.1 Cylinder, $M = 0.3$. Initial mesh and Mach contour. . . . . 48
10.2 Cylinder, $M = 0.3$. Estimated error distribution and adapted meshes, $h$-refinement, $N = 3$. . . . . 50
10.3 Cylinder, $M = 0.3$. Estimated error distribution and adapted meshes, $h$-refinement, $N = 4$. . . . . 51
10.4 Cylinder, $M = 0.3$. Estimated error distribution and adapted order distribution, $p$-refinement, $N = 3$. . . . . 52
10.5 Cylinder, $M = 0.3$. Estimated error distribution and adapted order distribution, $p$-refinement, $N = 4$. . . . . 53
10.6 Cylinder, $M = 0.3$. Estimated error distribution and adapted order distribution, $p$-refinement/coarsening, $N = 3$. . . . . 54
10.7 Cylinder, $M = 0.3$. Estimated error distribution and adapted order distribution, $p$-refinement/coarsening, $N = 4$. . . . . 55
10.8 Cylinder, $M = 0.3$. Drag convergence plot. . . . . 56
10.9 NACA 0012, $M = 0.4$, $\alpha = 5°$. Initial mesh and Mach contour. . . . . 58

10.10NACA 0012, $M = 0.4$, $\alpha = 5°$. Estimated error distribution and adapted meshes, $h$-refinement using entropy adjoint. . . . . 59
10.11NACA 0012, $M = 0.4$, $\alpha = 5°$. Estimated error distribution and adapted meshes, $h$-refinement using drag adjoint. . . . . 60
10.12NACA 0012, $M = 0.4$, $\alpha = 5°$. Adapted meshes obtained by Fidkowski *et al.* using discrete adjoint. . . . . 61
10.13NACA 0012, $M = 0.4$, $\alpha = 5°$. Estimated error distribution and adapted order distribution, $h$-refinement using entropy adjoint. . . . . 62
10.14NACA 0012, $M = 0.4$, $\alpha = 5°$. Estimated error distribution and adapted order distribution, $h$-refinement using drag adjoint. . . . . 63
10.15NACA 0012, $M = 0.4$, $\alpha = 5°$. Drag convergence plot. . . . . 64
10.16NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Initial mesh and Mach contour. . . 65
10.17NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Mach contour on the adapted mesh. 66
10.18NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Estimated error distribution and adapted meshes, $h$-refinement using entropy adjoint. . . . . 67
10.19NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Estimated error distribution and adapted meshes, $h$-refinement using drag adjoint. . . . . 68
10.20NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Drag adjoint solution. . . . . 69
10.21NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Estimated error distribution and adapted meshes, $hp$-refinement using entropy adjoint. . . . . 71
10.22NACA 0012, $M = 0.8$, $\alpha = 1.25°$. The discontinuity sensor plot for $hp$-refinement. . . . . 72
10.23NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Drag convergence plot. . . . . 73
10.24NACA 0012, $M = 0.4$, $\alpha = 5°$. Meshes after three levels of $h$-refinement. 75
10.25NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Meshes after three levels of $h$-refinement. . . . . 76
10.26NACA 0012, $M = 0.4$, $\alpha = 5°$. Drag convergence plot for different adjoint output functionals. . . . . 77
10.27NACA 0012, $M = 0.4$, $\alpha = 5°$. Lift convergence plot for different adjoint output functionals. . . . . 78

10.28NACA 0012, NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Drag convergence plot for different adjoint output functionals. . . . . 78
10.29NACA 0012, NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Lift convergence plot for different adjoint output functionals. . . . . 79

# Chapter 1

# Introduction

## 1.1 Motivation

In the past several decades, computational fluid dynamics (CFD) has played an indispensable role in the aerospace and related industries. The design and testing of Boeing 787 Dreamliner took 800,000 hours of computing time on Cray supercomputers and 15,000 hours of wind tunnel tests [5]. While the number of wind tunnel testing hours has not changed substantially since the design of the first Boeing 747, the number of prototype wings and nacelles tested in the wind tunnel is reduced from over a hundred to less than 10 [8]. With the fast advancing computational power, the computers are able to simulate problems that are increasing in size and complexity, for example vortical and turbulent flows. Ongoing research is developing more sophisticated solvers that can better make use of the new technologies (e.g. graphical processing units) to deliver superior performance in terms of accuracy and complexity of problems.

A typical CFD simulation involves the following steps: computer aided design (CAD) of a geometry, discrete mesh generation, and computation of the flow solution. In most practical circumstances, mesh generation takes days or weeks, compared to hours to days needed for a typical flow calculation. For a very complex geometry such as that of a Formula One car, mesh generation could involve weeks of work from the most experienced personnel in the area, because the process is labor intensive and

lack of automation.

On top of the time and cost required, the mesh quality is also of great importance. In the Drag Prediction Workshops (DPW) run by the American Institute of Aeronautics and Astronautics [27, 26, 52, 32, 53], the drag values obtained by participating groups consist of discrepancies that could have significant meanings in real design. It is also observed that the discrepancies are not significantly improved by the increased computing power over the years. Other than choices of discretization schemes and choices of sub-grid models, it is believed that different choices of mesh also yield different numerical results, even for the same number of degrees of freedom (DOF) [32]. Distribution of DOFs can significantly affect solution accuracy. Also, when a mesh is uniformly refined (i.e. increasing the number of DOFs while not changing their distribution), $L^2$ error convergence rate agrees with the order of the discretization scheme, but functional error convergence rate does not [2]. In order to achieve a better functional convergence, the distribution of DOFs needs to be altered, e.g. through mesh adaptation.

A desirable mesh should provide a high solution accuracy for a given number of DOFs. Even for the same geometry, the desirable meshes may look very different under different flow conditions. For example, the optimal meshes for a wing in a subsonic flow and in a supersonic flow are distinctive from each other, particularly in the regions of shock presence. The goal is to develop an automatic mesh generation process which is capable of dealing with complex geometries as well as complex flows, while producing an optimal mesh for solution accuracy.

### 1.1.1 Mesh adaptation and error estimation

Mesh adaptation has gained attention and popularity in the past two decades because it tackles the above issues described for mesh generation. The idea is to start with a coarse mesh, calculate an error indicator from the numerical solution, and adapt the mesh accordingly. Mesh adaptation aims to decrease and equidistribute the error, thereby achieving a certain level of accuracy with minimum cost. Extensive research has been done in mesh adaptation for both structured and unstructured mesh [1,

3, 9, 39, 40, 48, 54, 56, 58], in which error estimation is the key. Various error indicators have been proposed and tested before, including feature detection, error of discretization, residual indicator and adjoint-based error indicator.

Feature detection is based on the assumption that regions of flow features (e.g. shocks or vortices) are associated with large errors. While this approach is effective for certain problems [4, 58], difficulties arise for more complex problems. Firstly, for complex flow structures and interactions, feature detection becomes non-trivial and less accurate. Secondly, the assumption that regions of flow features are associated with large errors is not always true. It is possible that some seemingly benign regions are also error-prone. An example is that, if the cells upstream of a shock are causing errors in the solution, then the location of the shock in the numerical solution may not be accurate. In this case it is more helpful to adapt the cells upstream of the shock than those at the shock (which is not at the location of the shock in the true solution).

Discretization error arises from discretization of the continuous flow equations. Since the true solution is not known, the discretization error needs to be estimated by some solution reconstruction process. Residual indicator is also a cheap and effective adaptation indicator for simple and low dimensional flows. Zhang *et al.* compared these different error indicators and found that neither of them perform well for multi-dimensional flows [60].

The most popular approach nowadays is the adjoint-based error estimation. In engineering applications, there is more interest in the error in a functional output, such as lift or drag, than local errors. The adjoint-based method specifically estimates the error in an output of interest, and relates it to local residuals. It makes use of the duality concept to formulate an adjoint problem of the primal problem. Giles and Pierce first used the adjoint solutions for error correction for integral outputs [13, 14, 42]. Venditti *et al* applied it to one-dimensional and multi-dimensional Euler equations [55, 56]. Various people then extended it to multi-dimensional Navier-Stokes equations [39, 38]. So far the method has proved to be robust and effective, and it is applicable to general discretization schemes.

Mesh adaptation is usually done via local addition or removal of DOFs. The three classes of adaptation strategies are $h$-adaptation, $p$-adaptation and $hp$-adaptation. A fourth type, named $r$-adaptation, describes the relocation of current DOFs without adding or removing any of them, and is also called mesh movement. It is less commonly used due to its limitations and complications in multi-dimensions.

$h$- and $p$-adaptations both have advantages and limitations. $p$-adaptation provides an exponential error convergence rate with respect to DOF for smooth solutions [9], but requires a good physical mesh to start with, and cannot deal with flow discontinuities. $h$-adaptation has the advantage of possible anisotropy, as studied in [6, 59]. $hp$-adaptation is the combination of the two and merges the benefits of them. Oden *et al* and Rachowicz *et al* proposed strategies in designing $hp$ meshes, which have been applied to elliptic problems [46, 37, 7, 50, 18, 47].

In this thesis, both $h$- and $p$-adaptations are used and compared. $hp$-adaptation is also briefly attempted.

### 1.1.2 Continuous vs discrete adjoint

The adjoint method was first exploited in CFD for aerodynamic design optimization using control theory by Jameson [21, 22, 24]. There are two approaches to develop the adjoint equations: continuous and discrete. The continuous adjoint equation is formulated as the dual problem of the original flow problem, while the discrete adjoint equation is the dual problem of the discretized flow problem. The continuous adjoint equation is then discretized and solved independently from the flow equations. A comparison between the two approaches in the context of design optimization was studied by Nadarajah and Jameson [34, 35, 36]. In general, the differences in the results obtained from discrete and continuous adjoint are small, and discrete adjoint converges to continuous adjoint as mesh sizes are reduced. The derivation for discrete adjoint for explicit low order schemes (such as the finite volume schemes) can be very complicated and tedious. In addition, Nadarajah has noted that the adjoint gradient calculated for optimization is insensitive to the convergence of the flow and adjoint solutions.

In the context of error estimation and mesh adaptation, the differences between discrete and continuous adjoint are also worth studying, as well as whether the phenomena in optimization still hold true for mesh adaptation. So far, discrete adjoint has been widely used for mesh adaptation [10] and shown promising performance, but no one has applied continuous adjoint in this context [28]. One concern is whether continuous adjoint is capable of handling singularities in flow solutions. Despite that, little difference is expected in the results by continuous and discrete adjoint.

The implementation complexity of discrete adjoint largely depends on the flow discretization scheme as well as the implementation of the flow solver. If the full Jacobian matrix and an associated linear solver are available from the flow solver, then the discrete adjoint solutions can be obtained relatively easily. On the other hand, if the Jacobian matrix is not stored, as in the case of most explicit solvers, then the implementation of the adjoint solver can be very complex. In contrast, this is not an issue for continuous adjoint, as the adjoint equation is just an independent PDE that can be discretized and solved using any scheme of choice. Potentially, one adjoint solver can be used with any flow solvers.

In terms of the computational cost needed to obtain the adjoint solutions, continuous adjoint has the freedom to adopt a fast solver in expense of some accuracy. The cost of the discrete adjoint solution is again determined by the code structure of the flow solver. The added cost is small if the Jacobian matrix is available, and large otherwise.

The advantages of continuous adjoint has motivated this thesis to investigate the performance of continuous adjoint in mesh adaptation.

While lift, drag and moment coefficient are the common output of interest in engineering applications, in 2009, Fidkowski and Roe proposed an entropy adjoint approach for Euler and Navier-Stokes equations which measures the error in the net entropy flow out of the domain [11, 12]. A set of entropy variables have been proposed to automatically satisfy the continuous adjoint equation and its boundary conditions. That is, the adjoint solution can be obtained directly from the flow solution without the need for solving the adjoint equation. It is also shown that the entropy adjoint

gives comparable performance to discrete adjoint targeting other common outputs. In this work, the entropy adjoint method is also tested.

## 1.2 Outline

The thesis is organized into three parts. The first part (Chapter 2-5) talks about the theory behind mesh adaptation and error estimation, and derives the continuous and the discrete adjoint equations and error estimators. The entropy adjoint approach is also described. The second part (Chapter 6-8) consists of the implementation details of the flow solver and the continuous adjoint solver for Euler equations in 2D, using the high order spectral difference (SD) method. Chapter 8 also describes the use of mortar elements for non-conforming interfaces resulted from mesh adaptation. Lastly, the third part (Chapter 10) presents and discusses the test results.

# Part I

# Mesh Adaptation and Error Estimation

# Chapter 2

# Mesh Adaptation

In computational fluid dynamics, mesh generation determines the spatial allocation of degrees of freedom (DOFs) in the flow domain. Intuitively, an efficient mesh should be able to capture as many details as possible in the flow, while at the same time not overly resolved in the regions with mostly uniform flow. For the same number of DOFs, the mesh quality can significantly affect the solution accuracy. However, intelligent allocation of DOFs usually requires a lot of prior knowledge about the targeted flow problem. In practice, many man hours are needed to generate a satisfactory mesh for a complex problem such as the flow past a Formula-1 car.

Mesh adaptation tackles this difficulty in mesh generation beautifully. The process starts with a coarse mesh which is easy to create. A local adaptation indicator can then be computed from the numerical solution on the coarse mesh. DOFs would be added or removed locally according to this indicator, resulting in an 'adapted mesh'. This process can be repeated on the adapted mesh until a desired solution accuracy is reached. Because no prior knowledge of the problem is needed, the adaptation process is called 'automatic'.

Two issues arise from automatic mesh adaptation, namely how to compute the adaptation indicator and how to add or remove DOFs in a mesh. This chapter focuses on addressing the latter issue in this chapter, and the former is examined in the next chapter.

## 2.1 $h$, $p$, $hp$ Adaptation

There are two ways of adding DOFs: mesh refinement and order enrichment. The latter is only applicable to high order schemes such as Discontinuous Galerkin (DG) for obvious reasons. We usually use $h$ to denote the mesh width and $p$ to denote the order of the polynomials representing the numerical solution. Therefore we also name the two methods as $h$-refinement and $p$-refinement respectively. Correspondingly, there are $h$-coarsening and $p$-coarsening where DOFs are removed. $h$-coarsening is often not used, because we start with a coarse mesh and there should be no need to remove any DOFs in the process. $p$-coarsening, on the other hand, could be advantageous at flow discontinuities, where the solution is best represented by a linear function instead of a high order polynomial which results in oscillations in the solution profile.

### 2.1.1 $h$-adaptation

Figure 2.1 illustrates two ways of $h$-refinement on a two-dimensional quadrilateral cell. Isotropic refinement is independent of directions, and anisotropic refinement is dependent on the direction, which makes it more suitable in situations such as across a shockwave, where the flow gradient in one direction is significantly different from that in the other direction. Either way, the refinement results in some nodes on edges that are not vertices of all the cells sharing those edges, i.e. the "hanging nodes". Special treatments called "mortar elements" at the interfaces are needed for these hanging nodes, and it will be discussed in detail in Chapter 8. Three-dimensional hexahedral cells can be $h$-refined in similar ways.

$h$-refinement on a two-dimensional triangular mesh may take a slightly more complex strategy to keep the mesh free of hanging nodes. In this work we will not go into details of $h$-refinement of triangular or tetrahedral meshes.

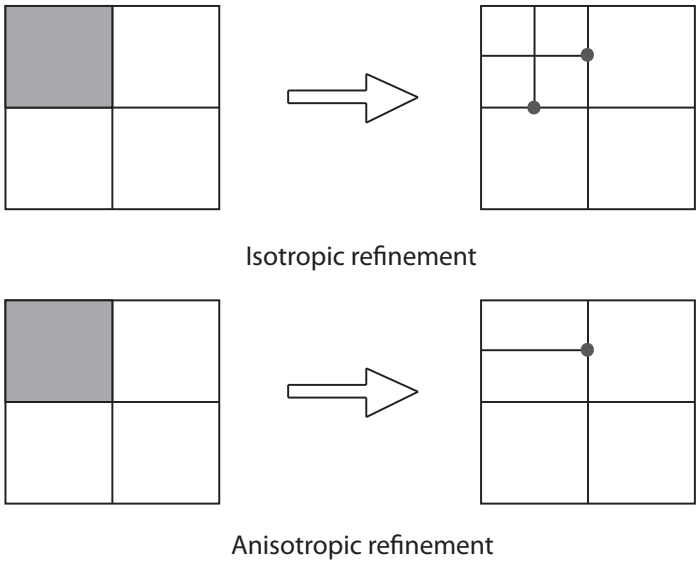



Figure 2.1: Isotropic and Anisotropic $h$-refinement on a quadrilateral grid.

### 2.1.2 $p$-adaptation

For low order methods that are prevalent in commercial softwares today, order enrichment is not possible. However, as high order methods gain more popularity, $p$-adaptation can become very useful in certain situations. In discontinuous high order methods such as the spectral difference method (SD), the solution values on the interfaces are discontinuous, and a common flux needs to be obtained for each pair of flux points. However, $p$-adaptation results in interfaces across which the two neighboring cells have unequal number of flux points on their faces. In such situations, mortar elements are needed as discussed in Chapter 8. (Note that $p$-adaptation on either triangular or quadrilateral meshes will result in such interfaces and need such mortar elements.)

### 2.1.3 $hp$-adaptation

$hp$-adaptation is simultaneous $h$- and $p$-adaptation. A mesh can be both $h$- and $p$-adapted at the same. Therefore decision needs to be made on if an element would be $h$- or $p$-adapted. It has been observed that $p$-refinement offers more benefits in

regions of smooth flows [29] (flows without discontinuities), whereas it is obvious that $h$-refinement would better model a flow discontinuity. Therefore a simple strategy is to $h$-refine the mesh near flow discontinuities and $p$-refine elsewhere.

## 2.2 Adaptation Indicator

The adaptation indicator identifies the regions to be refined or coarsened. Two most popular categories of adaptation strategies are called feature-based and output-based adaptation. The first category uses feature detection methods and refines in areas with flow features such as shocks or vortices. It is fast and easy to implement, but it is usually not robust enough for complex geometries. The second category targets a specific output of interest (such as lift or drag) and derives a posteriori error estimator for this output. Adaptation is then done according to the estimated error distribution. This method involves deriving and solving the so-called 'adjoint' equations, and is therefore also referred to as the adjoint-based adaptation.

The next two chapters will describe these two strategies, with more emphasis and details on the output-based approach.

# Chapter 3

# Feature-Based Error Indicator

Very often, the accuracy of a numerical flow solution is reflected by how well the main flow features such as shocks and vortices are captured. Therefore an intuitive adaptation strategy is to refine around flow features in order to capture them better. Feature detection can be done via measurements such as solution gradients or curvatures. The advantage of these methods are low computational cost and implementation simplicity. The disadvantage is its non-robustness. For complex structures, a mixture of various types of flow features may exist at the same time and accurate detection is difficult. Also, the method does not adapt based on an estimation of local solution errors. This means that some cells with large solution errors may not appear as the targets for refinement.

## 3.1 A Discontinuity Sensor

Here we give an example of a flow discontinuity sensor which will be used in this work for shock detection in *hp*-refinement. The discontinuity sensor was introduced by Persson and Peraire [41] in 2006 for shock detection in a high order finite element scheme, where the solution within each element is expressed in terms of a hierarchical family of orthogonal polynomials. For example, in one-dimension, the solution is represented by an expansion in terms of orthonormal Legendre polynomials. If the solution is smooth, the coefficients in the expansion would decay quickly. The idea is

to compare the coefficient of the polynomial basis of the highest order with the rest of the coefficients.

In one-dimension, the solution of order $p$ can be expressed as

$$u = \sum_{i=0}^{p} u_i P_i \tag{3.1}$$

where $P_i$'s are the Legendre basis functions. If we consider a truncated expansion of the same solution,

$$\hat{u} = \sum_{i=0}^{p-1} u_i P_i \tag{3.2}$$

Then the discontinuity sensor for each element $\Omega_k$ is defined as

$$S_k = \frac{(u - \bar{u}, u - \bar{u})_{\Omega_k}}{(u, u)_{\Omega_k}} \tag{3.3}$$

In two- and three-dimensions, $u$ and $\hat{u}$ can be expressed as

$$u = \sum_{i=1}^{N(p)} u_i \phi_i \tag{3.4}$$

$$\hat{u} = \sum_{i=1}^{N(p-1)} u_i \phi_i \tag{3.5}$$

# Chapter 4

# Adjoint-Based Error Estimation

The adjoint-based approach can both provide a local adaptation indicator and give an error estimation for the targeted output of interest.

In this chapter, we will first derive an *a posteriori* error estimator and the continuous adjoint equation. Then we will describe and compare the the discrete adjoint approach with the continuous adjoint approach.

## 4.1 A Posteriori Error Estimator

Consider a differential equation

$$\mathbf{R}(\mathbf{u}) = 0 \tag{4.1}$$

and an output functional $J(\mathbf{u})$. The goal is to find the sensitivity of $J$ to some disturbance $\delta\mathbf{R}$ in the residual.

