An adaptive framework for high-order, mixed-element numerical simulations

ARCHIVES
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUN 16 2014
LIBRARIES

by

Philip Claude Delhaye Caplan

B.Eng., McGill University (2012)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

© Massachusetts Institute of Technology 2014. All rights reserved.

Signature redacted

Author ....
Department of Aeronautics and Astronautics
May 22, 2014

Signature redacted

Certified by. .....
David L. Darmofal
Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Accepted by ....
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

An adaptive framework for high-order, mixed-element numerical simulations

by

Philip Claude Delhaye Caplan

Submitted to the Department of Aeronautics and Astronautics
on May 22, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This work builds upon an adaptive simulation framework to allow for mixed-element meshes in two dimensions. Contributions are focused in the area of mesh generation which employs the L^k norm to produce various mesh types. Mixed-element meshes are obtained by first using the L^∞ norm to create a right-triangulation which is then combined to form quadrilaterals through a graph-matching approach. The resulting straight-sided mesh is then curved using a nonlinear elasticity analogy. Since the element sizes and orientations are prescribed to the mesh generator through a field of Riemannian metric tensors, the adaptation algorithm used to compute this field is also discussed.

The algorithm is first tested through the L^2 error control of isotropic and anisotropic problems and shows that optimal mesh gradings can be obtained. Problems drawn from aerodynamics are then used to demonstrate the ability of the algorithm in practical applications. With mixed element meshes, the adaptation algorithm works well in practice, however, improvements can be made in the cost and error models. In fact, using the L^∞-generated meshes inherits the same properties of traditional triangulations while adding structure to the mesh. The use of the L^∞-norm in generating tetrahedral meshes is worth pursuing in the future.

Thesis Supervisor: David L. Darmofal
Title: Professor of Aeronautics and Astronautics

Acknowledgments

My experiences at MIT taught me more about life than they did about equations. I need to thank everyone who was involved in making my time here so memorable.

I start with my team of advisors, who pushed me when I needed to be pushed but whose knowledge, insight and humour kept me motivated over the past two years. Prof. David Darmofal, thank you for supporting me in exploring a wide range of topics and for encouraging me to always dig further. Dr. Steven Allmaras, your humble knowledge of all things CFD contributed to such a vast learning experience. Marshall, your coding practices leave me well-prepared for my next chapter (yes, I'll write unit tests). Bob, your honest opinion of my work was always appreciated; thank you for being available over the past few years and I will miss working with you.

I must also acknowledge the influence of my undergraduate professors at McGill University for shaping me into the engineer I am today: Prof. Charles Roth for giving me the opportunity to teach at an undergraduate level and Prof. John Lee for instilling a deep understanding of the fundamentals. I am eternally grateful to my mentor, Prof. Siva Nadarajah, for introducing me to reasearch in CFD and for leaving me well-prepared for the challenges at MIT.

In September 2012, Steven, Jun and I were thrown into 31-213 and I could not have been luckier to have shared so many laughs with you guys; you really made this experience worthwhile. Huafei, thanks for being so patient while getting us started with ProjectX. Carlee, Jeff, Savithru and Yixuan, thank you for restoring a fun atmoshpere in our office as we all complained about writing our theses. Gracias à los españoles para colonizar edificio 31: David, Ferran, Joel – this of course also includes Hemant, Abby and Dr. Nguyen. I will miss hearing the mix of Spanish and Catalan shouts as a fútbol match plays in the background. In particular, I thank Xevi for sharing his expertise in meshing, geometry and C++; your encouraging words kept

me on track i prometo que aprendré Haskell a l'estiu. I'd also like to thank Eric Dow for making sure all the computational resources in the ACDL were functioning smoothly and for teaching me the ways of system administration. Thank you to the second half of the ACDL: Patrick, Alessio, Chaitanya, Rémi, Giulia, Marc and Xun for getting everyone out of the lab to relax.

I thank all my roommates and neighbours for putting up with my drumming and to my friends for making my time outside the lab so enjoyable: Fernando, Michael, Rafa, Sami, Paul, Kavitha, Luis, Irene, Tim, Mélanie and Xavier.

Maman, your hourly messages made me feel as if I were back home in Montreal and I could not have completed this work without your unconditional love and support. Dad, you've been a source of inspiration in my pursuit of engineering studies and a role model and I thank you for always being there to support me. Je remercie aussi Grandmaman, Grandpapa and Nani for your endless encouragement throughout my studies.

This thesis is dedicated to my two favourite people with whom I can share anything. Despite 500 km between us, I love how our Skype calls are reminiscent of the times when we were 4, 7 and 9 years old. Thank you to my two brothers and best friends, Ryan and Alex.

Finally, I would like to acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada, the NASA Cooperative Agreement (# NNX12AJ75A) and ANSYS through their fellowship program.