Let $\mathbf{u_H}$ be the solution of the discretized problem (i.e. $\mathbf{R_H}(\mathbf{u_H}) = 0$ where $\mathbf{R_H}$ is the discretized differential operator on a mesh $H$) and define

$$\delta\mathbf{R} = \mathbf{R}(\mathbf{u_H}) - \mathbf{R}(\mathbf{u}) \tag{4.2}$$

$$\delta J = J(\mathbf{u_H}) - J(\mathbf{u}) \tag{4.3}$$

Since $\mathbf{u}$ is the exact solution of Equation (4.1), it follows that $\delta\mathbf{R} = \mathbf{R}(\mathbf{u_H})$. We can

linearize the above relations to get a first order approximation of the disturbances:

$$\begin{aligned} \delta\mathbf{R} &= \nabla_{\mathbf{u}}\mathbf{R}(\mathbf{u_H})(\mathbf{u_H}-\mathbf{u}) + \text{H.O.T.} \\ &\approx \nabla_{\mathbf{u}}\mathbf{R}(\mathbf{u_H})\delta\mathbf{u} \qquad (4.4) \\ \delta J &= \nabla_{\mathbf{u}}J(\mathbf{u_H})(\mathbf{u_H}-\mathbf{u}) + \text{H.O.T.} \\ &\approx \nabla_{\mathbf{u}}J(\mathbf{u_H})\delta\mathbf{u} \qquad (4.5) \end{aligned}$$

with $\delta\mathbf{u} = \mathbf{u_H} - \mathbf{u}$. Relation (4.4) can be written in weak form:

$$\forall\psi, \quad \langle\psi, \delta\mathbf{R}\rangle = \langle\psi, \nabla_{\mathbf{u}}\mathbf{R}(\mathbf{u_H})\delta\mathbf{u}\rangle \qquad (4.6)$$

Subtracting relation (4.6) from (4.5) leads to

$$\begin{aligned} \forall\psi, \quad \delta J - \langle\psi, \delta\mathbf{R}\rangle &= \nabla_{\mathbf{u}}J(\mathbf{u_H})\delta\mathbf{u} - \langle\psi, \nabla_{\mathbf{u}}\mathbf{R}(\mathbf{u_H})\delta\mathbf{u}\rangle \\ &= [\nabla_{\mathbf{u}}J(\mathbf{u_H}) - \langle\psi, \nabla_{\mathbf{u}}\mathbf{R}(\mathbf{u_H})\rangle] \cdot \delta\mathbf{u} \end{aligned}$$

If we can find $\psi_{\mathbf{H}}$ such that the right hand side vanishes, then we have

$$\boxed{\delta J = \langle\psi_{\mathbf{H}}, \delta\mathbf{R}\rangle = \langle\psi_{\mathbf{H}}, \mathbf{R}(\mathbf{u_H})\rangle} \qquad (4.7)$$

Therefore, $\psi_{\mathbf{H}}$ has to satisfy the 'adjoint equation'

$$\boxed{[\nabla_{\mathbf{u}}J(\mathbf{u_H}) - \langle\psi_{\mathbf{H}}, \nabla_{\mathbf{u}}\mathbf{R}(\mathbf{u_H})\rangle] \cdot \delta\mathbf{u} = 0} \qquad (4.8)$$

## 4.2 The Continuous Adjoint Equation

Let us now focus on the adjoint equation. Consider the following conservation law

$$\mathbf{R}(\mathbf{u}) = \frac{\partial\mathbf{F}_i}{\partial x_i}(\mathbf{u}) = 0 \qquad (4.9)$$

where summation over a repeated index is implied. Let $\mathbf{u_H}$ be the discrete solution to Equation (4.9) on mesh $H$. Linearize the residual around $\mathbf{u_H}$,

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}}|_{\mathbf{u_H}} = \frac{\partial}{\partial x_i} A_i \tag{4.10}$$

where

$$A_i = \frac{\partial \mathbf{F}_i}{\partial \mathbf{u}}|_{\mathbf{u_H}}$$

Given a unit vector $n = [n_1, n_2, \cdots]^T$, we define

$$\mathbf{F} = \mathbf{F}_i \cdot n_i \quad \text{and} \quad \frac{\partial \mathbf{F}}{\partial \mathbf{u}} = A_i \cdot n_i$$

Let us consider an output functional in the form

$$J(\mathbf{u}) = \int_{\partial\Omega} M(\mathbf{u})\, \mathrm{d}\sigma + \int_{\Omega} P(\mathbf{u})\, \mathrm{d}\Omega \tag{4.11}$$

We can derive the adjoint equation as follows. Starting with equation (4.8), we have

$$\begin{aligned}
0 &= \left[\frac{\partial J}{\partial \mathbf{u}} - \langle \psi, \frac{\partial \mathbf{R}}{\partial \mathbf{u}} \rangle\right] \cdot \delta\mathbf{u} \\
&= \left[\int_\Omega \frac{\partial P}{\partial \mathbf{u}}\, \mathrm{d}\Omega + \int_{\partial\Omega} \frac{\partial M}{\partial \mathbf{u}}\, \mathrm{d}\sigma - \int_\Omega \psi^T \frac{\partial A_i}{\partial x_i}\, \mathrm{d}\Omega\right] \cdot \delta\mathbf{u} \\
&= \left[\int_\Omega \frac{\partial P}{\partial \mathbf{u}}\, \mathrm{d}\Omega + \int_{\partial\Omega} \frac{\partial M}{\partial \mathbf{u}}\, \mathrm{d}\sigma - \int_{\partial\Omega} \psi^T A_i n_i\, \mathrm{d}\sigma + \int_\Omega \frac{\partial \psi^T}{\partial x_i} A_i\, \mathrm{d}\Omega\right] \cdot \delta\mathbf{u} \\
&= \left[\int_\Omega \left(\frac{\partial P}{\partial \mathbf{u}} + \frac{\partial \psi^T}{\partial x_i} A_i\right)\, \mathrm{d}\Omega\right] \cdot \delta\mathbf{u} + \int_{\partial\Omega} \left(\frac{\partial M}{\partial \mathbf{u}} - \psi^T A_i n_i\right) \cdot \delta\mathbf{u}\, \mathrm{d}\sigma
\end{aligned} \tag{4.12}$$

Canceling the first integrand in (4.12) leads to the actual adjoint equation satisfied by $\psi$

$$\frac{\partial P}{\partial \mathbf{u}} + \frac{\partial \psi^T}{\partial x_i} A_i \;\; = \;\; 0$$

Very often, the functional of interest does not depend on domain terms ($P = 0$) and

the adjoint equation reduces to

$$\boxed{A_i^T \frac{\partial \psi}{\partial x_i} = 0, \quad \text{on } \Omega} \tag{4.13}$$

The second term in (4.12) is treated as a Dirichlet boundary condition on $\psi$

$$\boxed{\left( \frac{\partial M}{\partial \mathbf{u}} - \psi^T A_i n_i \right) \cdot \delta \mathbf{u} = 0 \quad \text{on } \partial\Omega} \tag{4.14}$$

At this point, it is tempting to cancel out $\delta \mathbf{u} = \mathbf{u_H} - \mathbf{u}$, however, as we will see for the Euler equations, conserving this term can lead to dramatic simplifications in the derivations of the boundary conditions. We can therefore rewrite (4.14)

$$\psi^T \delta \mathbf{F} = \delta M \quad \text{on } \partial\Omega \tag{4.15}$$

where

$$\delta \mathbf{F} = \mathbf{F}(\mathbf{u_H}) - \mathbf{F}(\mathbf{u})$$
$$\delta M = M(\mathbf{u_H}) - M(\mathbf{u})$$

## 4.3 The Discrete Adjoint Approach

The approach described in the previous sections is called the continuous adjoint approach, and Equation (4.8) and (4.13) are called the continuous adjoint equation. In this approach, the adjoint equation and its boundary conditions are derived from linearizing the original differential equation (4.1) (also called the primal equation) and the output functional. The adjoint equation (4.13) can then be discretized and solved using any numerical scheme, which could be independent from the discretization scheme used on the primal equations.

In another approach, called the discrete adjoint approach, the primal equations are discretized first. The adjoint equation is then derived from linearizing the discretized

primal equations. Consider Equation (4.1) discretized on mesh $h$,

$$\mathbf{R_h}(\mathbf{u_h}) = 0 \tag{4.16}$$

where $\mathbf{u_h}$ is the discrete solution on mesh $h$. For a consistent discretization scheme, $\mathbf{u_h} \to \mathbf{u}$ as the mesh width goes to 0. Let $\mathbf{u_H}$ be the discrete solution on a coarser mesh $H$ and define

$$\begin{aligned} \delta\mathbf{R_h} &= \mathbf{R_h}(\mathbf{u_H}) - \mathbf{R_h}(\mathbf{u_h}) \\ \delta J_h &= J_h(\mathbf{u_H}) - J_h(\mathbf{u_h}) \end{aligned}$$

where $J_h$ is the output operator on mesh $h$. Linearize the above relations:

$$\delta\mathbf{R_h} \approx \nabla_{\mathbf{u_h}}\mathbf{R_h}(\mathbf{u_H})(\mathbf{u_H} - \mathbf{u_h}) \tag{4.17}$$

$$J_h \approx \nabla_{\mathbf{u_h}} J_h(\mathbf{u_H})(\mathbf{u_H} - \mathbf{u_h}) \tag{4.18}$$

Again, writing (4.17) in weak form and subtract it from (4.18) gives

$$\forall\psi, \quad \delta J_h - \langle\psi, \delta\mathbf{R_h}\rangle = [\nabla_{\mathbf{u_h}} J_h(\mathbf{u_H}) - \langle\psi, \nabla_{\mathbf{u_h}}\mathbf{R_h}(\mathbf{u_H})\rangle] \cdot (\mathbf{u_H} - \mathbf{u_h})$$

If we can find $\psi_\mathbf{h}$ such that the right hand side vanishes, then we have and error estimate

$$\boxed{\delta J_h = \langle\psi_\mathbf{h}, \delta\mathbf{R_h}\rangle = \langle\psi_\mathbf{h}, \mathbf{R_h}(\mathbf{u_H})\rangle} \tag{4.19}$$

and $\psi_\mathbf{h}$ has to satisfy now the *discrete* adjoint equation:

$$\boxed{[\nabla_{\mathbf{u_h}} J_h(\mathbf{u_H}) - \langle\psi_\mathbf{h}, \nabla_{\mathbf{u_h}}\mathbf{R_h}(\mathbf{u_H})\rangle] \cdot (\mathbf{u_H} - \mathbf{u_h}) = 0} \tag{4.20}$$

In practice, to save the cost of solving for $\psi_\mathbf{h}$ on the fine mesh, it could be approximated by $\psi_\mathbf{H}^\mathbf{h}$, which is the coarse mesh adjoint solution interpolated onto the fine mesh. The implementation of the discrete adjoint solve varies depending on the code structure. If the full Jacobian matrix ($\nabla_{\mathbf{u_h}}\mathbf{R_h}(\mathbf{u_H})$) has been calculated and stored, then it is relatively easy to compute the adjoint solution. Otherwise, the

implementation could be very complex.

Both the continuous and the discrete adjoint approaches are expected to give similar results for error estimation. In this work, the continuous adjoint approach will be used for easier implementation.

# Chapter 5

# The Entropy Adjoint Approach

The Euler equations and Navier-Stokes equations are systems of conservation laws which possess an 'entropy function'[15]. Fidkowski and Roe[12] made use of the symmetrization properties of the entropy functions and discovered a set of entropy variables that automatically satisfy an adjoint equation. This saves the cost of solving the adjoint equations and provides a cheap error indicator for mesh adaptation.

Recall the conservation law (4.9)

$$\mathbf{R}(\mathbf{u}) = \frac{\partial \mathbf{F}_i}{\partial x_i}(\mathbf{u}) = 0 \tag{5.1}$$

and an output functional that only depends on the boundary integrals ($P = 0$ in (4.11))

$$J(\mathbf{u}) = \int_{\partial\Omega} M(\mathbf{u})\, \mathrm{d}\sigma \tag{5.2}$$

and the adjoint equation and its boundary conditions derived previously:

$$A_i^T \frac{\partial \psi}{\partial x_i} = 0 \quad \text{on } \Omega \tag{5.3}$$

$$\left(\frac{\partial M}{\partial \mathbf{u}} - \psi^T A_i n_i\right) \cdot \delta\mathbf{u} = 0 \quad \text{on } \partial\Omega \tag{5.4}$$

A scalar function $U(\mathbf{u})$ is an entropy function for (5.1) if

1. $U$ satisfies the 'compatibility relation'

$$U_{\mathbf{u}} A_i = (f_i)_{\mathbf{u}}$$

where $A_i = (\mathbf{F}_i)_{\mathbf{u}}$ is the flux Jacobian and $f_i$ is some scalar function called entropy flux in the $x_i$ direction. The subscript $\mathbf{u}$ represents a first order derivative in $\mathbf{u}$, i.e. $(\cdot)_{\mathbf{u}} = (\partial/\partial\mathbf{u})(\cdot)$.

2. $U$ is a convex function of $\mathbf{u}$.

Define the entropy variables as $\mathbf{v} = U_{\mathbf{u}}^T$. It was then proved in [15] that the entropy variables satisfy the following:

- $\mathbf{u}_{\mathbf{v}}$ is symmetric and positive definite;
- $A_i \mathbf{u}_{\mathbf{v}}$ is symmetric.

Using these properties, (5.1) can be rewritten as follows:

$$\begin{aligned} 0 = \frac{\partial \mathbf{F}_i}{\partial x_i}(\mathbf{u}) = A_i \frac{\partial \mathbf{u}}{\partial x_i} = A_i \mathbf{u}_{\mathbf{v}} \frac{\partial \mathbf{v}}{\partial x_i} = \mathbf{u}_{\mathbf{v}} A_i^T \frac{\partial \mathbf{v}}{\partial x_i} \\ \Rightarrow A_i^T \frac{\partial \mathbf{v}}{\partial x_i} = 0 \end{aligned}$$

i.e. $\mathbf{v}$ satisfies the adjoint equation (5.3). Additionally, in order to satisfy the boundary condition (5.4),

$$\begin{aligned} 0 = M_{\mathbf{u}} - \mathbf{v}^T A_i n_i = M_{\mathbf{u}} - U_{\mathbf{u}} A_i n_i = M_{\mathbf{u}} - (f_i)_{\mathbf{u}} n_i = (M - f_i n_i)_{\mathbf{u}} = 0 \\ \Rightarrow J = \int_{\partial\Omega} f_i n_i \, \mathrm{d}\sigma \end{aligned}$$

Therefore, the entropy variables $\mathbf{v}$ are the adjoint solution to an output $J$ which measures the net entropy flow out of the domain. As a result, areas of spurious entropy generation will be targeted for adaptation.

## 5.1 For Euler Equations

Fidkowski and Roe proposed a choice for $U$ and the corresponding $\mathbf{v}$ which will symmetrize both the Euler (which will be discuss in Section 6.1) and the N-S equations:

$$\begin{aligned} U &= -\frac{\rho S}{\gamma - 1} \\ \mathbf{v} &= \left[\frac{\gamma - S}{\gamma - 1} - \frac{\rho u_i u_i}{2p}, \quad \frac{\rho u_i}{p}, \quad -\frac{\rho}{p}\right]^T \end{aligned} \tag{5.5}$$

where $S = \ln p - \gamma \ln \rho$ is the physical entropy, $\rho, p, u_i$ are density, pressure and velocity, and these flow quantities will also be defined in Section 6.1. A repeated index implies summation. The corresponding entropy flux is $f_i = u_i U$.

# Part II

# Implementation

# Chapter 6

# Spectral Difference Method

With the significantly improved computer power, high-order numerical methods have been made possible to model multi-dimensional flow problems. High-order methods have proved to be less dissipative than low-order methods and can therefore offer better accuracy in simulation of flows such as viscous flows and vortex dominated flows. Some existing high-order methods include the discontinuous Galerkin (DG) method, its variations such as the nodal DG method, the spectral difference (SD) method, and the flux reconstruction (FR) method, etc. Despite their advantage of providing more accurate flow solutions, these high-order schemes are not yet widely accepted in academia, and even much less in industry. They are generally more complicated to implement, with slow convergence rate and less robust for complex geometries and flows with discontinuities.

The spectral difference method is based on the differential form of the conservative equations. The absence of volume or surface integrals makes the SD implementation easier than the DG method. The scheme was first introduced by Kopriva *et al.* in 1996, as the 'conservative staggered-grid Chebyshev multi-domain method'[25]. It was then extended by Liu *et al.*[30], who first called it the spectral difference method, Wang *et al.*[57], who implemented it for simplex cells for 2D Euler equations, May *et al*[33], who applied it to 2D N-S equations, and Sun *et al.*[51], who implemented it for 3D hexahedral cells for 3D N-S equations. Premasuthan *et al.* then investigated and developed convergence acceleration techniques and shock capturing with artificial

viscosity[44, 45, 43]. Jameson in 2010 gave a proof for energy stability of the SD method for linear PDEs for all orders of accuracy[23].

This work uses the SD method for unstructured quadrilateral meshes for 2D compressible Euler equations and the adjoint equations, with high-order explicit Runge-Kutta time marching schemes. This chapter describes the implementation of the flow solver, and the next chapter will describe the adjoint solver.

## 6.1 Governing Equations

The Euler equations in conservative form can be written as

$$\frac{\partial w}{\partial t} + \frac{\partial F_i}{\partial x_i} = 0 \tag{6.1}$$

In 3D, the vector of conserved variables $w$ and the flux vectors $F_i$ are given by

$$w = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{pmatrix}, \quad F_i = \rho u_i \begin{pmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ E \end{pmatrix} + p \begin{pmatrix} 0 \\ \delta_{i1} \\ \delta_{i2} \\ \delta_{i3} \\ u_i \end{pmatrix} \tag{6.2}$$

where $\rho$ is the density of the fluid, $u_i$ is the flow velocity in the direction of $x_i$, $p$ is the pressure and $E$ is the specific total energy, which can be expressed in terms of specific internal energy, $e$, as

$$E = e + \frac{u_i u_i}{2}, \quad e = C_v T \tag{6.3}$$

where $C_v$ is the specific heat at constant volume and $T$ is the temperature of the fluid. The repeated index $i$ implies summation over all $i$'s. For an ideal gas,

$$C_v = \frac{R}{\gamma - 1}, \quad p = \rho R T$$

where $R$ is the specific gas constant and $\gamma$ is the ratio of specific heats. For air, $R$ is about $287 Jkg^{-1}K^{-1}$ and $\gamma$ is about 1.4. Therefore $p$ can be related to the conservative variables as

$$p = (\gamma - 1)\left[\rho E - \frac{1}{2}\rho u_i u_i\right] \tag{6.4}$$

## 6.2 The Transformed Equations

For an unstructured quadrilateral or hexahedral mesh, each element in the physical domain $\{x_i\}$ is firstly transformed into a standard element in the computational domain $\{\xi_i\} \in [0,1]^d$, where $d$ is the dimension. Equation (6.1) in the physical domain is transformed into the computational domain as follows:

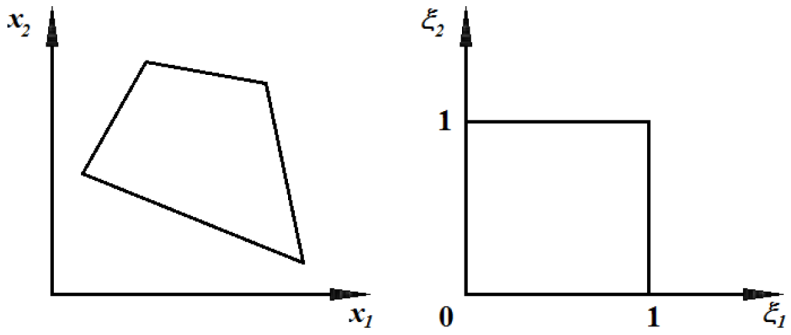



Figure 6.1: Transformation from $(x_1, x_2)$ to $(\xi_1, \xi_2)$.

$$\frac{\partial \tilde{w}}{\partial t} + \frac{\partial \tilde{F}_i}{\partial \xi_i} = 0 \tag{6.5}$$

where

$$\tilde{w} = |J| \cdot w, \quad \tilde{F}_i = |J| \cdot [J^{-1}]_{ij} F_j, \quad J = \frac{\partial \mathbf{x}}{\partial \xi} \tag{6.6}$$

## 6.3 The Basis Function and the Reconstructed Polynomials

Two staggered sets of points are defined in the standard element, namely the solution points and the flux points. The values of the conservative variables are calculated at the solution points, and the values of the flux vectors are computed at the flux points. In the following context, the conservative variables are also referred to as the solution variables. In each dimension, the number of solution points is one less than the number of flux points, such that the order of the reconstructed solution polynomial is one less than that of the reconstructed flux polynomial. Therefore the flux derivatives ($\partial \tilde{F}_i / \partial \xi_i$) are of the same order as the solution polynomial ($\tilde{w}$).

In each dimension, in order to reconstruct a solution polynomial of order $N-1$, $N$ solution points and $N+1$ flux points are needed. The solution points ($\xi_s$)are defined to be at the Gauss-Lobatto points:

$$\xi_s = \frac{1}{2}\left[1 - \cos\left(\frac{2s-1}{2N} \cdot \pi\right)\right], \quad s = 1, 2, ..., N \tag{6.7}$$

and the flux points ($\xi_{s+\frac{1}{2}}$) at the Legendre-Gauss-Quadrature points to ensure stability of the scheme, as suggested by Huynh[19] and proved by Jameson[23].

The basis functions in 1D are the Lagrange polynomials $\{h_i\}_{i=1,...,N} \subset \mathcal{P}_{N-1}$ interpolated from the solution points, and $\{l_{i+\frac{1}{2}}\}_{i=0,...,N} \subset \mathcal{P}_N$ interpolated from the flux points, i.e.

$$\begin{aligned} h_i(\xi) &= \prod_{\substack{s=1 \\ s \neq i}}^{N} \left(\frac{\xi - \xi_s}{\xi_i - \xi_s}\right) \\ l_{i+\frac{1}{2}}(\xi) &= \prod_{\substack{s=0 \\ s \neq i}}^{N} \left(\frac{\xi - \xi_{s+\frac{1}{2}}}{\xi_{i+\frac{1}{2}} - \xi_{s+\frac{1}{2}}}\right) \end{aligned}$$

The reconstructed solution and flux polynomials in higher dimensions are therefore just the tensor products of these 1D polynomials.

## 6.4 Implementation in 2D

A standard element in 2D with solution points and flux points is illustrated in Figure 6.2. The reconstructed solution polynomial in the standard element is

$$w(\xi_1, \xi_2) = \sum_{j=1}^{N} \sum_{i=1}^{N} \frac{\tilde{w}_{i,j}}{|J_{i,j}|} \cdot h_i(\xi_1) \cdot h_j(\xi_2) \tag{6.8}$$

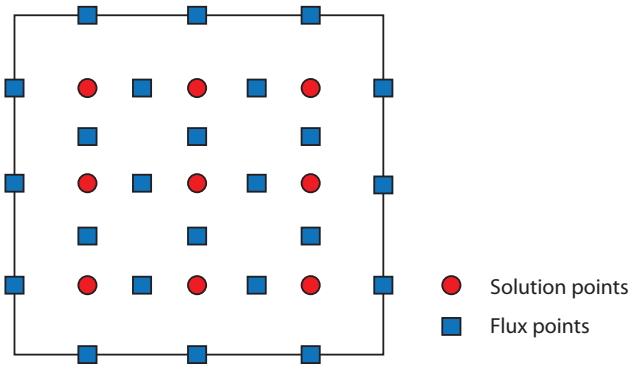



Figure 6.2: A SD standard element in 2D, with $N = 3$.

Similarly, the reconstructed flux polynomials are

$$\tilde{F}_1 = \sum_{j=1}^{N} \sum_{i=0}^{N} \tilde{F}_{1_{i+\frac{1}{2},j}} \cdot l_{i+\frac{1}{2}}(\xi_1) \cdot h_j(\xi_2) \tag{6.9}$$

$$\tilde{F}_2 = \sum_{i=1}^{N} \sum_{j=0}^{N} \tilde{F}_{2_{i,j+\frac{1}{2}}} \cdot h_i(\xi_1) \cdot l_{j+\frac{1}{2}}(\xi_2) \tag{6.10}$$

The reconstructed solution polynomials are continuous within each element, but discontinuous at the element interfaces. Therefore an approximate Riemann solver is needed to compute a common flux at each interface flux points. In this work, a Rusanov solver [49] is used. At each time step, the residual needed to update the solution vectors ($\partial\tilde{w}/\partial t$) is computed as follows:

1. Given the solution variables at the solution points, the solution variables are computed at the flux points using Equation (6.8).

2. The fluxes are computed at the interior flux points.

3. The fluxes at the interface flux points are computed using the Rusanov Riemann solver.

4. The flux derivatives at the solution points are calculated by differentiating Equation (6.9):

$$\left(\frac{\partial \tilde{F}_1}{\partial \xi_1}\right)_{i,j} = \sum_{j=1}^{N}\sum_{i=0}^{N} \tilde{F}_{1_{i+\frac{1}{2},j}} \cdot l'_{i+\frac{1}{2}}(\xi_1) \cdot h_j(\xi_2) \tag{6.11}$$

$$\left(\frac{\partial \tilde{F}_2}{\partial \xi_2}\right)_{i,j} = \sum_{i=1}^{N}\sum_{j=0}^{N} \tilde{F}_{2_{i,j+\frac{1}{2}}} \cdot h_i(\xi_1) \cdot l'_{j+\frac{1}{2}}(\xi_2) \tag{6.12}$$

5. Eventually, the residual at the solution points is

$$\left(\frac{\partial \tilde{w}}{\partial t}\right)_{i,j} = -\left(\frac{\partial \tilde{F}_1}{\partial \xi_1}\right)_{i,j} - \left(\frac{\partial \tilde{F}_2}{\partial \xi_2}\right)_{i,j} \tag{6.13}$$

## 6.5 Boundary Conditions

In this section we discuss the inviscid boundary condition treatment at a far field boundary and at a solid wall boundary.

### 6.5.1 Far field

The far field boundary treatment is the same as that introduced in [20]. At a subsonic far field boundary, the characteristics boundary condition is used to prevent outgoing waves being reflected back into the flow. The treatment is based on the introduction of Riemann invariants for flow normal to the boundary. Let subscripts $\infty$ and $e$ denote free stream values and values extrapolated from interior cells at the boundary respectively, and let $V_n$ and $c$ be the normal velocity to the boundary and the speed

of sound. The Riemann invariants are

$$\begin{aligned} R_\infty &= (V_n)_\infty - \frac{2}{\gamma - 1} c_\infty \\ R_e &= (V_n)_e + \frac{2}{\gamma - 1} c_e \end{aligned}$$

corresponding to incoming and outgoing waves. From these two equations, the normal velocity and the speed of sound at the boundary are given as

$$\begin{aligned} V_n &= \frac{R_\infty + R_e}{2} \\ c &= \frac{\gamma - 1}{4}(R_\infty - R_e) \end{aligned}$$

and the tangential velocity $V_t$ and entropy $S$ are specified as free stream values at inflow boundaries, and as extrapolated interior values at outflow boundaries. All flow quantities at the boundary can be determined consequently. In particular, the density and pressure at the boundary are given as

$$\begin{aligned} \rho &= \left(\frac{c^2}{\gamma S}\right)^{\frac{1}{\gamma - 1}} \\ p &= S\rho^\gamma \end{aligned}$$

At a supersonic far field boundary, the flow quantities are specified as free stream values if it is an inflow boundary, and as extrapolated interior values if it is an outflow boundary.

### 6.5.2 Wall surface

It has been suggested by Lu *et al.* that dual consistent boundary conditions lead to better convergence in functionals and more accurate functional error estimates[31, 17]. A scheme is dual (adjoint) consistent if the discretization leads to a discrete adjoint problem that is a consistent approximation of the corresponding continuous adjoint problem. In this work the dual consistent wall boundary conditions introduced by Hartmann for the DG method are used[16].

# Chapter 7

# Adjoint Solver

This chapter firstly develops the continuous adjoint equations for Euler equations and discusses the boundary conditions, and then presents the implementation of the adjoint solver in 2D with the spectral difference method.

## 7.1 The Adjoint Equations for Euler Equations

### 7.1.1 Flux Jacobian and functionals of interest

For the Euler equations as stated in Equation (6.1), define the Jacobians of the flux

$$A_i = \frac{\partial F_i}{\partial w}$$

Let $F$ be the flux across a boundary of normal $n$. Therefore, we have

$$F = F_i n_i = u_n \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ e \end{pmatrix} + p \begin{pmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ u_n \end{pmatrix}$$

where $u_n$ is the normal velocity $u_n = u_i n_i$. Note that if this boundary is a solid wall, the euler boundary conditions impose $u_n = 0$ and $F = p\,[0, n_1, n_2, n_3, 0]^T$.

Whether the flow is computed around a single airfoil, a full aircraft or even an automobile, engineers are interested in values such as the lift or the drag produced by the object. These values are good candidates for the functional $J$ as they do not depend on domain values leading to the important simplification $P = 0$ in expression (4.11).

Define a unit vector $i_D$ oriented in the direction of the far field flow $V_\infty$ and the normal to it $i_L$. The drag generated on the body $\partial\Omega_w$ by the pressure is given by

$$D = -\int_{\partial\Omega_w} p \;\; n \cdot i_D \;\; d\sigma \tag{7.1}$$

where $n$ is the local unit normal to the wall. Similarly, lift is given by

$$L = -\int_{\partial\Omega_w} p \;\; n \cdot i_L \;\; d\sigma \tag{7.2}$$

### 7.1.2 Adjoint equation

We give here the complete derivation of the adjoint equation and its boundary conditions for the case where the functional of interest is the drag. The derivations for other output functions can be done in a similar fashion.

#### Domain terms

In this case, the drag does not depend on domain terms and $P = 0$. The adjoint equation is given by (4.13)

$$A_i^T \frac{\partial \psi}{\partial x_i} = 0, \quad \text{on } \Omega$$

#### Boundary conditions

We consider boundary conditions in the form given by Equation (4.15):

$$\psi^T \delta F = \delta M \quad \text{on } \partial\Omega$$

The boundary $\partial\Omega$ of the domain $\Omega$ can be decomposed into 3 sub-boundaries: a wall boundary $\partial\Omega_w$, a cut behind the body $\partial\Omega_c$ and a far field boundary $\partial\Omega_\infty$. While we can ignore what happens on the cut, we need to be careful with the other two boundaries. Given the equation of the drag, it is obvious that this quantity does not depend on the values of the flow in the far field ($\delta M_{|\partial\Omega_\infty} = 0$). Therefore, we can take

$$\psi = 0 \quad \text{on } \partial\Omega_\infty$$

On the wall, things are more complex. We know that both $w$ the exact solution and $w_{\mathbf{H}}$ the numerical solution on a mesh $H$ satisfy the wall boundary conditions $u_n = u_{n\mathbf{H}} = 0$. If follows

$$F(w) = p \begin{pmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ 0 \end{pmatrix} \quad \text{and} \quad F(w_{\mathbf{H}}) = p_{\mathbf{H}} \begin{pmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ 0 \end{pmatrix}$$

As a consequence,

$$\delta F = \delta p \begin{pmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ 0 \end{pmatrix}, \text{ with } \delta p = p_{\mathbf{H}} - p$$

If we recall the expression of the drag (7.1), we see that $\delta M = \kappa \delta p$, with $\kappa = n \cdot i_D$. Hence, we obtain the condition that needs to be satisfied by $\psi$ on the wall boundary

after cancelling $\delta p$

$$\psi^T \cdot \begin{pmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ 0 \end{pmatrix} = \kappa$$

$$\Leftrightarrow \qquad \psi_2 n_1 + \psi_3 n_2 + \psi_4 n_3 = \kappa$$

**Formulation of the problem**

Finally, we can formulate concisely the problem we need to solve:

$$\begin{cases} A_i^T \dfrac{\partial \psi}{\partial x_i} = 0, & \text{on } \Omega \\ \psi = 0, & \text{on } \partial\Omega_\infty \\ \psi_2 n_1 + \psi_3 n_2 + \psi_4 n_3 = \kappa, & \text{on } \partial\Omega_w \end{cases} \tag{7.3}$$

## 7.2 The SD Adjoint Solver

### 7.2.1 The transformed equation

The implementation of the SD method for the adjoint equation is very similar to that for the Euler equations in the previous chapter. A pseudo time stepping term is then added to Equation (7.3) giving the transformed equation in the standard element:

$$\frac{\partial \psi}{\partial t} - C_i \frac{\partial \psi}{\partial \xi_i} = 0 \tag{7.4}$$

where

$$C_i = [J^{-1}]_{ij} A_j^T \quad \text{and} \quad J = \frac{\partial \mathbf{x}}{\partial \xi}.$$

### 7.2.2 Implementation in 2D

In 2D, at each time step the residual ($\partial\psi/\partial t$) can be obtained through the following steps:

1. Given at the solution points $\psi_{i,j}$, reconstruct $\psi$ with

$$\psi(\xi_1, \xi_2) = \sum_{i=1}^{N}\sum_{j=1}^{N} \psi_{i,j} h_i(\xi_1) h_j(\xi_2)$$

   and compute at the flux points $\psi_{i+\frac{1}{2},j}$ and $\psi_{i,j+\frac{1}{2}}$;

2. Compute the values of $\psi$ at the interface and boundary flux points;

3. Compute at the solution points

$$\frac{\partial\psi}{\partial\xi_1}|_{i,j} = \sum_{i=0}^{N}\sum_{j=1}^{N} \psi_{i+\frac{1}{2},j} l'_{i+\frac{1}{2},j}(\xi_1) h_j(\xi_2)$$
$$\frac{\partial\psi}{\partial\xi_2}|_{i,j} = \sum_{i=1}^{N}\sum_{j=0}^{N} \psi_{i,j+\frac{1}{2}} h_j(\xi_1) l'_{i,j+\frac{1}{2}}(\xi_2)$$

4. Eventually, the residual at the solution points is

$$\frac{\partial\psi}{\partial t}|_{i,j} = C_{1_{i,j}} \frac{\partial\psi}{\partial\xi_1}|_{i,j} + C_{2_{i,j}} \frac{\partial\psi}{\partial\xi_2}|_{i,j}$$

### 7.2.3 Interface values of $\psi$

At the interface flux points, the $\psi$ values interpolated from the left and right cells are discontinuous ($\psi_L \neq \psi_R$). We can define a common interface value $\psi^{(i)}$ as follows:

$$\psi^{(i)} = \frac{1}{2}(\psi_L + \psi_R) - \frac{1}{2}\alpha \cdot \mathbf{sign}(C) \cdot (\psi_L - \psi_R), \quad \alpha \in [0, 1]$$

where $\alpha = 0$ corresponds to a central flux and $\alpha = 1$ corresponds to an upwind flux. $\mathbf{sign}(C)$ is given by

$$\mathbf{sign}(C) = X \cdot \mathbf{sign}(\Lambda) \cdot X^{-1}, \quad C = X\Lambda X^{-1}.$$

where $\Lambda$ is a diagonal matrix.

### 7.2.4 Boundary conditions in 2D

A set of boundary values of $\psi$ that would satisfy the boundary conditions in Equation (7.3) are shown below.
Far field:

$$\psi^- = 0$$

Airfoil wall:

$$\begin{aligned}
\psi_1^- &= -\psi_1^+ \\
\psi_2^- &= \psi_2^+ + 2n_1(\kappa - n_1\psi_2^+ - n_2\psi_3^+) \\
\psi_3^- &= \psi_3^+ + 2n_2(\kappa - n_1\psi_2^+ - n_2\psi_3^+) \\
\psi_4^- &= -\psi_4^+
\end{aligned}$$

where $n = [n_1, n_2]^T$ is the unit normal vector. $\psi^+$ refers to the values of $\psi$ at flux points extrapolated from the solution in the boundary cell, and $\psi^-$ refers to the fictitious values of $\psi$ on the other side of the boundary interface.

# Chapter 8

# Mortar Elements

For the spectral difference method described in Chapter 6, on the interface shared by two adjacent cells, the flux points from one cell coincide with the flux points from the other cell. The Riemann problem is solved between each pair of flux points to give a common flux. However, as mentioned in Chapter 2, on quadrilateral or hexahedral meshes $h$ or $p$ refinement results in interfaces where the flux points on both sides do not match up in pairs. A special treatment called the 'mortar method' chosen by Kopriva[25, 43] which enables information transfer between neighboring cells with nonconforming interfaces.

The basic idea is that the two cells on both sides of the interface do not communicate with each other directly. Instead they both communicate with an intermediate construct called a 'mortar element'. The idea is to make a mapping between the reconstructed polynomial on a face and the polynomial representation on the mortar element. The solution polynomials on element faces are mapped onto the mortar element, the Riemann problem is then solved between each pair of points on the mortar element to get a common flux polynomial, and lastly the common flux polynomial is mapped back to the element faces connected to the mortar. Unweighted $L^2$ projection is used for mapping of polynomials between a face and a mortar element. This chapter describes and derives the mortar method in 2D.

## 8.1 $p$-Adaptation

$p$-adaptation results in neighboring cells with different orders. The interface connecting two cells has different number of flux points on either side. A mortar is chosen between the cell faces. To find the mapping between a cell face and a mortar, we consider a general case where the cell face and the mortar contain unequal numbers of flux points ($N_f$ and $N_m$ respectively), but have the same length (Figure 8.1.3). The polynomial approximation along the cell face and the mortar are

$$\begin{aligned} U^{(f)}(\xi) &= \sum_{j=1}^{N_f} U_j^{(f)} h_j^{(f)}(\xi) \quad \in \mathcal{P}_{N_f-1} \\ U^{(m)}(\xi) &= \sum_{j=1}^{N_m} U_j^{(m)} h_j^{(m)}(\xi) \quad \in \mathcal{P}_{N_m-1} \end{aligned} \tag{8.1}$$

with the superscripts $f$ and $m$ referring to the cell face and the mortar respectively.

### 8.1.1 Face to mortar

For mapping from the cell face to mortar, we seek the polynomial representation on the mortar that best approximates the polynomial on the cell face. I.e. given the vector $\mathbf{U}^{(f)} = (U_1^{(f)}, ..., U_{N_f}^{(f)})^T$, we need to find $\mathbf{U}^{(m)} = (U_1^{(m)}, ..., U_{N_m}^{(m)})^T$ to minimize

$$\int_0^1 \left(U^{(f)}(\xi) - U^{(m)}(\xi)\right)^2 d\xi$$

Taking derivative with respect to $\mathbf{U}^{(m)}$, the conditions to be satisfied are

$$\int_0^1 \left(U^{(f)}(\xi) - U^{(m)}(\xi)\right) h_j^{(m)}(\xi) d\xi = 0, \quad j = 1, ..., N_m \tag{8.2}$$

Substituting (8.1) into (8.2) to give:

$$M\mathbf{U}^{(m)} = S\mathbf{U}^{(f)} \Rightarrow \mathbf{U}^{(m)} = (M^{-1}S)\mathbf{U}^{(f)} \tag{8.3}$$

where $M$ and $S$ are $N_m \times N_m$ and $N_m \times N_f$ matrices defined as

$$\begin{aligned} M_{ij} &= \int_0^1 h_i^{(m)} h_j^{(m)} d\xi \\ S_{ij} &= \int_0^1 h_i^{(m)} h_j^{(f)} d\xi \end{aligned} \tag{8.4}$$

### 8.1.2 Mortar to face

Mapping from the mortar back to the cell face has the same derivation, with the role of the cell face and the mortar switched, i.e.

$$\mathbf{U}^{(f)} = (M^{-1}S)\mathbf{U}^{(m)} \tag{8.5}$$

where

$$\begin{aligned} M_{ij} &= \int_0^1 h_i^{(f)} h_j^{(f)} d\xi \\ S_{ij} &= \int_0^1 h_i^{(f)} h_j^{(m)} d\xi \end{aligned} \tag{8.6}$$

### 8.1.3 Mortar procedure

The complete algorithm is summarized below (Figure 8.1.3):

Figure 8.1: Mortar configuration for $p$-refinement.

1. Choose a mortar of the same order as the face with higher order. Here we assume the left face has order $N_{f_L}$, and the right face has order $N_{f_R} > N_{f_L}$. The mortar is chosen to have the higher order $N_m = N_{f_R}$.

2. Project the conserved variables from the left face onto the mortar (Equation (8.3)). On the other side, the conserved variables are just copied from the right face to the mortar.

3. Solve the Riemann problem between each pair of flux points to compute the common interface fluxes across the mortar.

4. Project the computed fluxes back to the left face (Equation (8.5)). On the right side, the fluxes are copied from the mortar to the right face.

## 8.2 $h$-Adaptation

$h$-refinement causes interfaces with 'hanging nodes' as discussed previously. Consider a general case where on the left side of an interface is one single cell and on the right side are K smaller cells, as shown in Figure 8.2.3. We choose K mortars of the same sizes and at the same locations as the cells on the right, so that the mappings between each mortar and the corresponding right cell faces are exactly the same as what has been described for $p$-refinement in the previous section. We need to find the mapping between the left cell face and each mortar.

Let $N_f$ denote the number of flux points on the left cell face, and $N_{m_k}$ the number of flux points on the $k$th mortar ($k = 1, ..., K$). Let $o^k$ and $s^k$ denote the offset and the scale of the $k$th mortar with respect to the left face. The polynomials on the cell face and on the $k$th mortar face are

$$\begin{aligned} U^{(f)}(\xi) &= \sum_{j=1}^{N_f} U_j^{(f)} h_j^{(f)}(\xi) \quad \in \mathcal{P}_{N_f-1} \\ U^{(m_k)}(z) &= \sum_{j=1}^{N_{m_k}} U_j^{(m_k)} h_j^{(m_k)}(z) \quad \in \mathcal{P}_{N_{m_k}-1} \end{aligned} \tag{8.7}$$

### 8.2.1 Face to mortar

Given $\mathbf{U}^{(f)} = (U_1^{(f)}, ..., U_{N_f}^{(f)})^T$, we need to find $\mathbf{U}^{(m_k)} = (U_1^{(m_k)}, ..., U_{N_{m_k}}^{(m_k)})^T$ such that the polynomial on the mortar face best approximates the polynomial on the cell face

for $\xi \in [o^k, o^k + s^k]$, corresponding to $z \in [0, 1]$ on the mortar face, where $\xi = o^k + s^k z$. That is, $\mathbf{U}^{(m_k)}$ minimizes

$$\int_0^1 \left(U^{(f)}(z) - U^{(m_k)}(o^k + s^k z)\right)^2 dz$$

Taking derivative with respect to $\mathbf{U}^{(m_k)}$, the conditions to be satisfied are

$$\int_0^1 \left(U^{(f)}(z) - U^{(m_i)}(o^k + s^k z)\right) h_j^{(m_k)}(z) dz = 0, \quad j = 1, ..., N_m \tag{8.8}$$

Similar to before, substitution and rearranging the terms gives

$$\mathbf{U}^{(m_k)} = (M^{-1}S)\mathbf{U}^{(f)} \tag{8.9}$$

where $M$ and $S$ are $N_{m_k} \times N_{m_k}$ and $N_{m_k} \times N_f$ matrices defined as

$$\begin{aligned} M_{ij} &= \int_0^1 h_i^{(m_k)}(z) h_j^{(m_k)}(z) dz \\ S_{ij} &= \int_0^1 h_i^{(m_k)}(z) h_j^{(f)}(o^k + s^k z) dz \end{aligned}$$

### 8.2.2 Mortar to face

When the polynomials on the $K$ mortars are mapped back to the left cell face, we want the face polynomial on $[o^k, o^k + s^k]$ to best approximate the polynomial on the $k$th mortar. Therefore we need to find $\mathbf{U}^{(f)}$ such that

$$\sum_{k=1}^{K} \int_{o^k}^{o^k + s^k} \left(U^{(f)}(\xi) - U^{(m_k)}(\xi)\right)^2 d\xi$$

is minimized. Taking derivatives and rearranging gives

$$\mathbf{U}^{(f)} = \sum_{k=1}^{K} (M^{-1}S^{(k)})\mathbf{U}^{(m_k)} \tag{8.10}$$

where

$$
\begin{aligned}
M_{ij} &= \int_0^1 h_i^{(f)}(xi) h_j^{(f)}(\xi) d\xi \\
S_{ij}^{(k)} &= s^k \int_0^1 h_i^{(f)}(o^k + s^k z) h_j^{(m_k)}(z) dz
\end{aligned}
$$

### 8.2.3 Mortar procedure

For an interface with a parent face on one side and $K$ child faces on the other, the complete algorithm is summarized below (Figure 8.1.3):

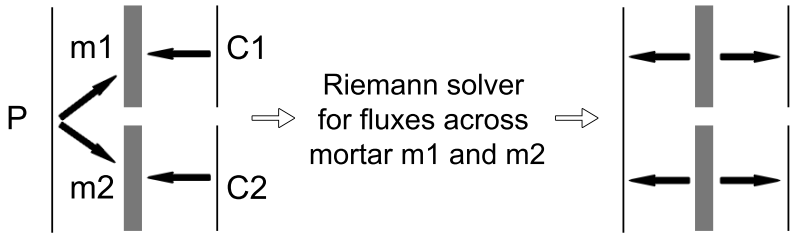



Figure 8.2: Mortar configuration for $h$-refinement. In this figure it is assumed $K = 2$.

1. Choose $K$ mortar faces of the same length and location as the child faces. The order of the $k$th mortar face is chosen to be the maximum or the order of the parent face and the order of the $k$th child face.

2. Project the conserved variables from the parent face onto the mortars (Equation (8.9)). On the other side, the conserved variables are just copied from the child faces to their corresponding mortars.

3. Solve the Riemann problem between each pair of flux points to compute the common interface fluxes across each mortar.

4. Project the computed fluxes back to the parent face (Equation (8.10)). On the other side, the fluxes are copied from each mortar to the corresponding child face.

# Chapter 9

# Adaptation Procedure

## 9.1 Error Indicator for an Element

Equation (4.7) gives an *a posteriori* error estimation for the output $J$:

$$\delta J = \langle \psi_{\mathbf{H}}, \mathbf{R}(\mathbf{u_H}) \rangle \tag{9.1}$$

However, when it is evaluated, the continuous residual $\mathbf{R}(\mathbf{u_H})$ is not readily available. We can approximate it by the discrete residual on a finer mesh $h$, $\mathbf{R_h}(\mathbf{u_H^h})$, where $\mathbf{u_H^h}$ is obtained by extrapolating $\mathbf{u_H}$ onto mesh $h$. We do the same for $\psi_{\mathbf{H}}$, and the error can be evaluated as

$$\delta J = \left(\psi_{\mathbf{H}}^{\mathbf{h}}, \mathbf{R_h}(\mathbf{u_H^h})\right) \tag{9.2}$$

For each element $\Omega_k$, the error indicator is defined as

$$\eta_k = \left(\psi_{\mathbf{H}}^{\mathbf{h}}, \mathbf{R_h}(\mathbf{u_H^h})\right)_{\Omega_k} \tag{9.3}$$

The finer mesh $h$ traditionally refers to a mesh with smaller mesh widths than the current mesh $H$. However this approach can be computationally expensive as it requires a lot more memory to be allocated. Another approach, as used by Fidkowski *et al.* in [12], is to obtain $h$ by increasing the interpolation order from $N$ to $N+1$ on the same mesh $H$.

Note that in estimating the error, two approximations are made. The first one occurs when linearizing $J$ and $\mathbf{R}$ (Equation (4.4), (4.5)). The second is done when the discrete residual $\mathbf{R_h}(\mathbf{u_H^h})$ is used instead of the continuous residual $\mathbf{R}(\mathbf{u_H})$, as the latter is not available.

To evaluate the inner product in Equation (9.3) over an element, the solution polynomials in Lagrange basis for the spectral difference method are converted to the orthogonal Legendre basis first.

## 9.2 $h$- and $p$-adaptation

The adaptation procedure is outlined as follows:

1. Solve the flow equations on the current mesh $H$ and with current order $N$ to obtain $\mathbf{u_H}$;
2. Solve the adjoint equations on the current mesh $H$ and with current order $N$ to obtain $\psi_\mathbf{H}$. For the entropy adjoint approach, calculate from $\mathbf{u_H}$ directly the entropy variables;
3. Project $\mathbf{u_H}$ and $\psi_\mathbf{H}$ into order $N+1$ space to obtain $\mathbf{u_H^h}$ and $\psi_\mathbf{H}^\mathbf{h}$;
4. Calculate the error indicator for each element $\eta_k$;
5. Refine/coarsen a fraction of the elements with the largest/smallest $|\eta_k|$;
6. Inject the solution to the adapted mesh;
7. Repeat from step 1 until the desired accuracy is reached.