Contents

1 Introduction **15**

1.1 Motivation 15

1.2 Outline 21

1.3 Background 22

1.3.1 Error estimation and adaptation 22

1.3.2 Mesh generation 22

2 Discretization, error estimation and adaptation **27**

2.1 Numerical discretization 27

2.2 Output error estimation 28

2.3 Metric construction 30

2.3.1 Mesh-metric duality 30

2.3.2 Mesh optimization via error sampling and synthesis 33

3 Mesh generation **37**

3.1 Fundamentals 37

3.1.1 Geometry representation 38

3.1.2 Metric field evaluation 38

3.1.3 Right triangle mesh generation 39

3.2 Surface mesh generation 41

3.3 Volume mesh generation 42

3.4 Curvilinear mesh generation 44

3.5 Examples 46

4 Numerical results **51**

4.1 L^2 error control 51

4.1.1 r^α-type corner singularity 52

4.1.2 $2d$ boundary layer 54

4.2 Compressible Navier-Stokes equations 59

4.2.1 Euler equations 59

4.2.2 Reynolds-averaged Navier-Stokes equations 69

4.2.3 Zero pressure gradient turbulent flat plate 71

4.2.4 Turbulent transonic RAE2822 airfoil 77

5 Conclusions **83**

5.1 Summary 83

5.2 Future work 85

5.2.1 Improving the local error model 85

5.2.2 Extension of the mesh generation algorithm to higher dimensions 85

5.2.3 Mesh curving 86

5.2.4 Alternative discretisations 87

5.3 Concluding remarks 87

A Improving the local error model **89**

A.1 Error modeling techniques 89

A.2 Error model rates 90

A.3 Application to L^2 error control 93

B Connectivity tables for the embedded discontinuous Galerkin method **97**

List of Figures

1-1 Memory footprint of d-simplices versus d-cubes for DG, HDG and EDG discretisations; a factor of 1 indicates both shapes incur the same memory cost 19

1-2 Memory footprint of equilateral tetrahedra versus right tetrahedra for DG, HDG and EDG discretisations 19

1-3 Reference shapes in two dimensions 20

1-4 Adaptive simulation framework 21

1-5 Initial and adapted mesh obtained using the quadrilateral subdivision technique, reproduced with permission from Marco Ceze and Krzysztof J. Fidkowski [10] 23

2-1 Mesh-metric duality 31

2-2 Triangle split configurations 34

2-3 Quadrilateral split configurations 34

3-1 Mesh generation algorithm 37

3-2 Interpretation of a right triangle with respect to a unit L^k ball 39

3-3 Graph-matching approach 40

3-4 Curve metric description with true geometry 41

3-5 Force used to elongate or compress mesh edges according to their length under a metric field 44

3-6 Primitive mesh operations 45

3-7 Edge length and element quality distributions for the cylinder example with an analytic metric field 48

3-8 Edge length and element quality distributions for the three-element MDA airfoil example with a discrete metric field 48

3-9 Meshes generated by analytic (cylinder) and discrete (three-element MDA airfoil) metric fields using equilateral (L^2), right (L^∞) and mixed-element mesh generation techniques 49

4-1 Error and degrees of freedom convergence for the corner singularity problem 55

4-2 $p = 1$, dof = 4,000 optimized for the corner singularity problem, $k^* = 0.44$ 56

4-3 $p = 3$, dof = 4,000 optimized for the corner singularity problem, $k^* = 0.67$ 57

4-4 Error and degrees of freedom convergence for the $2d$ boundary layer problem. The refinement factor for triangulations is set to $r = 2$ whereas it is set to $r = 4$ for mixed-element meshes. 60

4-5 $p = 3$, dof = 4,000 optimized mixed element meshe along with corresponding size and aspect ratio distributions for the $2d$ boundary layer problem with a refinement factor of $r = 2$, $k^*_{h_1} = 22.5$, $k^*_{ar} = -25$ . . . 60

4-6 $p = 1$, dof = 4,000 optimized meshes along with corresponding size and aspect ratio distributions for the $2d$ boundary layer problem, $k^*_{h_1} = 41.7$, $k^*_{ar} = -50$. The refinement factor for triangulations is set to $r = 2$ whereas it is set to $r = 4$ for mixed-element meshes. 61

4-7 $p = 3$, dof = 4,000 optimized meshes along with corresponding size and aspect ratio distributions for the $2d$ boundary layer problem, $k^*_{h_1} = 22.5$, $k^*_{ar} = -25$. The refinement factor for triangulations is set to $r = 2$ whereas it is set to $r = 4$ for mixed-element meshes. 62

4-8 Supersonic wedge problem setup 63

4-9 $p = 1$, 30,000 degree-of-freedom optimized meshes for the supersonic wedge case 65

4-10 $p = 3$, 30,000 degree-of-freedom optimized meshes for the supersonic wedge case 66

4-11 Mach number, total pressure and total temperature distributions across a horizontal line at $y = 0.4$ for the supersonic wedge case for all mesh types with $p = 1$ and $p = 3$ discretisations 67

4-12 Artificial viscosity added near shocks for the $p = 3$ solution to the supersonic wedge case with L^2, L^∞ and mixed-element meshes 68

4-13 Convergence rates for $p = 1$ and $p = 3$ adaptations for the supersonic wedge case with L^2, L^∞ and mixed-element meshes 68

4-14 $p = 1$, 20k dof-optimized mixed-element mesh for the zero pressure gradient flat plate case 73

4-15 $p = 1$, 20k dof-optimized triangulations for the zero pressure gradient flat plate case 74

4-16 $p = 2$, 20k dof-optimized mixed-element mesh for the zero pressure gradient flat plate case 75

4-17 $p = 2$, 20k dof-optimized triangulations for the zero pressure gradient flat plate case 76

4-18 Convergence rates for $p = 1$ and $p = 2$ adaptations for the zero pressure gradient flat plate case with all element types 76

4-19 Adapted meshes for the turbulent, transonic RAE2822 case at $p = 1$ and $p = 2$ solution orders 80

4-20 Recombined L^∞ meshes for the turbulent, transonic RAE2822 case at $p = 1$ and $p = 2$ solution orders 81

4-21 Mach number and mass adjoint for the turbulent, transonic RAE2822 case 81

4-22 Drag error estimate and degree-of-freedom convergence for the turbulent, transonic RAE2822 case 82

5-1 Ideal tetrahedron and triangular-based prism suitable for recombination into hexahedra 86

5-2 Mapping between ideal, master and physical elements for a $q = 3$ quadrilateral 87

5-3 Mesh of Ursa generated by converting a grayscale image to an isotropic metric tensor field within a rectangular domain 88

A-1 $1d$ split configuration 90

A-2 Effective exponent in error model of Eq. (A.3) obtained using both error modeling technqiues 93

A-3 Errors and rates obtained with both models on a coarse mesh, $p = 3$, $n = 4$ 95

A-4 Errors and rates obtained with both models on a fine mesh, $p = 3$, $n = 20$ 96

List of Tables

1.1 `dof` and `nnz` expressions for each Galerkin method 16

1.2 Local connectivity tables for different element types, $l_{i,j}$ 17

1.3 Coupling factors for `nnz` calculation 18

3.1 Number of triangles (t) and quads (q) for the mesh generation examples 47

4.1 Mesh size correlation rates away from the origin for the L^2 error control applied to the corner singularity problem 54

4.2 Mesh size and aspect ratio correlation rates away from the wall for the L^2 error control applied to the $2d$ boundary layer problem 59

4.3 Properties of the 30,000 degree-of-freedom adapted meshes for the supersonic wedge case 64

4.4 Properties of the adapted meshes for the turbulent, transonic RAE2822 case, $p = 1$. L^∞-R stands for L^∞-Recombined. 79

4.5 Properties of the adapted meshes for the turbulent, transonic RAE2822 case, $p = 2$. L^∞-R stands for L^∞-Recombined. 79

B.1 Connectivity factor, $c_{i,j}$, for triangles using the EDG method 98

B.2 Connectivity factor, $c_{i,j}$, for quadrilaterals using the EDG method . . 98

B.3 Connectivity factor, $c_{i,j}$, for tetrahedra using the EDG method 98

B.4 Connectivity factor, $c_{i,j}$, for hexahedra using the EDG method 98

Chapter 1

Introduction

1.1 Motivation

The numerical wind tunnel is widely accepted as an invaluable design tool for engineers. The ability to simulate physical phenomena around complex geometries at a lower cost and time investment than experimental methods facilitates the design of next-generation aircraft. The reliability of computational methods has, however, been an active research area for the last few decades [54].

Interest in unstructured high-order methods has given rise to some general approaches for accurately simulating the physics around complex geometries at lower computational costs than existing second-order methods. In the realm of finite-element methods, these techniques branch into continuous (CG), discontinuous (DG) or hybridized (HDG, EDG) Galerkin discretisations which vary based on the nature of the function spaces housing the numerical solution as well as the number of unknowns in the resulting system of equations.

Obtaining accurate numerical solutions under available computational resources is facilitated by an adaptive solution process, in which either the mesh size (h) or polynomial space of the solution (p) are refined (coarsened) to concentrate degrees of freedom near solution features. In general, these features are not known *a priori* and

Table 1.1: dof and nnz expressions for each Galerkin method

	DG	HDG	EDG
ndof	$\mathtt{ndof}_d$	$g_{d-1,d}\mathtt{ndof}_{d-1}$	$\sum_{i=0}^{d-1} g_{i,d}\mathtt{ndof}_i^{int}$
nnz	$\mathtt{c}\,\mathtt{ndof}_d$	$\mathtt{c}\,g_{d-1,d}\mathtt{ndof}_{d-1}$	$\sum_{i=0}^{d-1} \mathtt{c}_i g_{i,d}\mathtt{ndof}_i^{int}$

error estimation techniques can be employed to localize errors in the current discretisation. In addition, element shape plays an important role in an adaptive solution process. For example, quadrilateral elements can provide better alignment with local solution features than triangular elements. This work aims at investigating the use of mixed-elements in a mesh-adaptative solution process.

The memory consumption of the DG, HDG and EDG schemes with d-simplices or d-cubes is first considered. Note the analysis for the EDG method is equivalent to that of the continuous Galerkin method with static condensation. The mesh is assumed to be very large such that the number of boundary faces is negligible compared to the number of elements. Without loss of generality, a single conservation law is used in the analysis. The degrees of freedom (ndof) and number of non-zero entries in the Jacobian of the governing equations (nnz) are counted:

$$\mathtt{ndof} \equiv \frac{\text{number of degrees of freedom}}{\text{number of elements}}, \quad \mathtt{nnz} \equiv \frac{\text{number of non-zero Jacobian entries}}{\text{number of elements}}.$$

The objective of this analysis is to compare the cost of performing a finite-element simulation within some domain $\Omega \subset \mathbb{R}^d$ tessellated by meshes composed of either d-simplices or d-cubes. The degrees of freedom can be calculated using the expressions in Table 1.1, where the quantities $\mathtt{ndof}_i$ represent the degrees of freedom of an i-dimensional entity, which varies based on element shape. The superscript ^{int} indicates only internal degrees of freedom are counted. For example, an i-dimensional cube has $\mathtt{ndof}_i = (p+1)^i$ and $\mathtt{ndof}_i^{int} = (p-1)^i$. The weights, $g_{i,d}$, represent the global number of i-dimensional elements relative to the number of elements in the entire