## 9.3 $hp$-adaptation

For $hp$-adaptation, the idea is to $h$-refine the elements with large errors and flow discontinuities, and $p$-refine those with large errors and smooth flows. While the error

indicator is still the adjoint-weighted residuals, the discontinuity sensor introduced in Equation (3.3) is used to measure smoothness of the flow within each element:

$$S_k = \frac{(u - \bar{u}, u - \bar{u})_{\Omega_k}}{(u, u)_{\Omega_k}}$$

The adaptation procedure is outlined as follows:

1. Solve the flow equations and the adjoint equations on the current mesh;
2. Calculate the error indicator $\eta_k$ for each element using the same method as in the previous section;
3. Calculate the discontinuity sensor $S_k$ for each element using Equation (3.3);
4. Tag a fraction of the elements with the largest estimated error $|\eta_k|$;
5. For each of these elements, $h$-refine it if $S_k > S_{threshold}$ and $p$-refine it otherwise;
6. Inject the solution to the adapted mesh;
7. Repeat from step 1 until the desired accuracy is reached.

# Part III

# Test Cases

# Chapter 10

# Test Cases

The proposed adjoint-based adaptation methodology has been tested on 2D Euler equations. Three groups of tests are conducted. In the first group, the entropy adjoint approach is applied to a half-cylinder in subsonic flow, and comparisons between the results from $h$- and $p$-adaptations are made. The second group consists of tests on a NACA 0012 airfoil in subsonic flow. The entropy adjoint and the drag adjoint indicators are compared. Both $h$- and $p$-refinement are tested. In the third group, $h$-refinement is done on a NACA 0012 airfoil in transonic flow, where shocks are present. This examines the ability of handling flow discontinuities by the continuous adjoint method. Also, some simple experiments on $hp$-adaptation are conducted.

## 10.1 Subsonic Flow Past a Half-Cylinder with Entropy Adjoint Approach

The first test case is a 2-D cylinder in subsonic flow at $M = 0.3$. The cylinder is calculated by ensuring symmetry about the $x$-axis. This is done by modeling only the upper half of the cylinder and enforcing symmetric boundary conditions at the boundary along $x$-axis. For visualization purposes, all the plots are done on the full cylinder. The initial mesh, as shown in Figure 10.1, has 512 cells for a half-cylinder. The computational domain extends to about 200 times the chord length in order to

minimize the effects of waves reflected from far field boundaries. The Mach contour on the initial mesh indicates smooth subsonic flow everywhere in the domain.

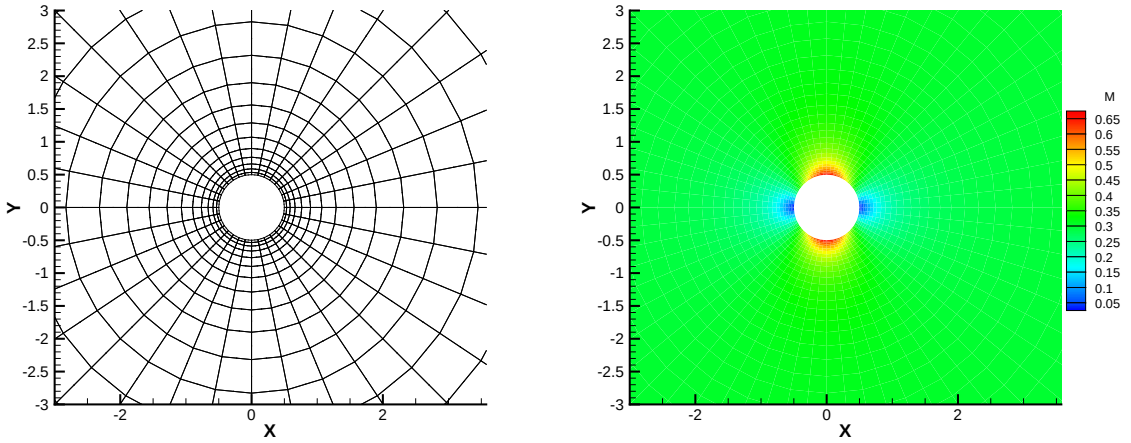



Figure 10.1: Initial mesh and Mach contour for the cylinder test case, $M = 0.3$.

The objective of this test case is to study the performance of the entropy adjoint approach, and to compare $h$- and $p$-adaptation. Three groups of tests are conducted: $h$-refinement, $p$-refinement, and $p$-refinement/coarsening. Only the entropy adjoint is used for error estimation.

### 10.1.1 $h$-refinement

In this sub-section, $h$-refinement is conducted with the order of accuracy $N$ being the same in every element within the flow domain. Three levels of $h$-refinement are done for solutions with $N = 3$ and $N = 4$ respectively. 10% of total number of elements are refined at each level of refinement. Figure 10.2 shows the results for $N = 3$, in particular the estimated error (in logarithmic scale) before each level of refinement and the adapted mesh after each level of refinement. A few observations can be made. Firstly, the estimated errors on the initial mesh are larger in cells that are closer to the surface of the cylinder. Secondly, these cells with larger errors are refined, and the estimated errors in these cells after refinement are noticeably reduced. Thirdly,

repeated adaptations of such kind drives towards equi-distribution of the estimated error, i.e. similar values in all cell elements.

Figure 10.3 shows the corresponding results for $N = 4$. The errors are plotted on the same scale as for $N = 3$. It can be seen that the estimated error of the solution on the initial mesh with $N = 4$ is everywhere smaller than that with $N = 3$. The adapted meshes are also slightly different: with $N = 4$, the refined cells are more concentrated close to the surface of the cylinder.

### 10.1.2 *p*-refinement

In this sub-section, *p*-refinement is performed while no change is made to the physical mesh. Figure 10.4 shows the results for three levels of adaptation, refining 10% of the cells at each level, starting with $N = 3$ in all cells. The areas that are adapted show a similar pattern as *h*-refinement. Figure 10.5 shows the results for *p*-refinement starting with $N = 4$. Both the error distribution and order distribution are plotted on the same scale as in Figure 10.4. Again it can be seen that the refinement is more concentrated around the cylinder in the case $N = 4$. This phenomenon is shown in both *h*- and *p*-refinement.

### 10.1.3 *p*-refinement/coarsening

In the above sub-sections, only refinement is looked at. In practice, *h*-coarsening is often not necessary as the starting mesh is usually very coarse. Once a cell has been refined, it is preferred not to be coarsened again because the better accuracy achieved on the fine cells would have been lost. It is similar for *p*-coarsening if the starting order is low. Intuitively, *p*-coarsening is only needed when the starting order in the whole domain is high. In the aspect of implementation, *p*-coarsening can be easily done using the same framework for *p*-refinement. In contrast, *h*-coarsening would introduce additional complexity.

In this sub-section, a combination of *p*-refinement and coarsening is used. Three levels of adaptation are done, with 20% of the cells being refined and 10% being coarsened at each level. Again, two sets of experiments are done, starting with $N = 3$

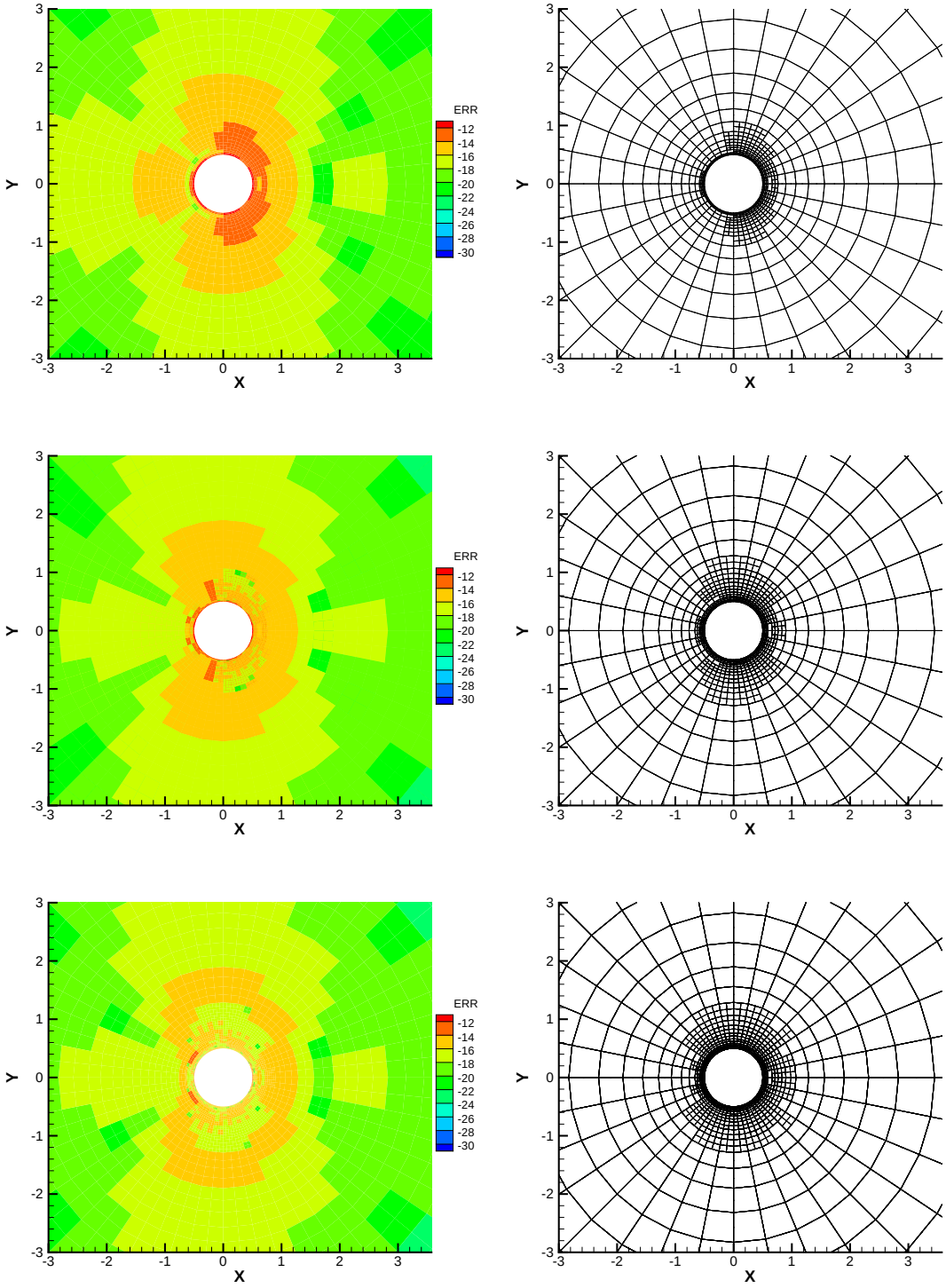



Figure 10.2: Left: error distribution before the 1st, 2nd and 3rd level of $h$-refinement. Right: the mesh after the 1st, 2nd and 3rd level of $h$refinement. $N = 3$ in the whole domain.

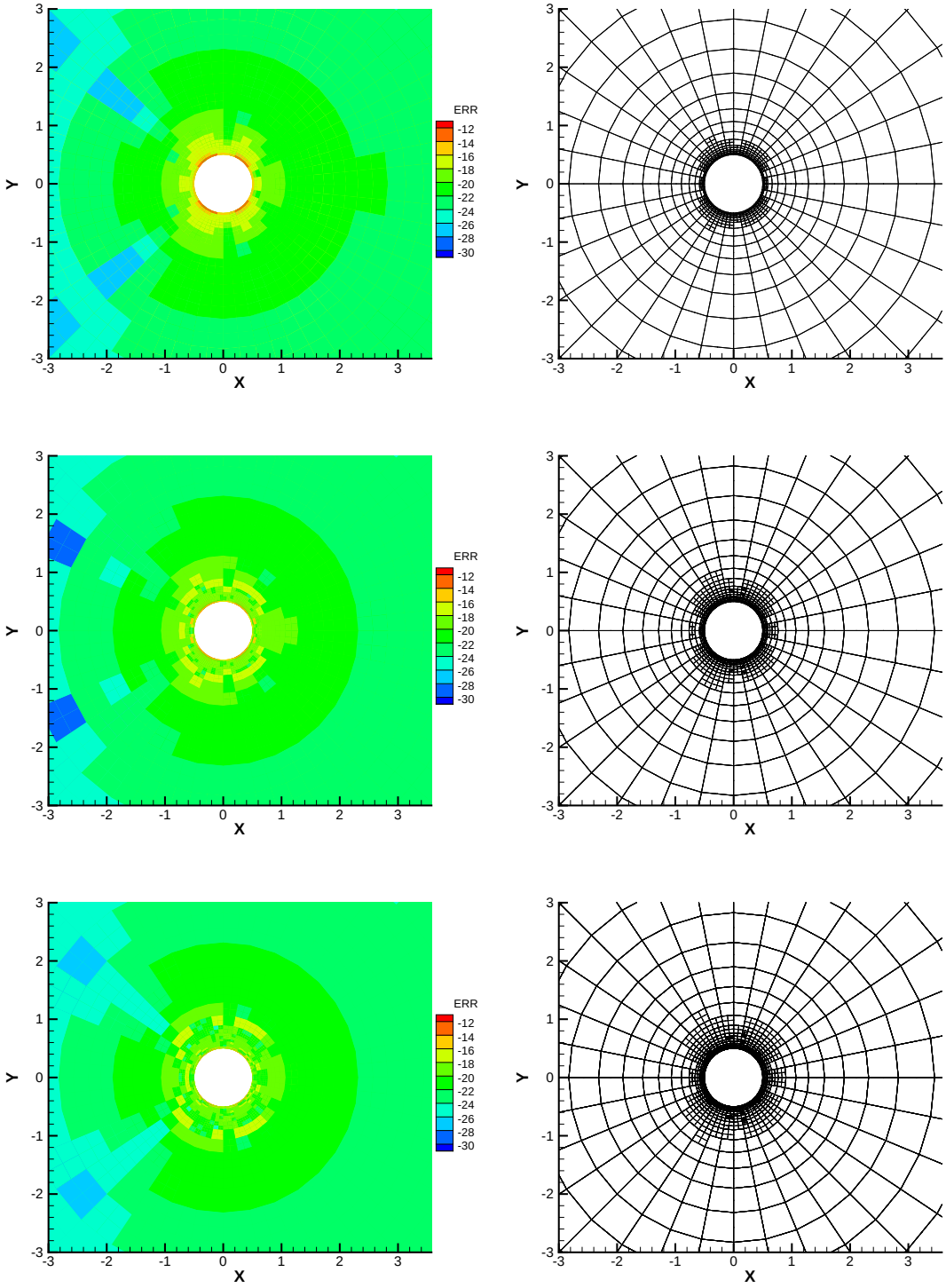



Figure 10.3: Left: error distribution before the 1st, 2nd and 3rd level of $h$-refinement. Right: the mesh after the 1st, 2nd and 3rd level of $h$refinement. $N = 4$ in the whole domain.

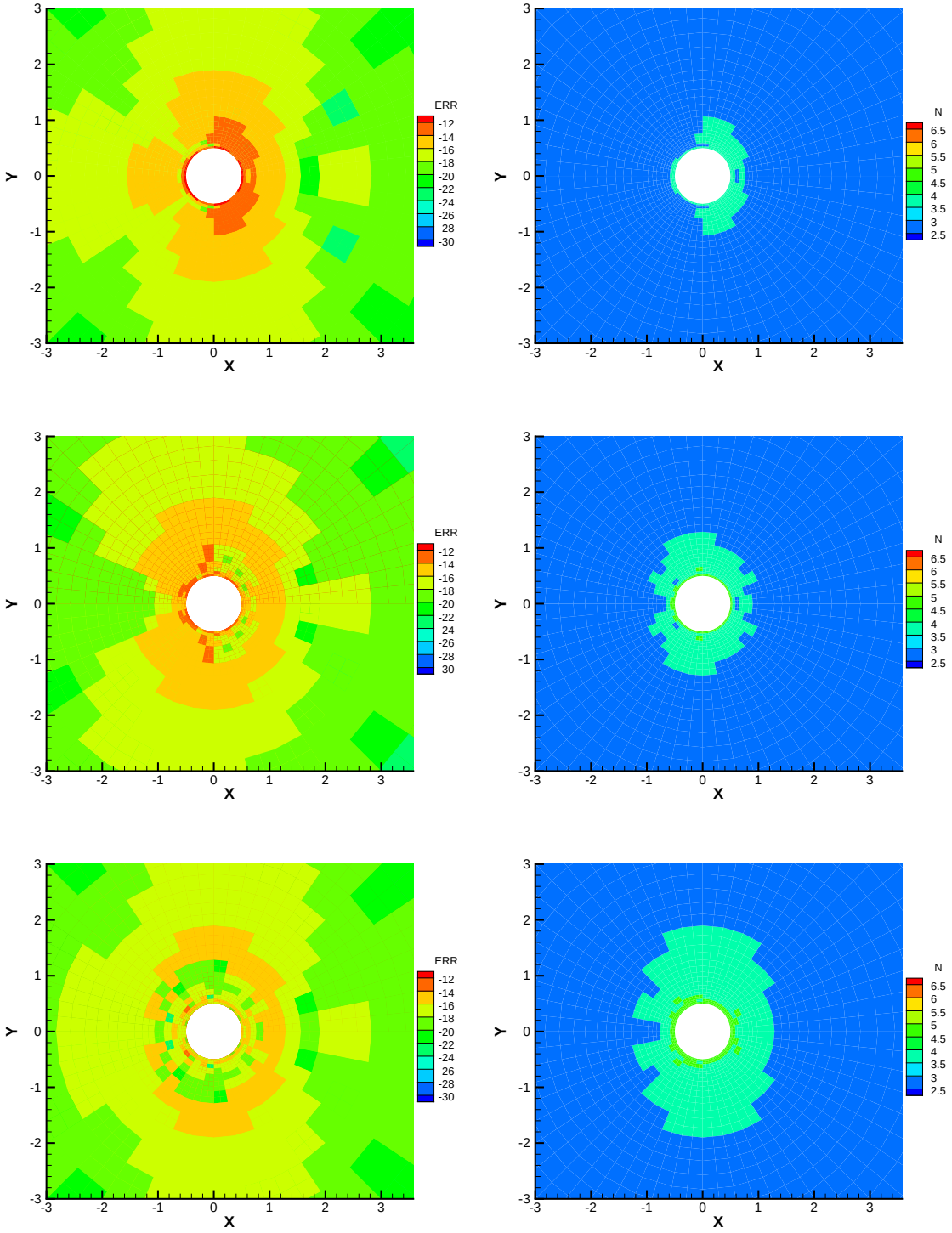



Figure 10.4: Left: error distribution before the 1st, 2nd and 3rd level of $p$-refinement. Right: the order distribution after the 1st, 2nd and 3rd level of $p$-refinement. Initially $N = 3$ in the whole domain.

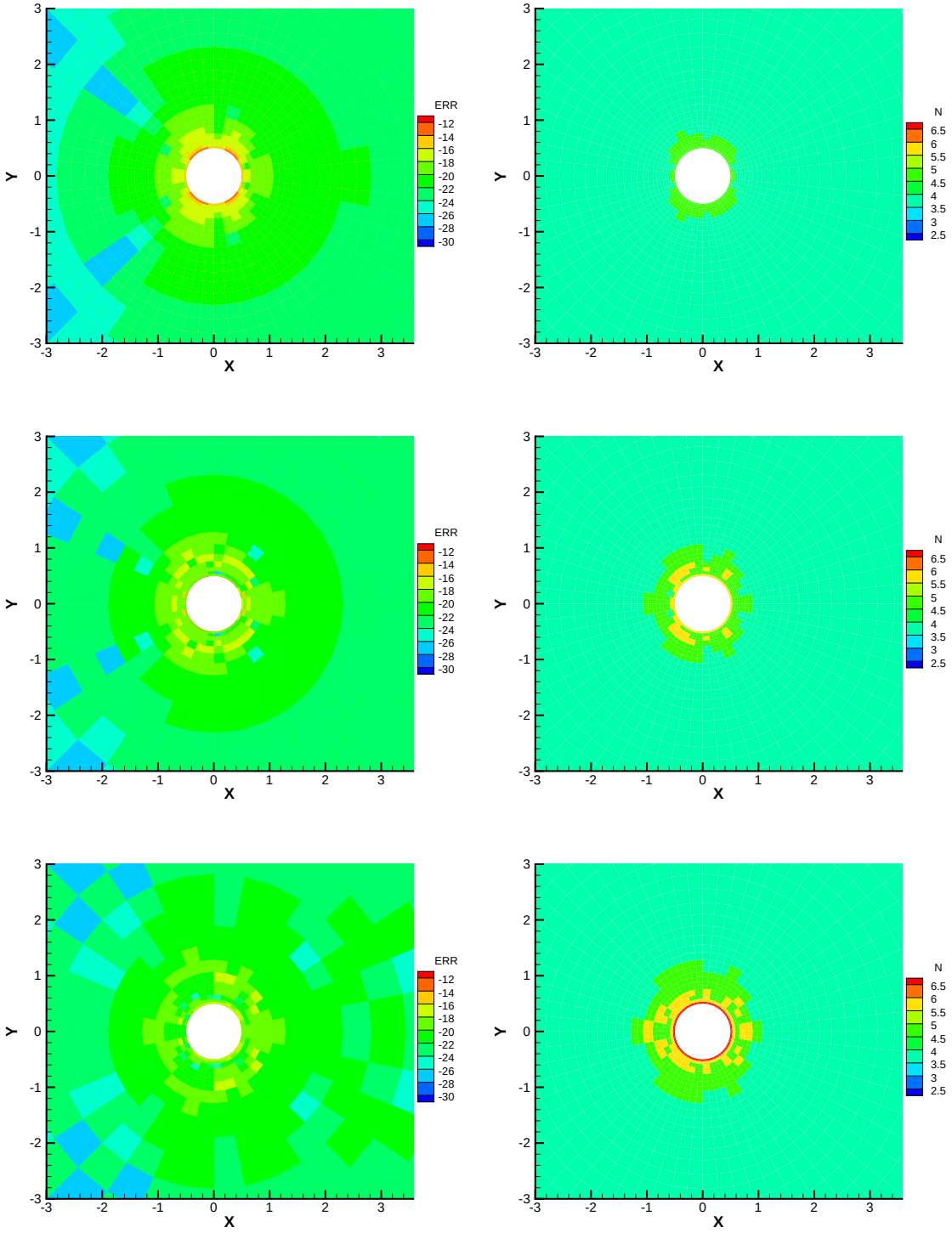



Figure 10.5: Left: error distribution before the 1st, 2nd and 3rd level of $p$-refinement. Right: the order distribution after the 1st, 2nd and 3rd level of $p$-refinement. Initially $N = 4$ in the whole domain.

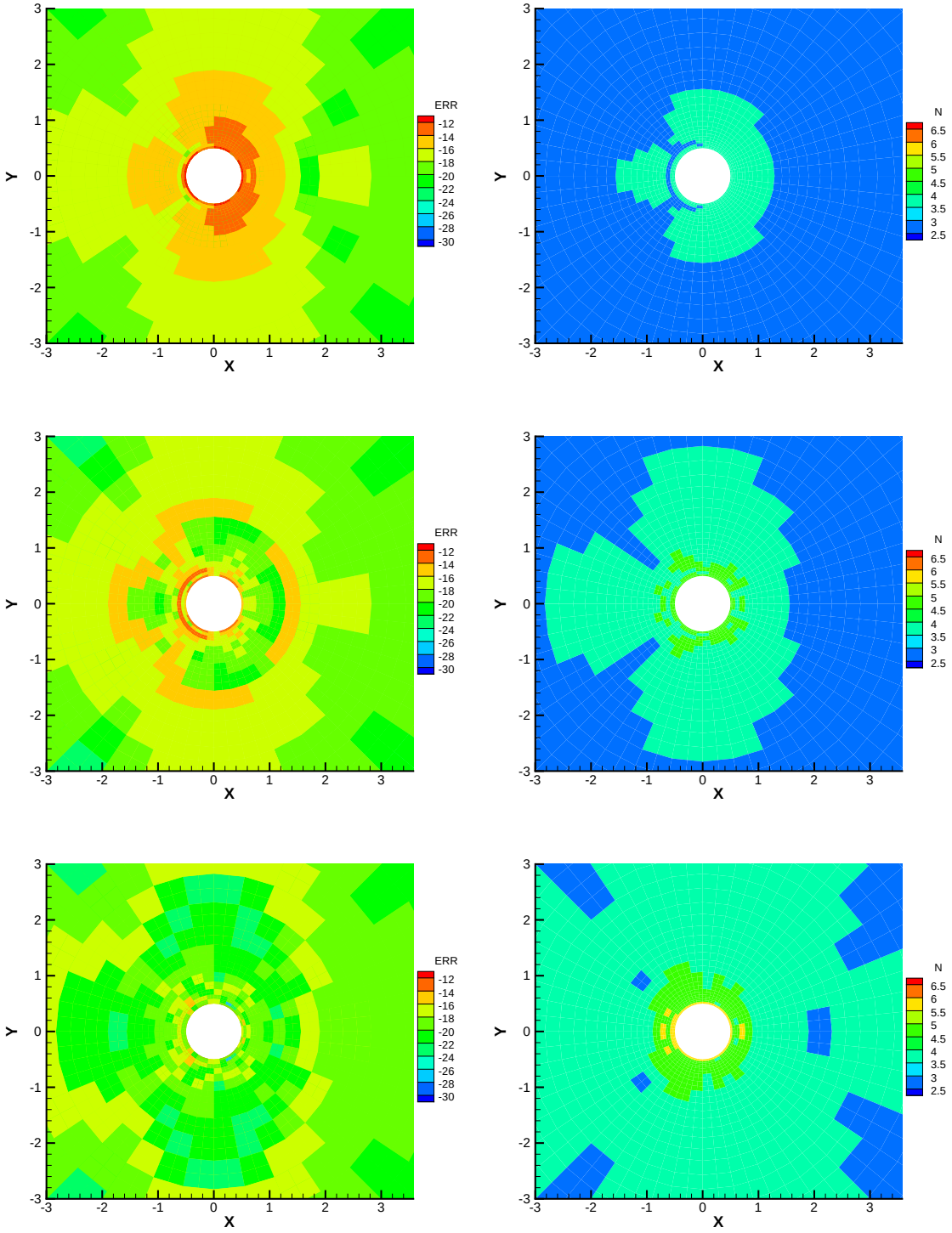



Figure 10.6: Left: error distribution before the 1st, 2nd and 3rd level of $p$-refinement/coarsening. Right: the order distribution after the 1st, 2nd and 3rd level of $p$-refinement/coarsening. Initially $N = 3$ in the whole domain.

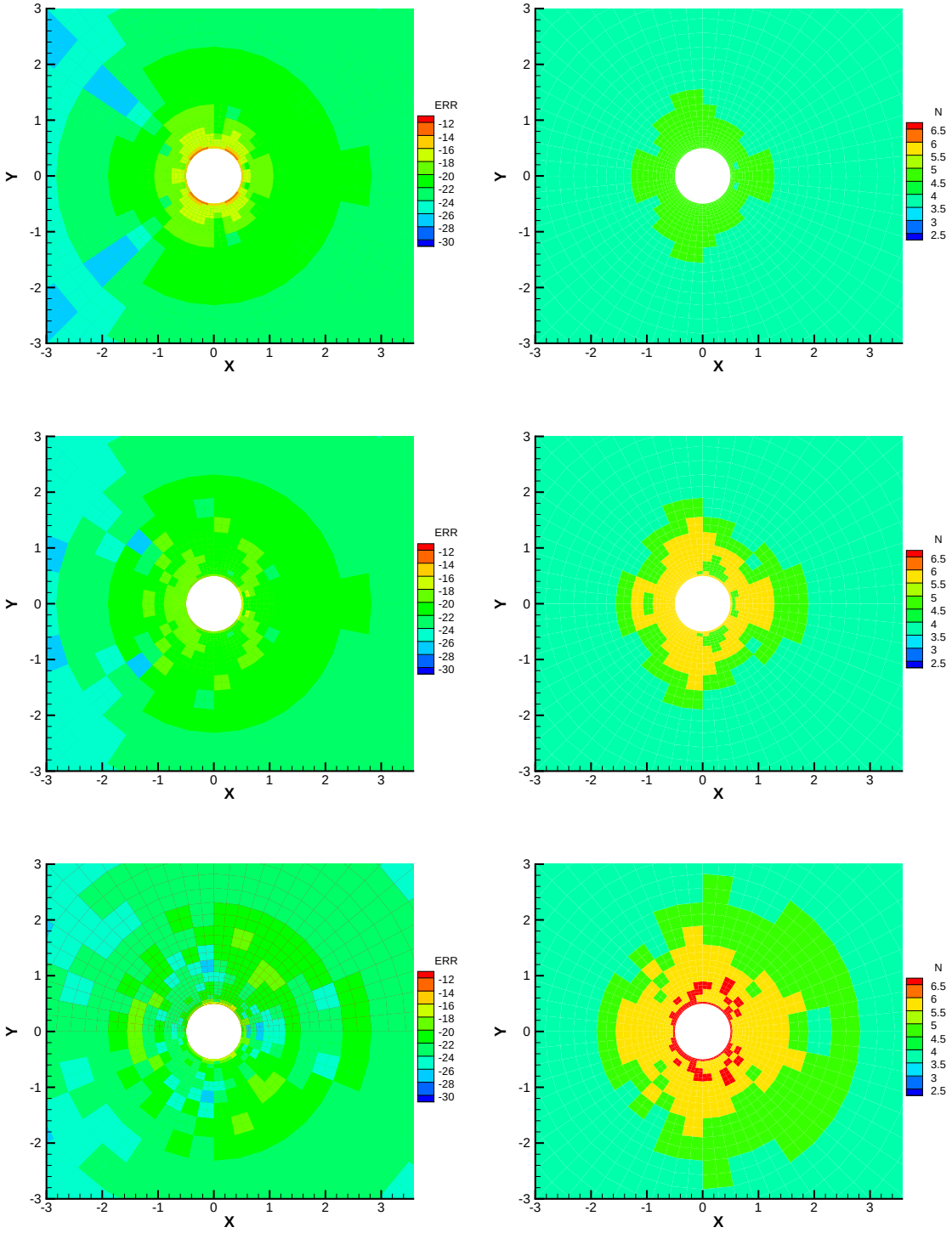



Figure 10.7: Left: error distribution before the 1st, 2nd and 3rd level of $p$-refinement/coarsening. Right: the order distribution after the 1st, 2nd and 3rd level of $p$-refinement/coarsening. Initially $N = 4$ in the whole domain.

and $N = 4$ respectively, and the results are shown in Figure 10.6 and 10.7. It appears that the order distribution is more 'fragmented' than that in Figure 10.4 and 10.5. This implies more mortar elements needed and therefore a higher computational cost, which may not be desirable in practice.

### 10.1.4 Comparison of $h$- and $p$-refinement

For a cylinder in a subsonic inviscid flow, the theoretical drag value is zero. Therefore the true error in drag is known for a numerical solution. The error in drag coefficient against the number of DOFs for various adaptation strategies is plotted in Figure 10.8. Uniform $p$-refinement consists of solutions on the initial mesh with $N = 2, 3, 4$ and 5. Uniform $h$-refinement is done with $N = 4$ globally, starting with the initial mesh. At each level of refinement every cell is $h$-refined. The results for adaptive $h$-refinement, $p$-refinement and $p$-refinement/coarsening, starting with $N = 3$ and $N = 4$, are also plotted.

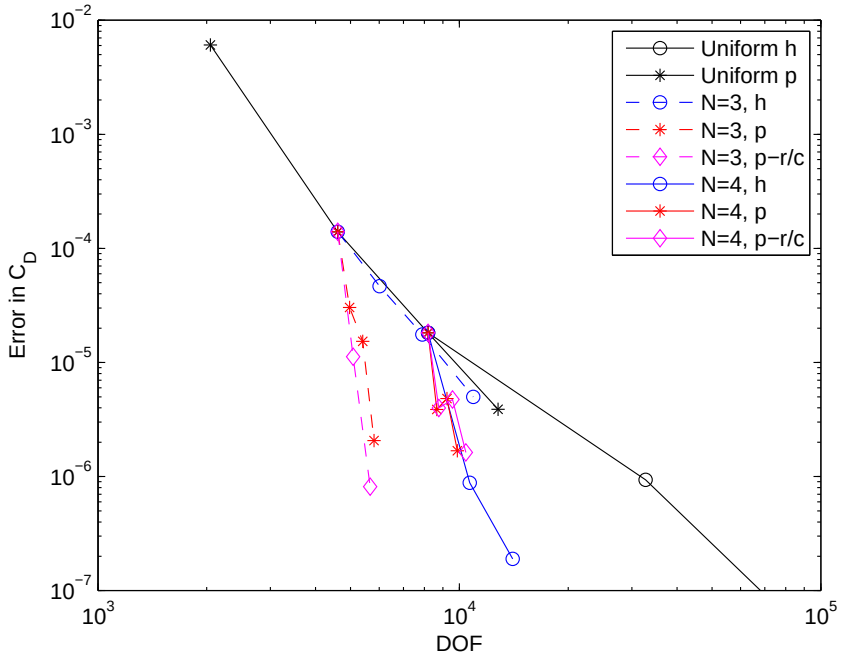



Figure 10.8: Plot of error in drag vs number of degrees of freedom (DOFs) for the cylinder test case.

A few observations can be made. Firstly, all adapted meshes provide better accuracy than uniformly refined meshes for a given number of DOF. This shows that the entropy adjoint error estimation correctly identifies the cells that are the main contributors to the total error. Although the target output is the net entropy flow out of the domain, the error in drag is also greatly reduced on the adapted meshes. This implies that the cells causing large errors in net entropy flow are also causing large errors in drag. This claim will be further proved and discussed in the next section.

When the starting order is $N = 3$, then $p$-adaptations have a much steeper error convergence rate than $h$-adaptation, which is only slightly better than uniform refinement. This agrees with the claim that for a given number of DOFs, higher order schemes offer better accuracy than lower order schemes. With $N = 3$, splitting into a finer mesh is not as effective as raising the order of approximation polynomials on the mesh.

However, at $N = 4$, $h$-adaptation does show competitive results as $p$-adaptations. This suggests that $N = 4$ provides sufficient accuracy in a way that raising the order no longer gives more advantages than $h$-refining the cells. It also suggests that it is best to do $h$-refinement with a high order scheme.

## 10.2 Subsonic Flow Past an Airfoil

Having compared $h$- and $p$-adaptations, the tests in this section aim to assess the performance of error estimation using entropy adjoint and drag adjoint for a NACA 0012 airfoil in subsonic flow of $M = 0.4$ at an angle of attack $\alpha = 5°$. The initial mesh and the Mach contour on the initial mesh are shown in Figure 10.9. The mesh is a C-mesh with 512 elements. The computational domain extends to about 100 times the chord length. The Mach contour shows a smooth flow without flow discontinuities.

### 10.2.1 $h$-Refinement

$h$-refinement is tested using the entropy and the drag adjoint error estimation respectively. $N = 4$ globally. Three levels of refinement is done in each case, with 10% of

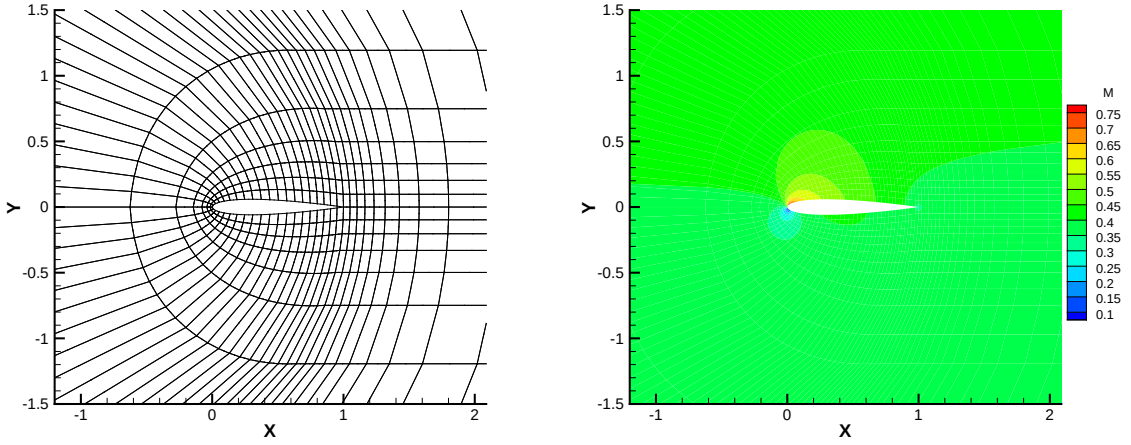



Figure 10.9: Initial mesh and Mach contour for the NACA 0012 test case, $M = 0.4$, $\alpha = 5°$.

the cells being refined at each level. Figure 10.10 and 10.11 show the results for the entropy and the drag adjoint respectively. The estimated error on the initial mesh using the entropy adjoint (first plot in Figure 10.10) indicates that large error occurs at the leading edge, trailing edge and along the stagnation streamline off the leading edge. The corresponding plot with the drag adjoint (first plot in Figure 10.11) shows a similar pattern, except that along the wall surfaces near the leading and trailing edges the error is also relatively large. Note that these two graphs are not plotted using the same color scale, because they are estimating local errors contributing to different functional outputs. The final adapted meshes in these two figures reflect the estimated error distributions.

It is worth mentioning that in others' attempts on the same case using the discrete adjoint approach, the adapted meshes show similar patterns. Figure 10.12 show the adapted meshes obtained by Fidkowski and Roe using the discrete adjoint approach with DG scheme, $N = 3$ globally. Several target output functionals are tested. Eight levels of anisotropic $h$-refinement are done in each case. The resulted meshes are more elaborate because of more levels of adaptation, but the refined areas are also focused around the trailing and leading edges, as well as the wall surfaces and the stagnation lines.

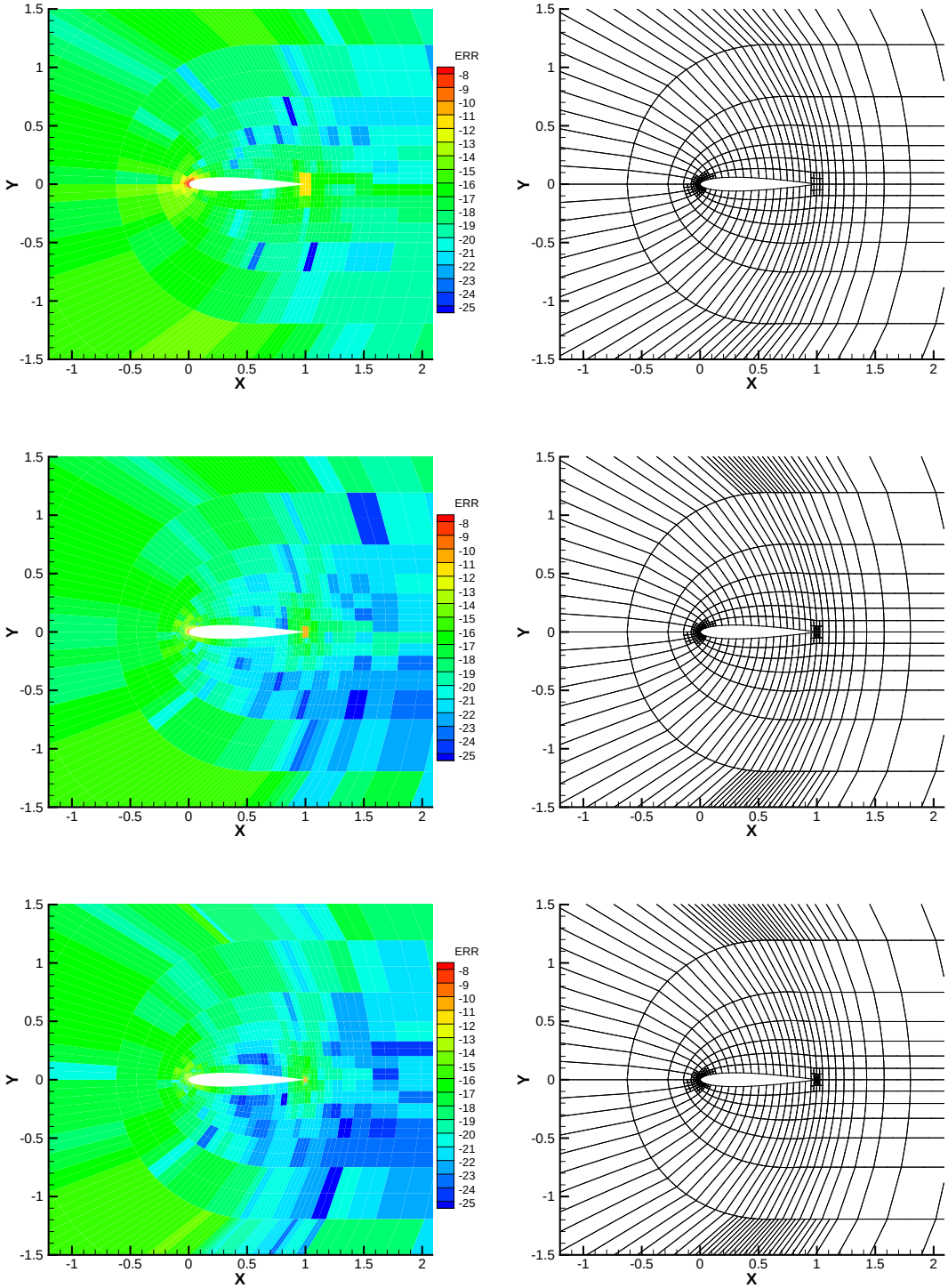



Figure 10.10: Left: error distribution before the 1st, 2nd and 3rd level of $h$-refinement using the entropy adjoint. Right: the mesh after the 1st, 2nd and 3rd level of $h$-refinement.

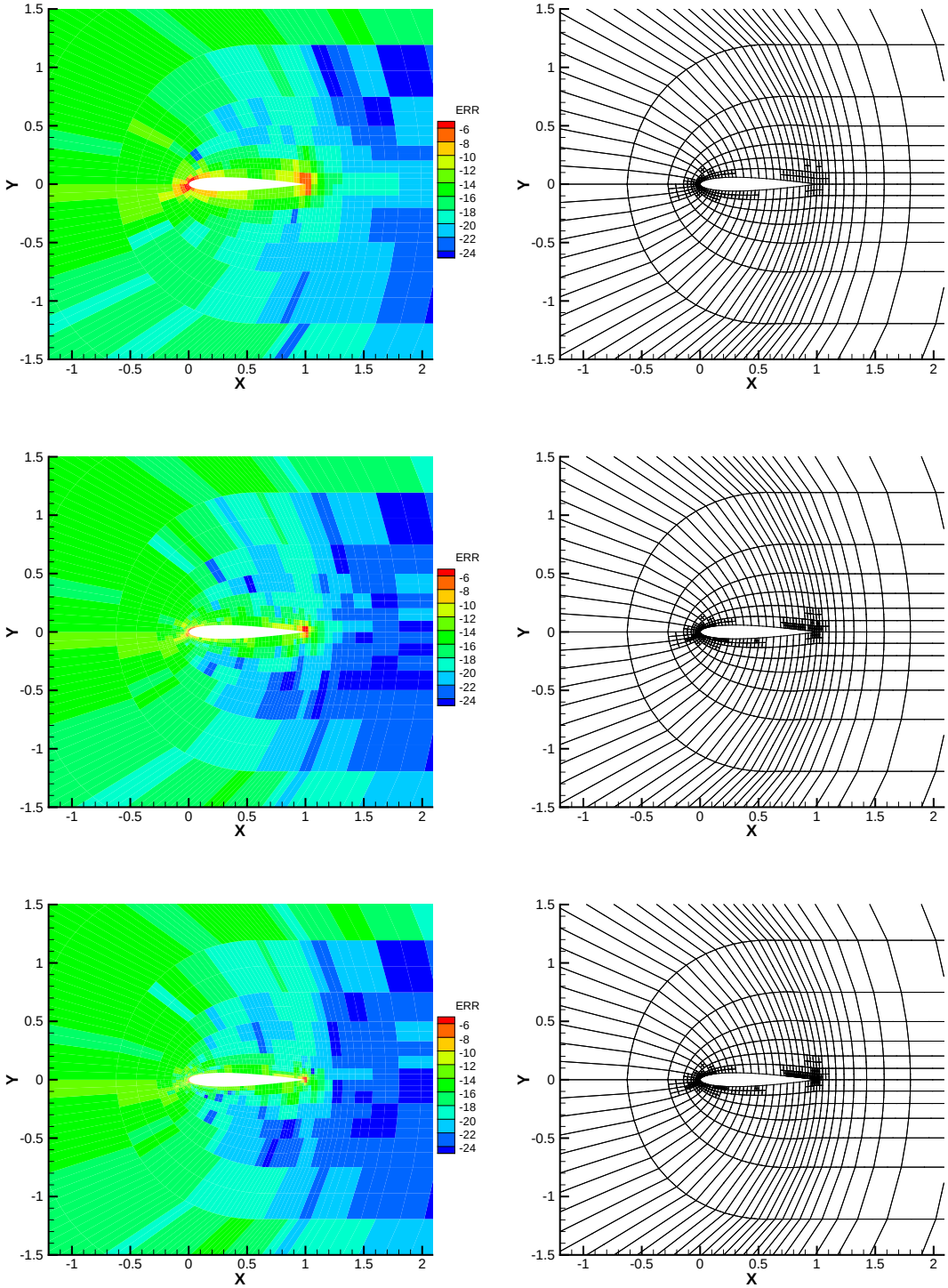



Figure 10.11: Left: error distribution before the 1st, 2nd and 3rd level of $h$-refinement using the drag adjoint. Right: the mesh after the 1st, 2nd and 3rd level of $h$-refinement.

It seems to suggest that firstly, although the estimated error distributions can vary slightly by the target output used, they share more similarities than discrepancies. A cell which has large errors for one output tends to have large error for the other outputs, too. Secondly, continuous and discrete adjoint methods give similar results, as the mathematical derivations have predicted.