Table 1.2: Local connectivity tables for different element types, $l_{i,j}$

Element	Tri			Quad			Tet				Hex			
dim	0	1	2	0	1	2	0	1	2	3	0	1	2	3
0	1	6	6	1	4	4	1	12	20	20	1	6	12	8
1	2	1	2	2	1	2	2	1	36/7	36/7	2	1	4	4
2	3	3	1	4	4	1	3	3	1	2	4	4	1	2
3	-	-	-	-	-	-	4	6	4	1	8	12	6	1

mesh (d-dimensional entities). These are computed from

$$g_{i,d} = \frac{l_{d,i}}{l_{i,d}}$$

where $l_{i,d}$ represents the local number of d-dimensional entities connected to an i-dimensional one. A simple example consists of computing $g_{1,2}$ for a triangulation, which gives the number of edges in a mesh relative to the number of triangles; $g_{1,2} = 3/2$ since there are three edges per triangle and two triangles per edge. The assumed connectivity tables for the current work are given in Table 1.2, which differ from those presented in [26] since equilateral simplices are assumed here whereas Huerta et al. assume structured (right-angled) simplices. More specifically, the ball around a three-dimensional mesh vertex is assumed to be an icosahedron instead of a cube. The connectivity of equilateral and right-angled triangles used here is identical to that in [26]. Similarly, the number of non-zero entries in the Jacobian can be computed by summing the weighted contributions of each d-dimensional entity as given in Table 1.1, where the coupling factor, c_i represents the connectivity of an i-dimensional entity with other entities; this factor reduces to a single value, c, for the DG and HDG methods. For each method, the coupling factor is given in Table 1.3. Note the $c_{i,j}$ connectivity entries for the EDG method are used unchanged from [26] and are provided using the current notation in Appendix B.

Under the assumption of unit-length equilateral simplices and unit-length cubes in some reference space, the reduction in the number of cubes versus simplices required

Table 1.3: Coupling factors for nnz calculation

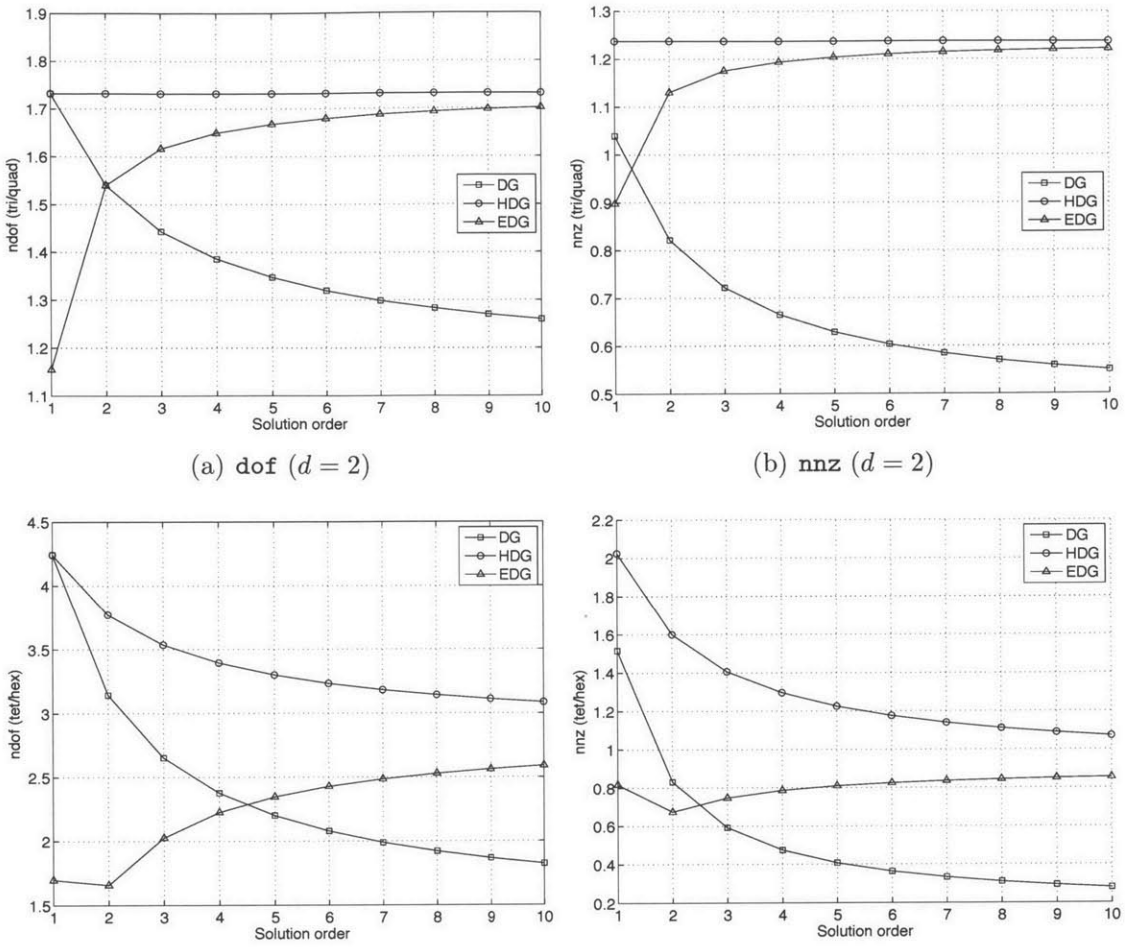
Method	Coupling factor
DG	$\mathtt{c} = (l_{d,d-1} + 1)\mathtt{ndof}_d$
HDG	$\mathtt{c} = (l_{d-1,d} l_{d,d-1} - 1)$
EDG	$\mathtt{c}_i = \sum_{j=0}^{d} c_{i,j} \mathtt{ndof}_j^{int}$

to fill Ω is determined by the volume ratio of unit d-cubes to d-simplices:

$$f \equiv \frac{n_{cubes}}{n_{simplices}} = \frac{\sqrt{d+1}}{d!\sqrt{2^d}} \leq 1. \tag{1.1}$$

The memory footprint of the above discretisations have been derived by modifying the work of Huerta et al. [26] to account for equilateral simplicial meshes. Additionally accounting for the above volume correction, the memory footprint ratios of d-simplices to d-cubes are shown in Fig. 1-1.

Figs. 1-1(a) & (c) show that 2-, 3-cubes consume less memory when storing solution variables, regardless of the discretisation method or solution order. The memory needed to store the Jacobian matrix is, in general, the primary memory consumer of an implicit solver. As shown in Fig. 1-1(b), in two dimensions, the ratio of memory consumed by the HDG method with triangles versus quadrilaterals is unaffected by the solution order. In addition, a $p = 1$ discretisation with the DG method is more costly with triangles but immediately becomes more costly with quadrilaterals for $p > 1$. On the other hand, an EDG discretisation with quadrilaterals is advantageous for $p > 1$. In three dimensions, the HDG method is always less costly with hexahedra than with tetrahedra. Similar to the case in two dimensions, hexahedra are less costly for a $p = 1$ DG discretisation but tetrahedra are more advantageous with this scheme for $p > 1$. Also observe the EDG discretisation is always less costly when using tetrahedra. Of course, these figures merely demonstrate the difference in memory consumption. Adding structure to a mesh, thereby increasing solver robustness, may trump the higher memory cost of cubes over simplices.



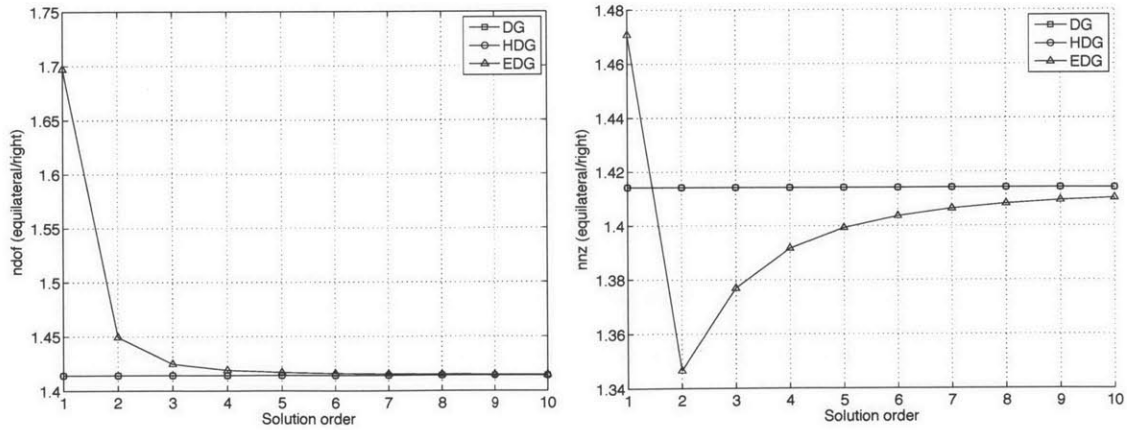
(a) **dof** ($d = 2$)

(b) **nnz** ($d = 2$)

(c) **dof** ($d = 3$)

(d) **nnz** ($d = 3$)

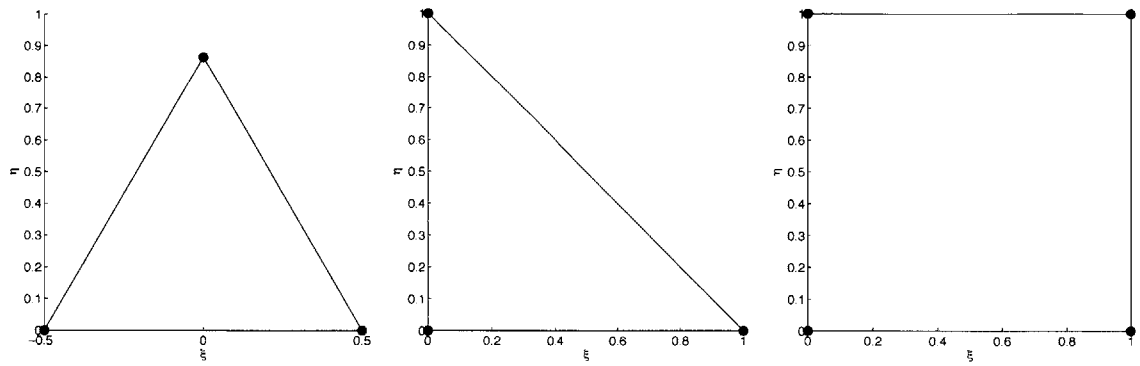
Figure 1-1: Memory footprint of d-simplices versus d-cubes for DG, HDG and EDG discretisations; a factor of 1 indicates both shapes incur the same memory cost



(a) **dof**

(b) **nnz**

Figure 1-2: Memory footprint of equilateral tetrahedra versus right tetrahedra for DG, HDG and EDG discretisations



(a) Equilateral triangle (b) Right triangle (c) Quadrilateral

Figure 1-3: Reference shapes in two dimensions

The above analysis purely compares unit d-cubes to equilateral d-simplices. It is interesting to note, however, that right triangles and right tetrahedra reduce the number of elements required to fill Ω while introducing structure to the mesh, assuming the legs of the right simplices have unit length. For reference, the assumed shapes of equilateral triangles, right triangles and quadrilaterals in a two-dimensional reference space are shown in Fig. 1-3.