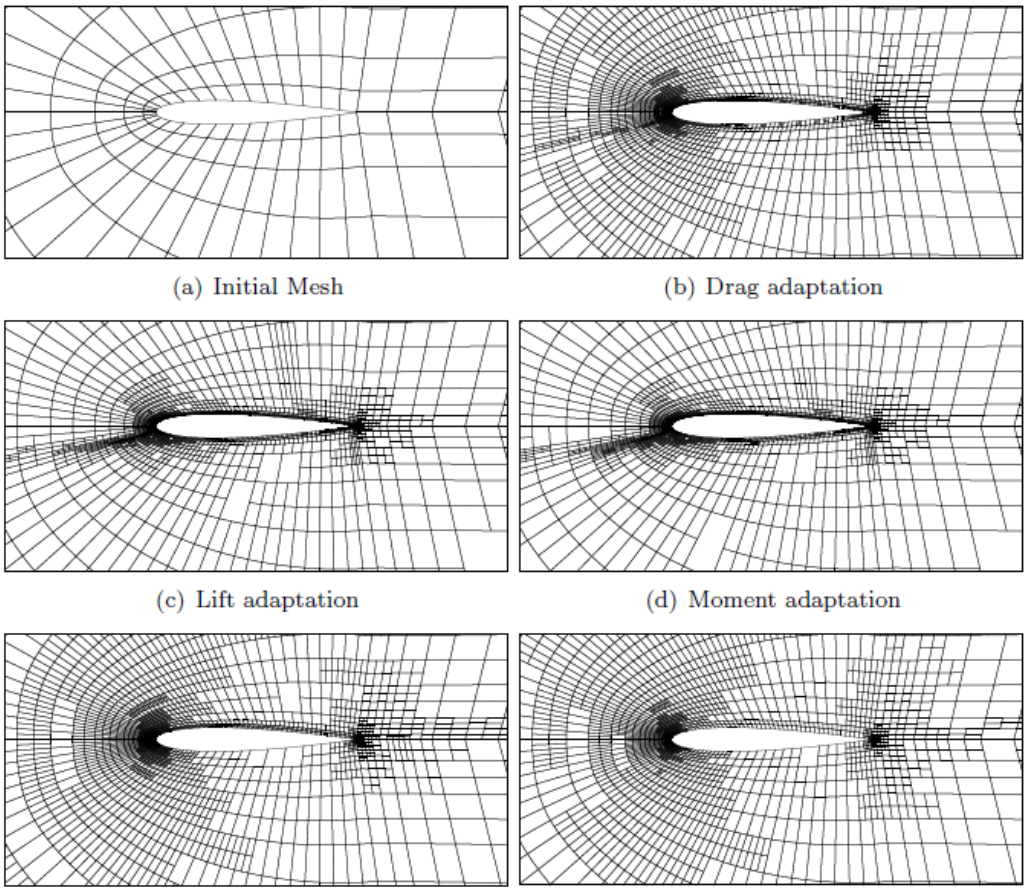



Figure 10.12: The adapted meshes obtained by Fidkowski and Roe on the same test case, using the discrete adjoint approach [12]. Eight levels of anisotropic $h$-refinement are used. Third order DG scheme is used ($N = 3$).

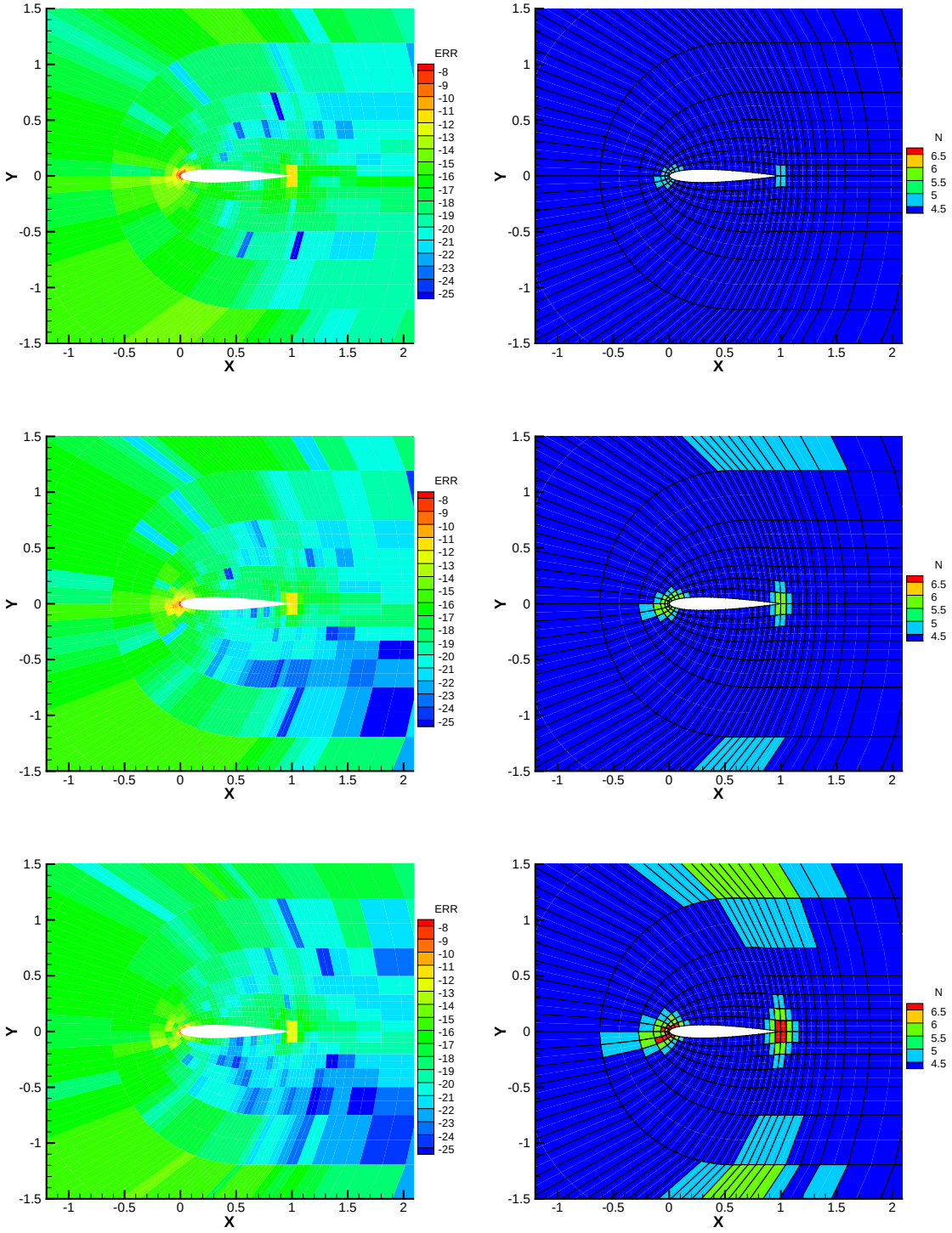



Figure 10.13: Left: error distribution before the 1st, 2nd and 3rd level of $p$-refinement using the entropy adjoint. Right: the order distribution after the 1st, 2nd and 3rd level of $p$-refinement.

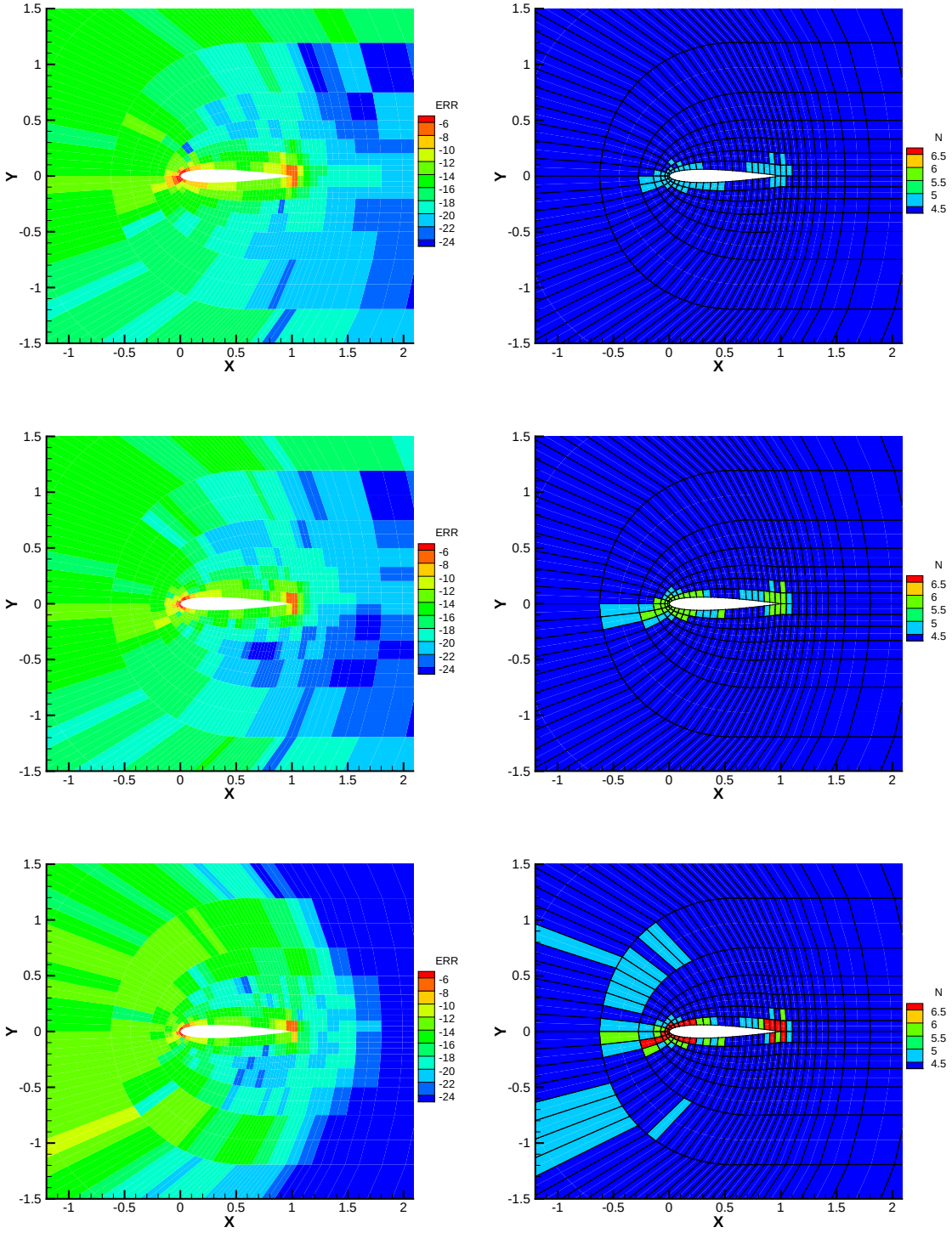



Figure 10.14: Left: error distribution before the 1st, 2nd and 3rd level of $p$-refinement using the drag adjoint. Right: the order distribution after the 1st, 2nd and 3rd level of $p$-refinement.

### 10.2.2 $p$-Refinement

Three levels of $p$-refinement are done, starting with $N = 4$ globally. Figure 10.13 and 10.14 show the results from the entropy and the drag adjoint respectively. The adapted order distributions display very similar patterns to the corresponding adapted meshes using $h$-refinement.

### 10.2.3 Drag convergence plot

'True' drag coefficient is obtained by projecting the drag coefficient values obtained from uniform refinement. The error in $C_D$ is then plotted against the number of DOFs. The drag convergence rates for the entropy adjoint and the drag adjoint are similar. This confirms that the entropy adjoint method gives comparable performance as the drag adjoint, while having the significant advantage that no adjoint solver is needed.

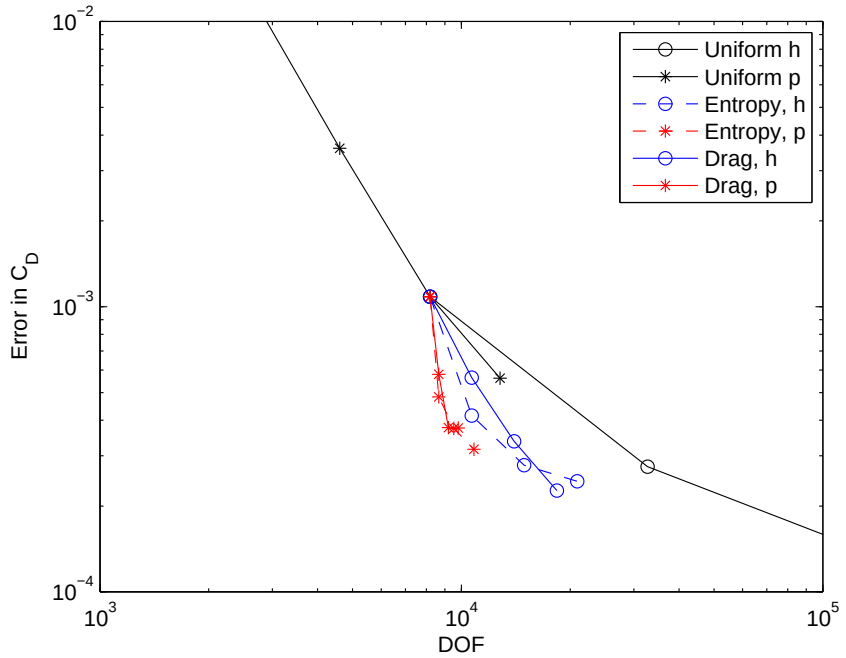



Figure 10.15: Plot of error in drag vs number of degrees of freedom (DOFs) for the NACA 0012 in subsonic flow test case.

## 10.3 Transonic Flow Past an Airfoil

The last test case is a NACA 0012 airfoil in transonic flow, $M = 0.8$, $\alpha = 1.25°$. The initial mesh is the same as the one used for NACA 0012 in subsonic flow, as shown in Figure 10.16. The Mach contour on the initial mesh shows a shockwave on the upper surface of the airfoil. $N = 3$ is used for all tests in this section. Artificial viscosity is used to stabilize the shock. Even so, there are still oscillations in the solution profile, and the shock is smeared across one to two cell widths. To resolve flow discontinuities better, the grid near the shock needs to be finer, and the order of accuracy in these cells is preferably low. In this section, $h$-refinement is tested first, and the results from entropy and drag adjoint are compared. Then $hp$-adaptation is also discussed for this test case.

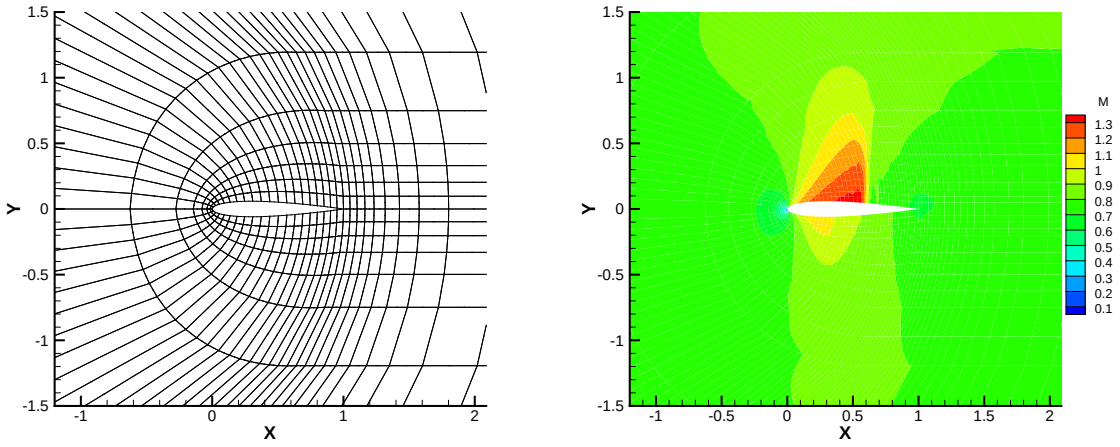



Figure 10.16: Initial mesh and Mach contour for the NACA 0012 test case, $M = 0.8$, $\alpha = 1.25°$.

### 10.3.1 $h$-Refinement

As in the previous test cases, three levels of refinement are conducted with 10% of cells refined at each level. Results using the entropy and the drag adjoint (Figure 10.18 and 10.19) show that both adjoint indicators identify the shocks and the leading and trailing edges as regions to be refined. Drag adjoint traces out the leading

edge stagnation line but entropy adjoint does not. The error plots show this slight discrepancy between the two adjoint indicators. Generally speaking, the adaptation patterns are very similar, implying that there is a high correlation between the cells contributing large errors to drag and those to the net entropy flow out of the domain. It is expected that this observation would extend to the other commonly chosen targeted outputs used for adjoint, e.g. lift and moment coefficients. The Mach contour on the adapted mesh shows a much sharper shock profile (Figure 10.17).

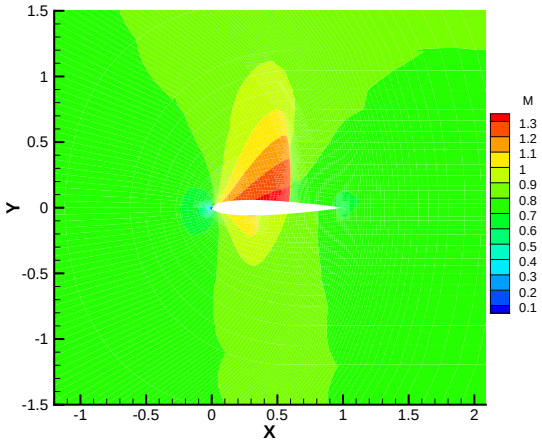



Figure 10.17: The Mach contour on the adapted mesh after three levels of refinement using entropy adjoint.

Another observation is that the continuous adjoint approach meets no difficulties when the flow solution is discontinuous. This is because the adjoint solution is smooth even at flow discontinuities such as shocks. Figure 10.20 shows the drag adjoint solution. All four components of the adjoint vector are smooth near the shock, yet the adjoint weighted residuals yield the error distribution as seen in Figure 10.19 where the shock stands out.

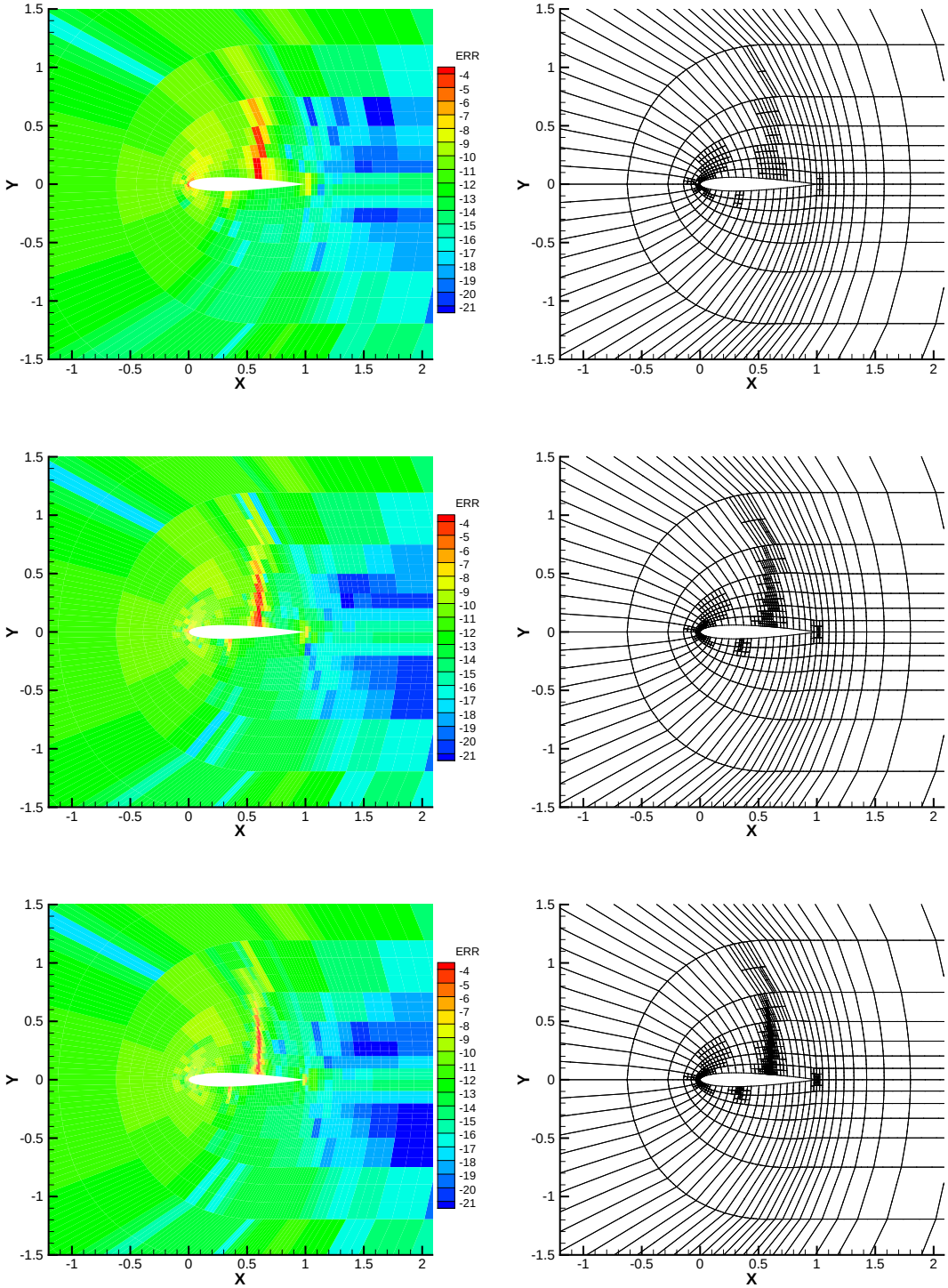



Figure 10.18: Left: error distribution before the 1st, 2nd and 3rd level of $h$-refinement using the entropy adjoint. Right: the mesh after the 1st, 2nd and 3rd level of $h$-refinement.