Since the connectivity for both equilateral and right triangles is identical ($d = 2$), the memory cost of all discretisations with these shapes differ only by the area ratio, $2/\sqrt{3}$, which states that right triangular meshes are always less costly than equilateral ones at all solution orders. In three dimensions, the assumed connectivities for equilateral and right tetrahedral meshes are different. The cost of using equilateral tetrahedra versus right tetrahedra is shown in Fig. 1-2. Only the purely discontinuous discretisations (DG, HDG) exhibit the same behaviour observed in two dimensions. That is, the cost of using equilateral versus right tetrahedra differs only by the volume ratio, $\sqrt{2}$. With the EDG method, however, the cost ratio of storing the non-zero entries of the Jacobian increases for higher solution orders.

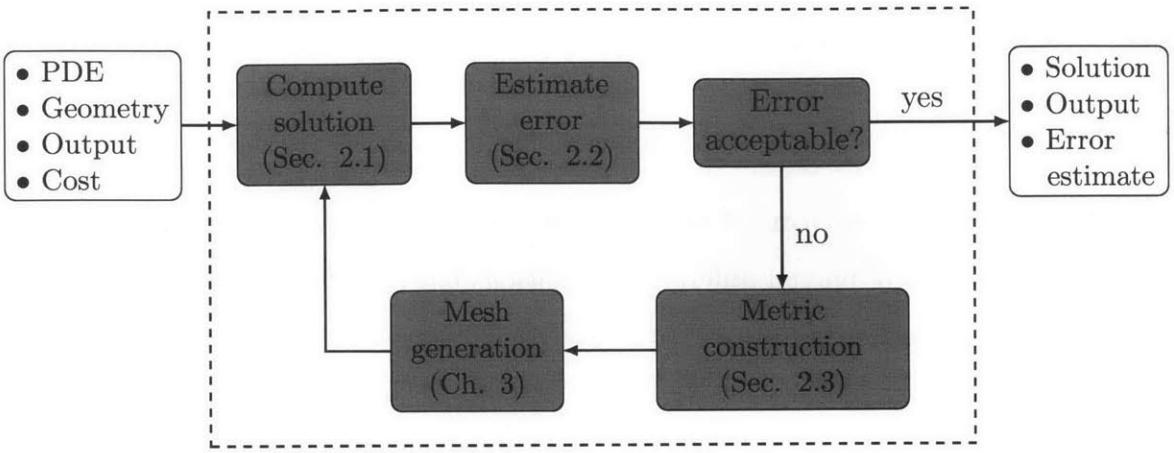


Figure 1-4: Adaptive simulation framework

1.2 Outline

Chapter 2 first presents the algorithm used to compute the mesh-adapted solution for a given computational cost. Next, Chapter 3 develops an anisotropic mixed-element mesh generation algorithm. Numerical experiments in Chapter 4 both verify and demonstrate, through applications drawn from computational fluid dynamics, the effectiveness of the developed algorithm. Some conclusions and recommendations for future work are made in Chapter 5.

For reference, the adaptive simulation framework used in this work is shown in Fig. 1-4. In general, the algorithm takes as input: a description of the domain geometry, an output of interest, a set of governing equations and a measure of the desired computational cost prescribed by the user. The solver iterates towards a solution by adapting the mesh to reduce the error in the output of interest. As such, discretisation, error estimation and mesh generation techniques will be discussed. This work contributes to the *Metric construction* and *Mesh generation* blocks.

1.3 Background

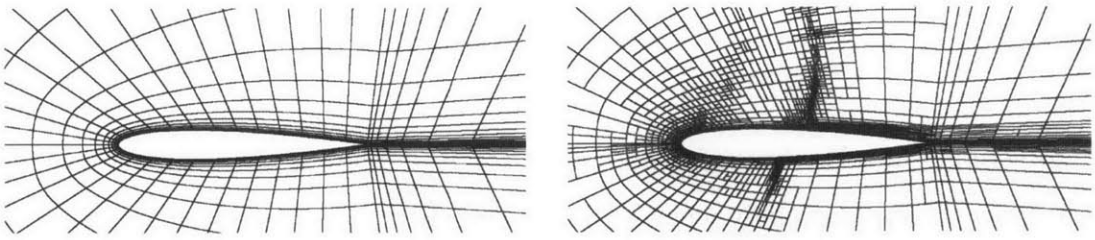
1.3.1 Error estimation and adaptation

Simplex-based adaptation The importance of mesh adaptation for solving high-order discretisations of partial differential equations has been noted in [61]; in particular, anisotropic mesh adaptation has received a considerable amount of interest in the past several years [33, 35, 36, 37, 38, 39, 59, 60]. Notably, Loseille and Alauzet introduce the *continuous mesh framework* (Sec. 2.3.1) in which they reduce the discrete mesh optimization problem to that of a continuous metric optimization. They compute a requested metric directly by minimizing the continuous interpolation error [34]. The current work utilizes this duality between a metric and a mesh with applications to high-order discretisations. Here, the cost-constrained error minimization problem is solved via numerical techniques as in [58, 60]. While previous work has focused solely on simplicial mesh adaptation, this work extends these adaptation techniques to mixed-element meshes.

Quadrilateral subdivision Hartmann and Houston [20, 21, 22, 23] study output-based error estimation techniques along with adaptive refinement on quadrilateral meshes. Similarly, Ceze and Fidkowski [10] build upon this work in which they perform x, y or xy refinemenent in $2d$ to drive their adaptive process. It is important to note that their solver requires the treatment of hanging nodes, as shown in Fig. 1-5(b). In addition, Georgoulis et al. [17, 18] study hp-refinement strategies for quadrilateral elements for solving second-order partial differential equations with discontinuous Galerkin methods. They make use of the dual-weighted residual error estimate to drive h or p, isotropic or anisotropic refinement.