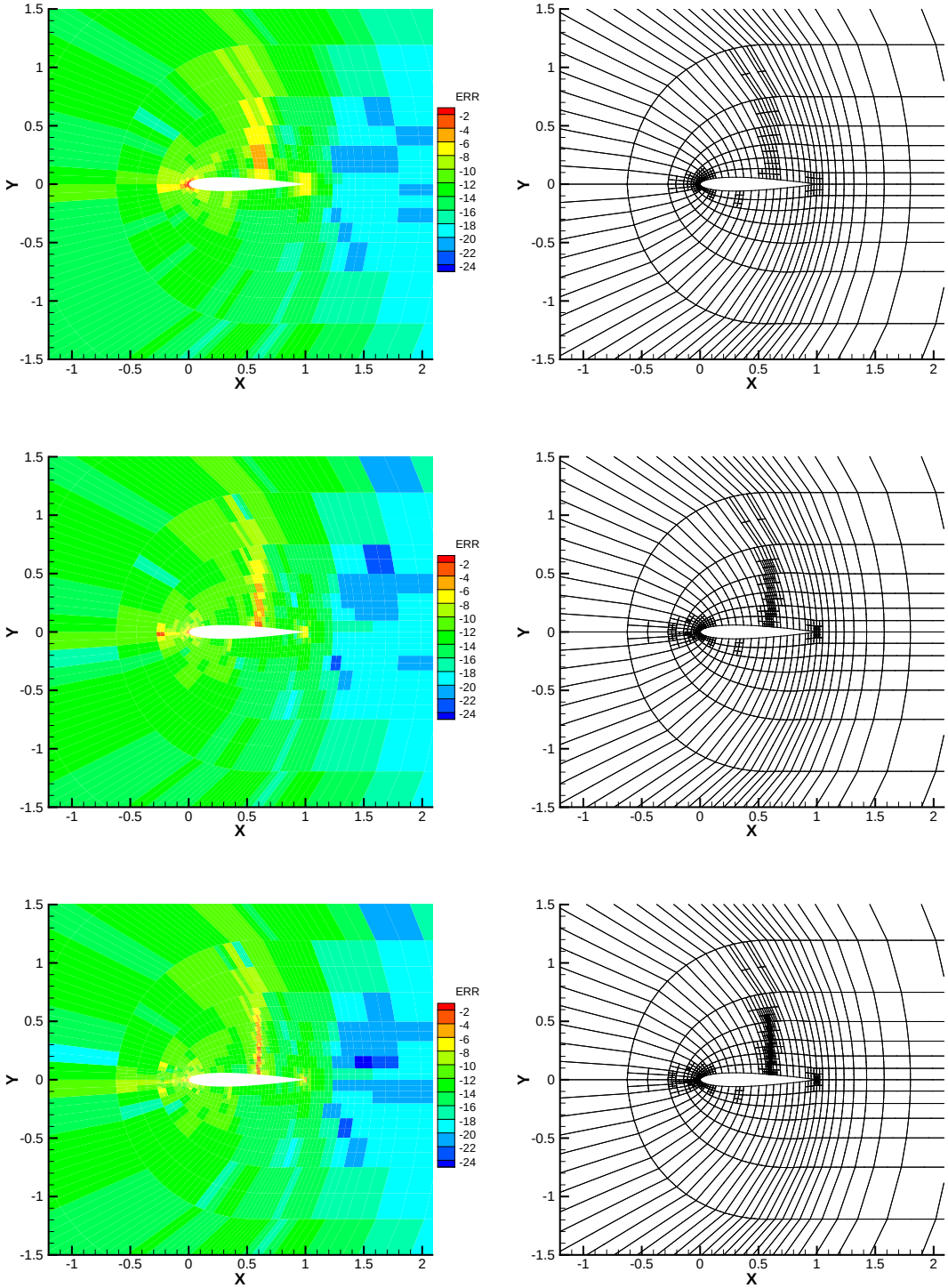



Figure 10.19: Left: error distribution before the 1st, 2nd and 3rd level of $h$-refinement using the drag adjoint. Right: the mesh after the 1st, 2nd and 3rd level of $h$-refinement.

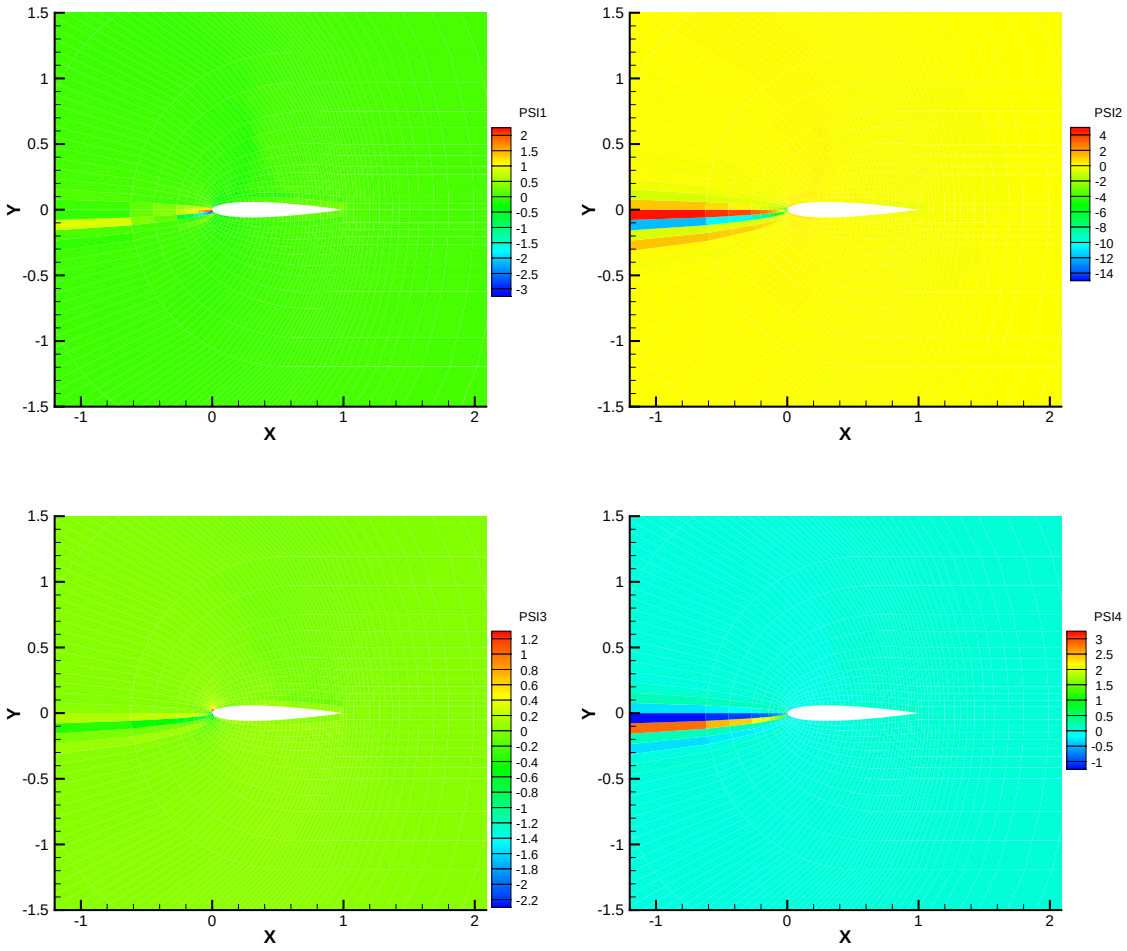



Figure 10.20: The adjoint solution for the drag adjoint equation on the initial mesh of the NACA 0012 airfoil, $M = 0.8$, $\alpha = 1.25°$. Top left: $\psi_1$; top right: $\psi_2$; bottom left: $\psi_3$; bottom right: $\psi_4$.

### 10.3.2 *hp*-Refinement

*hp*-adaptation allows both *h*-refinement and *p*-refinement to be done to a mesh. As discussed for the cylinder test case, when the global order of the scheme is low, *p*-refinement gives a better accuracy for a given number of DOFs. Therefore, a simple strategy for *hp*-adaptation would be to *h*-refine the cells near flow discontinuities, and to *p*-refine the cells with large errors elsewhere. Error estimation is done with the entropy adjoint method. Detection of flow discontinuities is done using the discontinuity sensor given by Equation 3.3. The adaptation procedure has been described in Chapter 9.

Figure 10.21 shows the adapted mesh and order distribution for three levels of *hp*-refinement. The cells near the shock and near the leading edge stagnation point are *h*-refined. Figure 10.22 shows the discontinuity sensor for the solution before each level of adaptation, and it captures the shock well for the unstructured mesh which is not aligned with the shock.

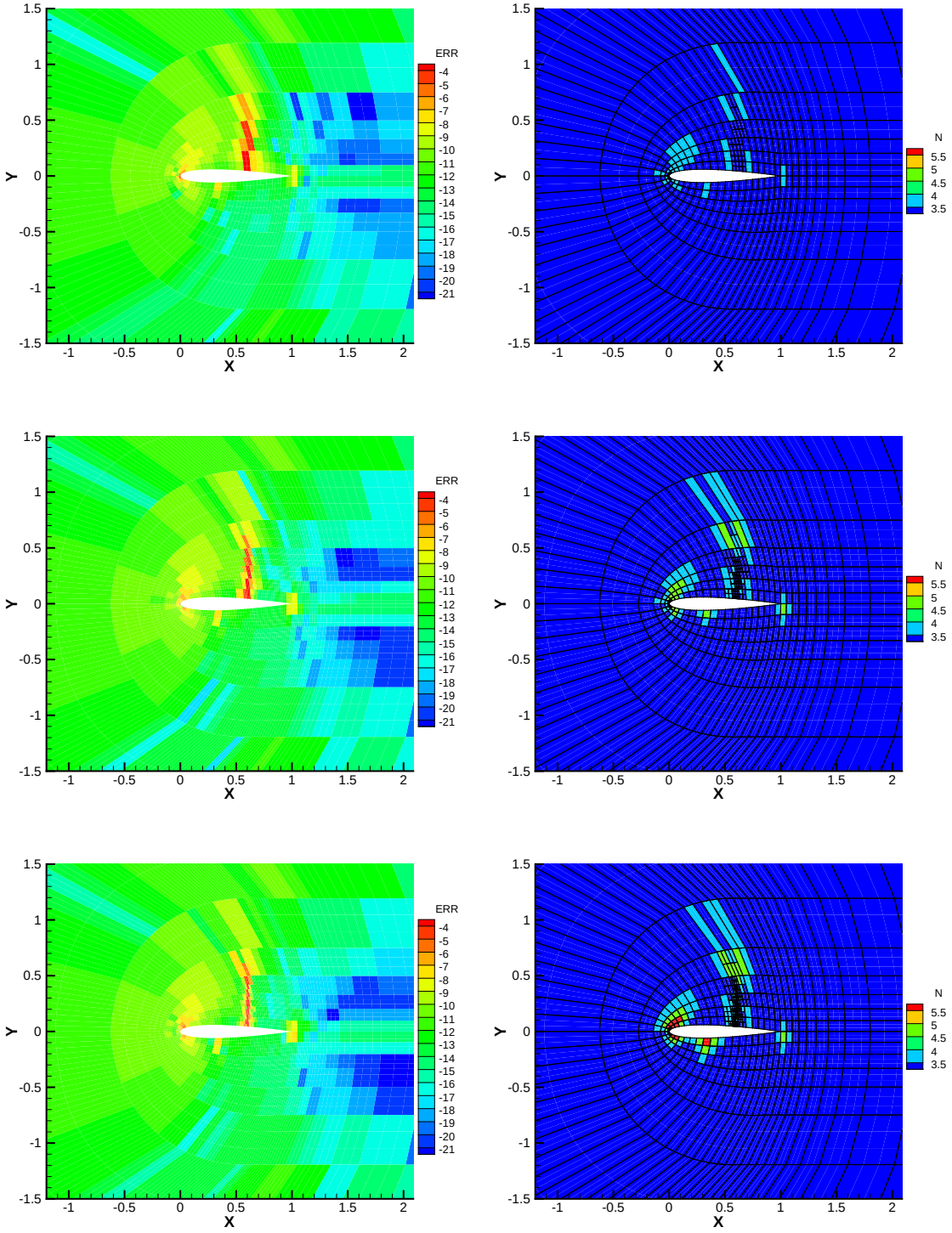



Figure 10.21: Left: error distribution before the 1st, 2nd and 3rd level of *hp*-refinement using the entropy adjoint. Right: the mesh and order distribution after the 1st, 2nd and 3rd level of *hp*-refinement.

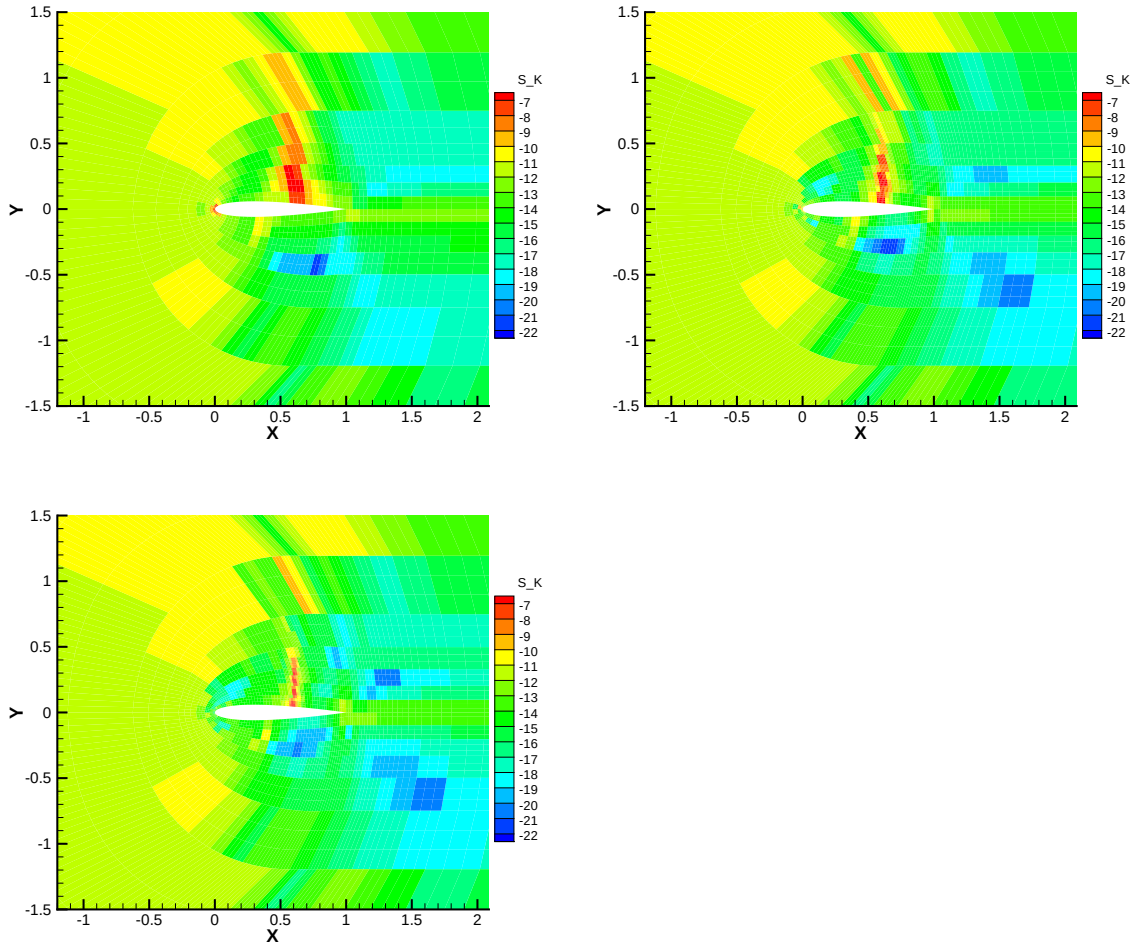



Figure 10.22: The discontinuity sensor $S_k$ before the 1st, 2nd and 3rd level of *hp*-refinement.

### 10.3.3 Drag convergence plot

The drag convergence plot in Figure 10.23 agrees with the previous observation that entropy and drag adjoint indicators give similar performance. The simple strategy used for *hp*-refinement proves to increase the error convergence rate for the first two levels of adaptation. A more robust strategy would be needed to fully explore the potential benefit of *hp*-adaptivity.

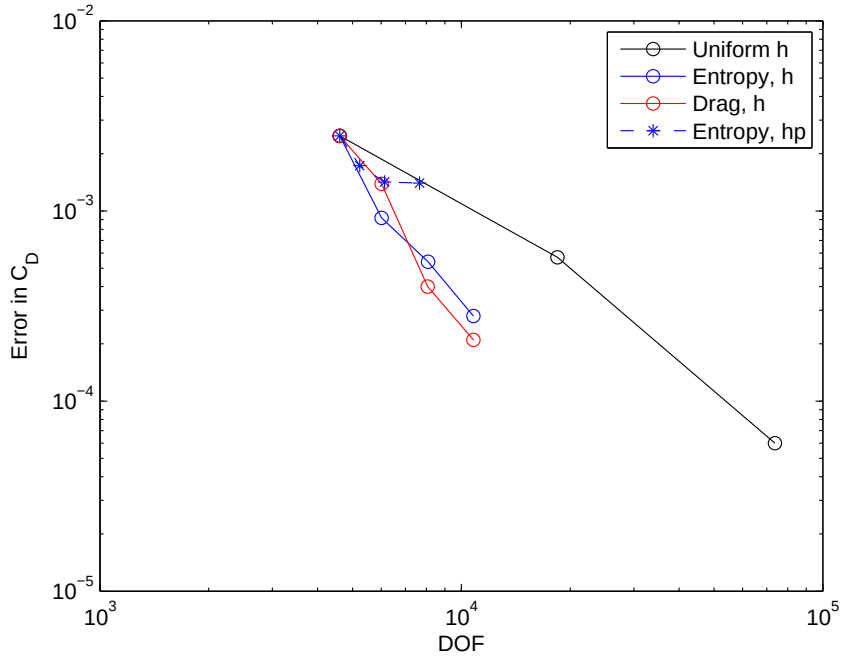



Figure 10.23: Plot of error in drag vs number of degrees of freedom (DOFs) for the NACA 0012 in transonic flow test case.

## 10.4 Output Functionals

In previous sections, results from drag and entropy adjoints are compared and discussed. While the drag adjoint solutions are computed using a high order adjoint solver and the entropy adjoint solutions were computed directly from the flow solutions, the results show that they give similar adapted meshes as well as drag convergence. In practice, there are some other popular choices of output functionals for

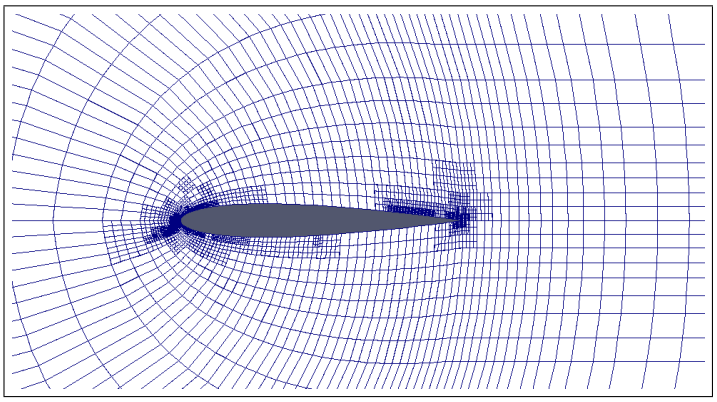
derivation of adjoint equations, such as lift and moment. In theory, only the errors in the targeted output functional are minimized in the adapted meshes. However, if similar cells contribute to errors in various outputs, then the adapted meshes using these various adjoint output functional are also similar, leading to significant error reduction in these outputs simultaneously.

If the accuracies in more than one output are crucial (for example both lift and drag are important for a wing or an aircraft) and these outputs lead to very different adapted mesh patterns, then multiple adjoints could be used in combination. A simple strategy is to let the cell-wise error indicator to be the sum of the normalized cell-wise estimated errors in each output functional.

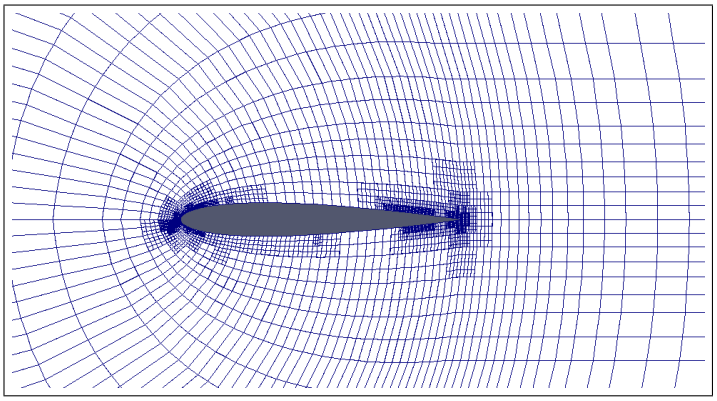
Lift adjoint and a combination of lift and drag adjoints are tested on the NACA 0012 airfoil in subsonic and transonic flows, with identical flow conditions as those in the previous sections. The adapted meshes are shown in Figure 10.24 and 10.25, and the drag convergence and lift convergence plots in Figure 10.26, 10.27, 10.28 and 10.29. For the subsonic case, the refined meshes from lift adjoint and drag adjoint share many similarities. The only major difference is that the cells on the lower surface near the trailing edge are refined by lift adjoint. A combination of lift and drag adjoint leads to a compromise between the two. Overall, the three meshes look highly similar, and the drag convergence and lift convergence plots show that they also give almost identical error convergence rates in $C_L$ and $C_D$. For the transonic case, while the adapted meshes from lift adjoint and from drag adjoint appear highly similar, the combination of lift and drag adjoint results in refinement of cells along the sonic line, too. The drag convergence plot shows that this leads to less error reduction in $C_D$. This suggests that the simple strategy of summing estimated errors in each output functional is not sophisticated enough and needs to be improved.

## 10.5 Cost of the Adjoint Solver

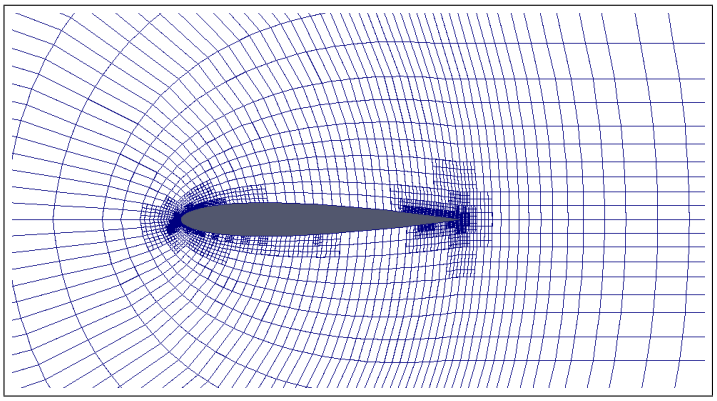
It is noticed during the experiments that the estimated error distribution converges quickly during the adjoint solve. That is, for mesh adaptation purposes, the accuracy of the adjoint solution is not crucial. (This is probably not true if the purpose is to



(a) Drag adjoint

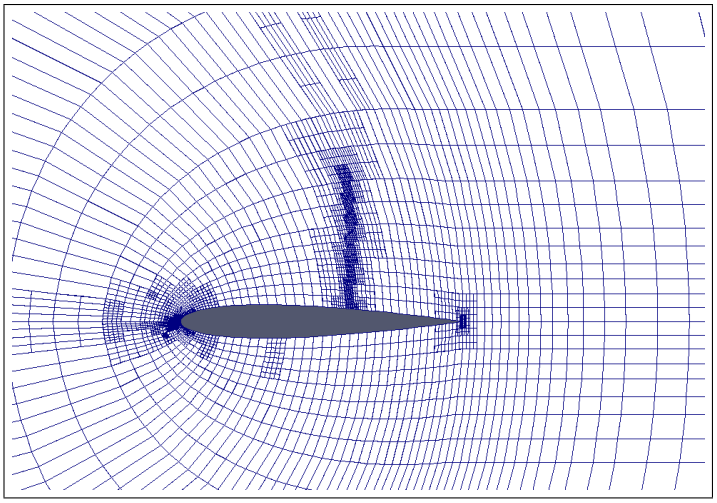


(b) Lift adjoint

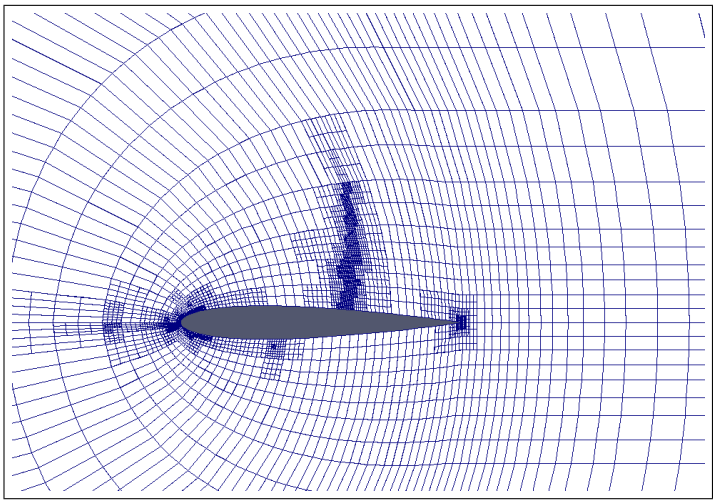


(c) Lift+Drag adjoint

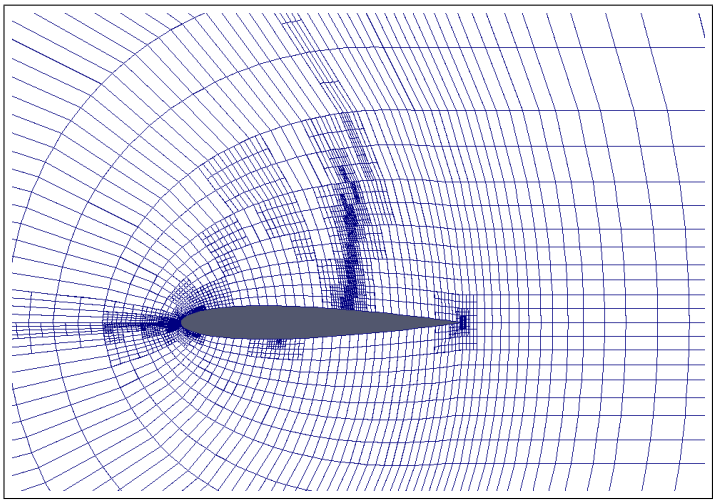
Figure 10.24: NACA 0012, $M = 0.4$, $\alpha = 5°$. Meshes after three levels of $h$-refinement.