1.3.2 Mesh generation

Delaunay-based mesh generation Delaunay-based methods are popular for generating isotropic meshes. These methods have been implemented in softwares such as `Triangle` [52] in two dimensions and `TetGen` [53] in three dimensions. An anisotropic



(a) Initial (b) Adapted

Figure 1-5: Initial and adapted mesh obtained using the quadrilateral subdivision technique, reproduced with permission from Marco Ceze and Krzysztof J. Fidkowski [10]

extension of the empty circumellipse property is used in the highly successful `bamg` (Bidimensional Anisotropic Mesh Generator) [24]. Similarly, Dobrzynski and Frey have extended the Delaunay kernel to generate anisotropic tetrahedral meshes [13].

These techniques make use of mesh modification operators, such as face swap, vertex insertion or edge collapse. The extension of these operations to cubes is challenging since these operations propagate further into neighbouring elements than simplices. The analog of a circumellipse from a triangle to a quadrilateral is also poorly defined.

Advancing front mesh generation Advancing front methods [1, 32, 43, 44, 46, 50] are advantageous because they generally align elements with domain boundaries, an attractive property when intricate mesh grading is required near geometry surfaces such as in a boundary layer. Marcum and Alauzet [1] recently presented a method for generating boundary layer meshes and creating a metric-conforming mesh in the remaining volume.

Remacle [50] uses the L^∞ norm to generate directional, isotropic right-triangle meshes suitable for recombination into quadrilaterals. While their method appears to be fast and robust, the extension of the empty circumcircle property in the L^∞-norm is not straightforward in the anisotropic case, let alone the extension to three

dimensions.

One striking disadvantage of the advancing front method is the arbitrary nature in which fronts collide, especially when advancing quadrilaterals directly. In the isotropic case in two dimensions, a number of heuristic smoothing and repairing techniques can be applied to repair front collisions [6, 43]. As fronts collide in the anisotropic case, metric conformity becomes an issue. Also note that mesh features may not necessarily lie near domain boundaries, such as in the case of an oblique shock wave. In these cases, fronts should ideally be setup near these metric features; a possible solution would be to use feature detection algorithms, however, this is a separate research topic and may not even extend to three dimensions.

Variational approaches Variational approaches to mesh generation consist of posing the discrete mesh generation problem as a continuous optimization problem in which some energy is minimized to compute vertex locations and mesh topology. In [30], the authors minimize the approximation error of surface quadrangulations. Also, Lévy [40] introduces an approach, `Vorpaline`, for generating anisotropic surface meshes. The idea is to embed both the domain $\in \mathbb{R}^3$ and metric $\in \mathbb{R}^3$ into a surface in $\mathbb{R}^6$. The six-dimensional isotropic Voronoi diagram is then computed, constrained and projected back into $\mathbb{R}^3$, giving the metric-conforming triangulation of the original surface.

They also discuss methods for generating quadrilateral and hexahedral meshes by defining the energy of a mesh in some L^p norm and optimizing this energy for the coordinates of the constrained Voronoi tessellation [41]. They note this method has converged well for anisotropic ratios of 100:1; here, anisotropic ratios on the order of 10^6:1 are sought.

Primitive operations A simple method for generating metric-conforming meshes was first introduced by Bossen and Heckbert [9]. The method makes use of simple

mesh operations such as node insertion, collapse, smoothing and edge swapping to match a requested sizing and orientation field. A similar method is used by [15] where the authors use an L^∞ norm to measure edge lengths in the reference space. Due to their simplicity, these algorithms extend well to three dimensions, as demonstrated by the `EPIC` (Edge Primitive Insertion and Collapse) code produced by The Boeing Company [42]. Loseille [38] uses a similar approach to generate three-dimensional anisotropic meshes and also discusses the use of a boundary layer metric to add mesh structure near surfaces for viscous simulations.

Chapter 2

Discretization, error estimation and adaptation

This chapter presents the adaptation algorithm used in this work, from the discretisation scheme to metric construction.

2.1 Numerical discretization

A general system of conservation laws within some domain $\Omega \in \mathbb{R}^d$ is given by

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot [\mathcal{F}_c(\mathbf{u}, \mathbf{x}, t) - \mathcal{F}_d(\mathbf{u}, \nabla \mathbf{u}, \mathbf{x}, t)] = \mathcal{S}(\mathbf{u}, \nabla \mathbf{u}, \mathbf{x}, t), \quad \forall\, \mathbf{x} \in \Omega \tag{2.1}$$

with boundary conditions

$$\mathcal{B}(\mathbf{u}, \mathcal{F}_d(\mathbf{u}, \nabla \mathbf{u}, \mathbf{x}, t) \cdot \mathbf{n}; \mathrm{bc}) = 0, \qquad \forall\, \mathbf{x} \in \partial\Omega. \tag{2.2}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the state vector with m components, $\mathcal{F}_c$ denotes the convective flux, $\mathcal{F}_d$ is the diffusive flux, $\mathcal{S}$ is a source and $\mathcal{B}$ is the boundary condition operator.

First, Ω is tessellated into non-overlapping elements, $\mathcal{T}_h$, with characteristic size h. Eqs. (2.1) with (2.2) are then solved by introducing a discontinuous finite element

space:

$$\mathcal{V}_{h,p} = \left\{ \mathbf{v}_{h,p} \in \left[L^2(\mathcal{T}_h) \right]^m : \mathbf{v}_{h,p} \circ \phi_q(\kappa) \in \left[\mathcal{P}^p(K) \right]^m, \forall \kappa \in \mathcal{T}_h \right\}, \tag{2.3}$$

where $\phi_q(\kappa)$ is the q-th order diffeomorphic mapping from physical element κ to master element K and $\mathcal{P}^p(K)$ denotes the complete p-th order solution space on K.