(a) Drag adjoint



(b) Lift adjoint



(c) Lift+Drag adjoint

Figure 10.25: NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Meshes after three levels of $h$-refinement.

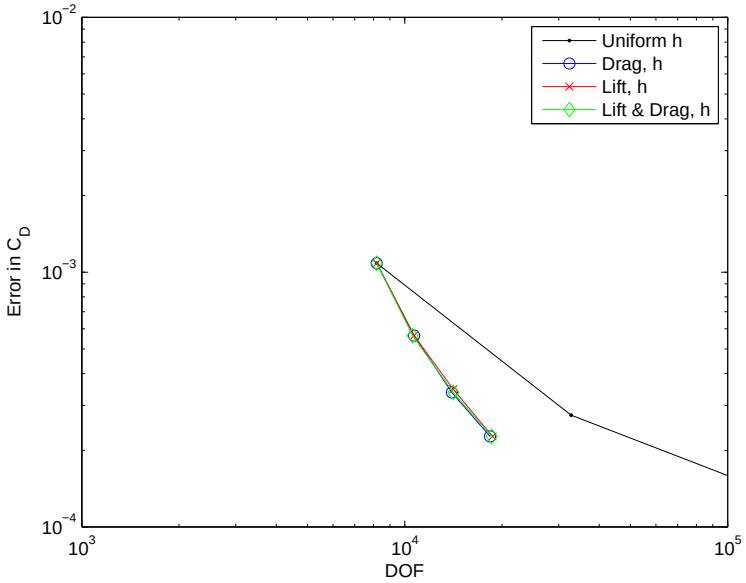



Figure 10.26: NACA 0012, $M = 0.4$, $\alpha = 5°$. Drag convergence plot for different adjoint output functionals.

accurately estimate the error in the target output functional.) This implies potential cost savings from the adjoint solver. There is no need to let the global residual to drop to machine zero for the adjoint solution. The adjoint equations can also be solved with lower orders than the flow equations. In fact, an independent adjoint solver which is cheap and fast would be ideal. This can only be done using the continuous adjoint method, and is one of the main advantages of the continuous adjoint over the discrete adjoint.

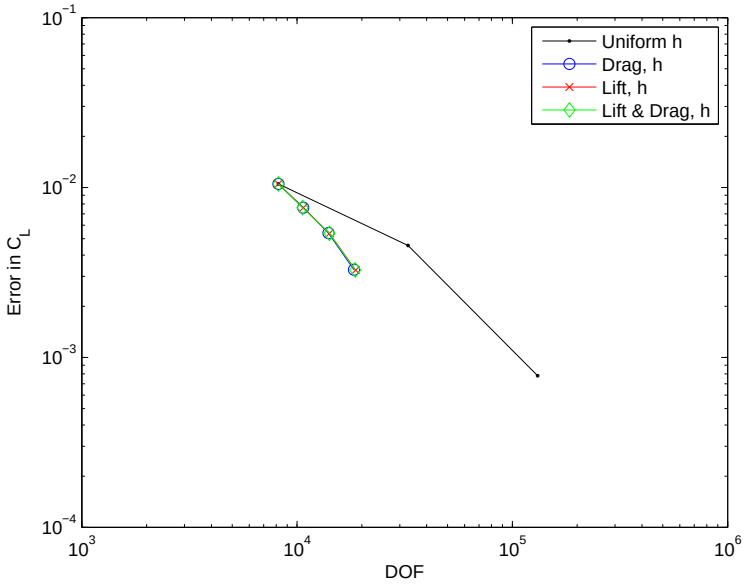



Figure 10.27: NACA 0012, $M = 0.4$, $\alpha = 5°$. Lift convergence plot for different adjoint output functionals.

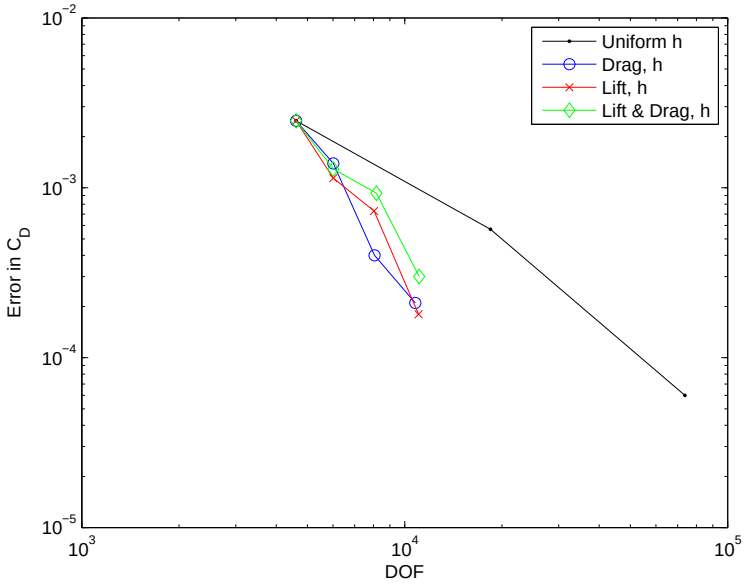



Figure 10.28: NACA 0012, NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Drag convergence plot for different adjoint output functionals.

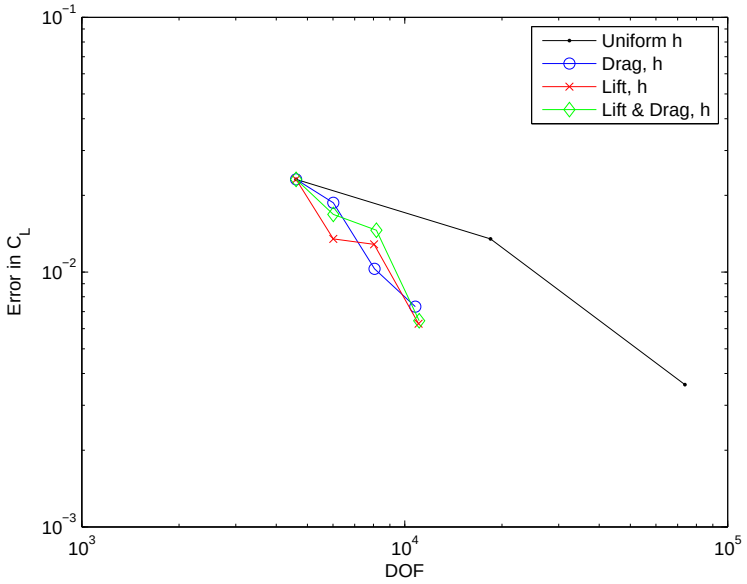



Figure 10.29: NACA 0012, NACA 0012, $M = 0.8$, $\alpha = 1.25°$. Lift convergence plot for different adjoint output functionals.

# Chapter 11

# Conclusions and Future Work

## 11.1 Summary and Conclusions

This thesis has presented an *a posteriori* error estimation for an integral functional of the solution of a general PDE, by formulating the dual or adjoint problem of the original PDE. Specifically, the continuous adjoint equations are derived and solved for Euler equations using the high order spectral difference method. Mesh adaptation is done using this adjoint-based error indicator. Both $h$- and $p$-adaptations are conducted. Both entropy adjoint and drag adjoint are tested. Test cases include a cylinder in subsonic flow, a NACA 0012 airfoil in subsonic flow and in transonic flow. The conclusions are summarized below.

**Continuous adjoint for error estimation and mesh adaptation**

With continuous adjoint mesh adaptation, the estimated errors in the refined cells are reduced, i.e. adaptation drives error towards equi-distribution. Functional (e.g. drag) error convergence against DOFs is faster for adapted meshes than uniformly refined meshes.

Qualitative comparisons are made from the adapted meshes from this work and those in the literature obtained using discrete adjoint. Both of them yield very similar adaptation patterns. For example, for a lifting NACA 0012 airfoil in subsonic flow, refinement occurs around the leading and trailing edges and along the stagnation line.

For the same airfoil in transonic flow, shock regions are also refined. In addition, the solution of the continuous adjoint equation is smooth across shocks, eliminating the concern that the continuous adjoint method may not handle flow discontinuities well.

It is also discovered that the accuracy of the adjoint solution is not crucial for mesh adaptation. The error distribution plot stabilizes after merely two to three orders of magnitude drop in global residual of the adjoint solution, and the refinement pattern remains the same for more accurate adjoint solutions.

These observations suggest that the continuous adjoint method, while providing similar performance as the discrete adjoint, has several advantages that can be exploited. Potentially, one stand-alone adjoint solver can be partnered with any flow solver, and the adjoint solver can be designed to trade accuracy for speed (e.g. using a low order finite volume scheme with multigrid). The cost saving is particularly significant if the flow solver uses a high order method with explicit time-stepping, where the implementation complexity and computational overhead for the discrete adjoint approach is very high.

High order methods are gaining more popularity recently, and meshes suitable for these methods can be very different from those used for the conventional low order methods. Since the gist of high order methods is to obtain more accurate solutions at a given cost, mesh efficiency becomes particularly important. Therefore, we believe that mesh adaptation will become an indispensable tool in the future.

**The entropy adjoint approach**

The entropy adjoint method proposed by Fidkowski and Roe [12] is also tested in this work. In this approach no adjoint equations need to be solved. The entropy variables which can be calculated from the flow solutions directly are proved to automatically satisfy the adjoint equations targeting the net entropy flow out of the domain. Compared with the drag adjoint on adaptation performance for NACA 0012 airfoil in subsonic and transonic flows, the entropy adjoint, although targeting a different output, performs as well as the drag adjoint. The adaptation patterns are very similar. The error in drag vs DOF plots show that the drag error convergence rates for entropy and drag adjoint are comparable. This suggests that regions contributing

large errors in one output are likely to be contributing large errors in other outputs, so that the choice of target output does not have a large impact on refinement results. The entropy adjoint method, therefore, is proved to be an effective yet cheap mesh adaptation approach for Euler equations. However, this observation may not hold for viscous flows, where the entropy adjoint error indicator is likely to track down the cells with large entropy errors, e.g. the wake, but the other adjoint indicators may behave differently.

**$h$- vs $p$-adaptation**

$h$- and $p$-adaptations are compared for a cylinder in subsonic flow. For $h$-refinement, it is observed that the refinement pattern depends on the global order of the flow solver scheme $N$. When $N$ is small, the initial error tends to be large and less concentrated, and therefore the regions refined are more spread. When $N$ is large, the initial error is small in regions of smooth flows, and therefore the regions refined are more concentrated around places such as body surfaces, leading and trailing edges of an airfoil, etc.

It is known that in absence of flow discontinuities, high order schemes provide better accuracies than low order schemes for the same number of DOFs. Therefore, if the global order of accuracy is low, then $p$-refinement is expected to offer a better accuracy than $h$-refinement for a given number of DOFs. This is confirmed by the results from the cylinder test case for $N = 3$. This also suggests that $h$-refinement is best used with a higher order scheme.

$hp$-adaptation is also briefly attempted based on the idea that $p$-refinement is more suitable for smooth flows while $h$-refinement better resolves flow discontinuities. A discontinuity sensor is used. With this simple strategy, $hp$-adaptation on a NACA 0012 airfoil in transonic flow is shown to give comparable performance as $h$-refinement. More subtle and sophisticated strategies will be needed to exploit the potential benefits of $hp$-adaptation.

## 11.2 Future Work

**Extension to Navier-Stokes equations and simplex elements**

The continuous adjoint method and the entropy adjoint approach presented in this thesis can both be extended to Navier-Stokes equations and to three-dimensional flows in the future. For viscous flows, it is worth investigating whether the entropy adjoint indicator still behaves similarly as the other adjoint indicators.

Mesh adaptation mechanisms are also more complicated in three dimension grids. For triangular and tetrahedral elements, $h$-refinement can become a mixture of refinement and mesh movement. In practice, for complex geometries, an unstructured mesh with simplex elements or mixed elements is much easier to construct than with only hexahedral elements. Therefore a code should be capable of adapting all types of elements. Anisotropic $h$-refinement should also be studied for both quadrilateral/hexahedral and triangular/tetrahedral meshes, as it is potentially suitable for boundary layers, shocks, etc.

***hp*-adaptation**

*hp*-adaptation has the potential of achieving exponential functional convergence. It combines $h$- and $p$-adaptation and the advantages of both, and it fully makes use of the flexibility of high order methods. For example, *hp*-adaptation can help with shock capturing, which is one of the main challenges for high order methods. Flow discontinuities are best presented by low order (e.g. first order) solutions over small cell widths. Therefore one idea is a combination of $h$-refinement and $p$-coarsening around flow discontinuities.

**A universal adjoint solver**

Another future project can be a black-box implementation of a stand-alone adjoint solver which takes in a mesh and the flow solutions and outputs the adjoint solutions and the estimated errors. For the purpose of mesh adaptation, the cost of the adjoint solver is the top priority.

# Bibliography


[1] M.J. Aftosmis and M.J. Berger. Multilevel error estimation and adaptive *h*-refinement for cartesian meshes with embedded boundaries. *AIAA-2002-0863*, 2002.

[2] J. Andren, H. Gao, M. Yano, D.L. Darmofal, C. Ollivier-Gooch, and Z.J. Wang. A comparison of higher-order methods on a set of canonical aerodynamics applications. *AIAA-2011-3230*, 2011.

[3] T.J. Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Element Anal. Design*, 25:243–273, 1997.

[4] L. Baskett and R. Haimes. Feature extraction of shear layers. *AIAA-2001-2665*, 2001.

[5] Boeing. `http://www.boeing.com/commercial/787family/programfacts.html`.

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

[7] L. Demkowicz, W. Rachowicz, and Ph. Devloo. A fully automatic *hp*-adaptivity. *Journal of Scientific Computing*, 17:117–142, 2002.

[8] B. Douglas. Super computing at boeing commercial airplanes. `http://www.nitrd.gov/subcommittee/hec/hecrtf-outreach/sc03/sc03_hecrtf_%dball.pdf`.

[9] K.J. Fidkowski. *A Simplex Cut-Cell Adaptive Method for High-Order Discretizations of the Compressible Navier-Stokes Equations*. PhD thesis, MIT, 2007.

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

[11] K.J. Fidkowski and P.L. Roe. Entropy-based mesh refinement, i: The entropy adjoint approach. *AIAA Paper 2009-3790*, 2009.

[12] K.J. Fidkowski and P.L. Roe. An entropy adjoint approach to mesh refinement. *SIAM J. Sci. Comput*, 32(3):1261–1287, 2010.

[13] M.B. Giles and N.A. Pierce. Adjoint equations in cfd: Duality, boundary conditions and solution behavior. *AIAA-97-1850*, 1997.

[14] M.B. Giles and N.A. Pierce. Improved lift and drag estimates using adjoint euler equations. *AIAA-99-3293*, 1999.

[15] A. Harten. On the symmetric form of systems of conservation laws with entropy. *Journal of Computational Physics*, 49:151–164, 1983.

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

[17] J.E. Hicken and D.W. Zingg. The role of dual consistency in functional accuracy: Error estimation and superconvergence. *AIAA 2011-3855*, 2011.

[18] P. Houston and E. Suli. 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.

[19] H.T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous galerkin methods. *AIAA Computational Fluid Dynamics Meeting*, 2007.

[20] A. Jameson. Solution of the euler equations for two dimensional transonic flow by a multigrid method. *Applied Mathematics and Computation*, 13:327–356, 1983.

[21] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988.

[22] A. Jameson. Aerodynamic design via control theory. *Science*, 245:361–371, 1989.

[23] A. Jameson. A proof of the stability of the spectral difference method for all orders of accuracy. *Journal of Scientific Computing*, 45(1-3):348–358, 2010.

[24] A. Jameson, N. Pierce, and L. Martinelli. Optimum aerodynamic design using the navier-stokes equations. *Journal of Theoretical and Computational Fluid Mechanics*, 10:213–237, 1998.

[25] D.A. Kopriva. A conservative staggered-grid chebysheve multidomain method for compressible flows. ii. a semi-structured method. *Journal of Computational Physics*, 128:475–488, 1996.

[26] K. Laflin, J. Vassberg, R. Wahls, J. Morrison, O. Brodersen, M. Rakowitz, E. Tinoco, and J.-L. Godard. Summary of data from the second aiaa cfd drag prediction workshop. *AIAA-2004-0555*, 2004.

[27] D. Levy, T. Zickuhr, J. Vassberg, S. Agrawal, R. Wahls, S. Pirzadeh, and M. Hemsch. Data summary from the first aiaa cfd drag prediction workshop. *Journal of Aircraft*, 40(5):875–882, 2003.

[28] L.Y. Li, Y. Allaneau, and A. Jameson. Continuous adjoint approach for adaptive mesh refinement. *AIAA Paper 2011-3982*, 2011.

[29] Y. Li, S. Premasuthan, and A. Jameson. Comparison of h- and p- adaptations for spectral difference methods. *AIAA Paper 2010-4435*, 2010.

[30] Y. Liu, M. Vinokur, and Z.J. Wang. Spectral difference method for unstructured grids i: Basic formulation. *Journal of Computational Physics*, 216:780–801, 2006.

[31] J.C.C. Lu. *An A Posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin of Optimal Control Problems*. PhD thesis, Massachusetts Institute of Technology, 2005.

[32] D. Mavriplis. Third drag prediction workshop results using the nsu3d unstructured mesh solver. *Journal of Aircraft*, 45(3):750–761, 2008.

[33] G. May and A. Jameson. A spectral difference method for the euler and navier stokes equations. *AIAA 2006-304*, 2006.

[34] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA-2000-0667*, 2000.

[35] S. Nadarajah and A. Jameson. Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization. *AIAA-2001-2530*, 2001.

[36] S.K. Nadarajah. *The Discrete Adjoint Approach to Aerodynamic Shape Optimization*. PhD thesis, Stanford University, 2003.

[37] J.T. Oden, L. Demkowicz, W. Rachowicz, and T.A. Westermann. Toward a universal *h-p* adaptive finite element strategy, part 2. a posteriori error estimation. *Computer Methods in Applied Mechanics and Engineering*, 77:113–180, 1989.

[38] T. Oliver. *A High-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations*. PhD thesis, MIT, 2008.

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

[40] J. Peraire, M. Vahdati, K. Morgan, and O.C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72:449–466, 2002.

[41] P.O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous galerkin methods. *AIAA 2006-112*, 2006.

[42] N.A. Pierce and M.B. Giles. Adjoint recovery of superconvergent functionals from pde approximations. *SIAM Review*, 42(2):247–264, 2000.

[43] S. Premasuthan. *Towards and Efficient and Robust High Order Accurate Flow Solver for Viscous Compressible Flow*. PhD thesis, Stanford University, 2010.

[44] S. Premasuthan, C. Liang, and A. Jameson. A spectral difference method for viscous compressible flows with shocks. *AIAA Paper 2009-3785*, 2009.

[45] S. Premasuthan, C. Liang, and A. Jameson. Computation of flows with shocks using spectral difference scheme with artificial viscosity. *AIAA Paper 2010-1449*, 2010.

[46] W. Rachowicz, J.T. Oden, and L. Demkowicz. Toward a universal *h-p* adaptive finite element strategy, part 2. design of *h-p* meshes. *Computer Methods in Applied Mechanics and Engineering*, 77:181–212, 1989.

[47] W. Rachowicz, D. Pardo, and L. Demkowicz. Fully automatic *hp*-adaptivity in three dimensions. *Computer Methods in Applied Mechanics and Engineering*, 195:4816–4842, 2006.

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

[49] V.V. Rusanov. Calculation of interaction of non-steady shock waves with obstacles. *Journal of Computational and Mathematical Physics USSR*, 1:267–279, 1961.

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

[51] Y. Sun, Z.J. Wang, and Y. Liu. High-order multidomain spectral difference method for the navier-stokes equations on unstructured hexahedral grids. *Communications in Computational Physics*, 2:310–333, 2007.

[52] J. Vassberg, E. Tinoco, M. Mani, O. Brodersen, B. Eisfeld, R. Wahls, J. Morrison, T. Zickuhr, K. Laflin, and D. Mavriplis. Summary of the third aiaa cfd drag prediction workshop. *AIAA-2007-260*, 2007.

[53] J. Vassberg, E. Tinoco, M. Mani, B. Rider, T. Zickuhr, D. Levy, O. Brodersen, B. Eisfeld, S. Crippa, R. Wahls, J. Morrison, D. Mavriplis, and M. Murayama. Summary of the fourth aiaa cfd drag prediction workshop. *AIAA-2010-4547*, 2010.

[54] D.A. Venditti. *Grid Adaptation for Functional Outputs of Compressible Flow Simulations*. PhD thesis, MIT, 2002.

[55] D.A. Venditti and D.L. Darmofal. Adjoint error estimation and grid adaptation for funcional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164:204–227, 2000.

[56] D.A. Venditti and D.L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176:40–69, 2002.

[57] Z.J. Wang, Y. Liu, G. May, and A. Jameson. Spectral difference method for unstructured grids ii: Extension to the euler equations. *Journal of Scientific Computing*, 32:45–71, 2006.

[58] G.P. Warren, W.K. Anderson, J.T. Thomas, and S.L. Krist. Grid convergence for adaptive methods. *AIAA-91-1592*, 1991.

[59] G. Xia, D. Li, and C.L. Merkle. Anisotropic grid adaptation on unstructured meshes. *AIAA-2001-0443*, 2001.

[60] X.D. Zhang, M.-G. Vallet, J. Dompierre, P. Labbe, D. Pelletier, J.-Y. Trepanier, R. Camarero, J.V. Lassaline, L.M. Manzano, and D.W. Zingg. Mesh adaptation using different error indicators for the euler equations. *AIAA-2001-2549*, 2001.