The weak form of the governing equations are obtained by restricting $\mathbf{u}$ to $\mathcal{V}_{h,p}$, multiplying Eq. (2.1) by the test function $\mathbf{v}_{h,p}$ and integrating by parts over each element, yielding

$$\sum_{\kappa \in \mathcal{T}_h} \left[\int_\kappa \mathbf{v}_{h,p}^t \frac{\partial \mathbf{u}_{h,p}}{\partial t} + \mathcal{R}_{h,p}(\mathbf{u}_{h,p}, \mathbf{v}_{h,p}) \right] = 0, \quad \forall\, \mathbf{v}_{h,p} \in \mathcal{V}_{h,p}, \tag{2.4}$$

where the residual operator, $\mathcal{R}_{h,p}(\cdot,\cdot) : \mathcal{V}_{h,p} \times \mathcal{V}_{h,p} \to \mathbb{R}$ can be broken into the convective, diffusive and source discretizations:

$$\mathcal{R}_{h,p}(\mathbf{u}_{h,p}, \mathbf{v}_{h,p}) = \mathcal{R}^c_{h,p}(\mathbf{u}_{h,p}, \mathbf{v}_{h,p}) + \mathcal{R}^d_{h,p}(\mathbf{u}_{h,p}, \mathbf{v}_{h,p}) + \mathcal{R}^s_{h,p}(\mathbf{u}_{h,p}, \mathbf{v}_{h,p}). \tag{2.5}$$

Roe's approximate Riemann solver is used to discretize the convective flux, the second form of Bassi and Rebay for the diffusive flux and the asymptotically dual-consistent form of Bassi *et al.* for the source term. Boundary conditions are weakly imposed by prescribing the fluxes on faces that lie on $\partial\Omega$. For further details of the discretisations used in this work the reader is referred to [56, 58].

2.2 Output error estimation

As highlighted in Chapter 1 the adaptation algorithm targets a specified output of interest. In the most general setting, this output is computed by some functional $\mathcal{J}(\cdot) : \mathcal{V} \to \mathbb{R}$. Denote $J = \mathcal{J}(\mathbf{u})$ as the exact output and $J_{h,p} = \mathcal{J}_{h,p}(\mathbf{u}_{h,p})$ as the approximated one; the error introduced by the discretisation is

$$\mathcal{E}_{\text{true}} \equiv J - J_{h,p}. \tag{2.6}$$

Since the exact output is not known, the dual-weighted residual (DWR) method is used to estimate Eq. (2.6). This method, proposed by Becker and Rannacher [5], weights the residual operator by the adjoint solution

$$\mathcal{E}_{\text{true}} = \mathcal{R}_{h,p}(\mathbf{u}_{h,p}, \boldsymbol{\psi}) \tag{2.7}$$

where $\boldsymbol{\psi} \in \mathcal{W} \equiv \mathcal{V} + \mathcal{V}_{h,p}$ is the exact solution to the dual problem,

$$\overline{\mathcal{R}}'_{h,p}[\mathbf{u}, \mathbf{u}_{h,p}](\mathbf{w}, \boldsymbol{\psi}) = -\overline{\mathcal{J}}'_{h,p}[\mathbf{u}, \mathbf{u}_{h,p}](\mathbf{w}), \quad \forall\, \mathbf{w} \in \mathcal{W} \tag{2.8}$$

where $\overline{\mathcal{R}}'_{h,p}[\mathbf{u}, \mathbf{u}_{h,p}] : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$ and $\overline{\mathcal{J}}'_{h,p}[\mathbf{u}, \mathbf{u}_{h,p}] : \mathcal{W} \to \mathbb{R}$ are the mean value linearizations of the residual operator and output functional, respectively, defined by

$$\overline{\mathcal{R}}'_{h,p}[\mathbf{u}, \mathbf{u}_{h,p}](\mathbf{w}, \mathbf{v}) \equiv \int_0^1 \mathcal{R}'_{h,p}[\theta\mathbf{u} + (1-\theta)\mathbf{u}_{h,p}](\mathbf{w}, \mathbf{v})\, \mathrm{d}\theta, \tag{2.9}$$

$$\overline{\mathcal{J}}'_{h,p}[\mathbf{u}, \mathbf{u}_{h,p}](\mathbf{w}) \equiv \int_0^1 \mathcal{J}'_{h,p}[\theta\mathbf{u} + (1-\theta)\mathbf{u}_{h,p}](\mathbf{w})\, \mathrm{d}\theta \tag{2.10}$$

where $\mathcal{R}'_{h,p}[z](\cdot,\cdot)$ and $\mathcal{J}'_{h,p}[z](\cdot)$ denote the Fréchet derivative of $\mathcal{R}_{h,p}(\cdot,\cdot)$ and $\mathcal{J}_{h,p}(\cdot)$ with respect to the first argument evaluated about the state z.

Since the exact adjoint solution is not computable, an approximate solution, $\boldsymbol{\psi}_{h,\hat{p}} \in \mathcal{V}_{h,\hat{p}}$, satisfying

$$\overline{\mathcal{R}}'_{h,p}[\mathbf{u}_{h,p}](\mathbf{v}_{h,\hat{p}}, \boldsymbol{\psi}_{h,\hat{p}}) = -\overline{\mathcal{J}}'_{h,p}[\mathbf{u}_{h,p}](\mathbf{v}_{h,\hat{p}}), \quad \forall\, \mathbf{v}_{h,\hat{p}} \in \mathcal{V}_{h,\hat{p}} \tag{2.11}$$

is obtained where $\mathcal{T}_h$ is fixed but the solution space of the adjoint equation is enriched to $\hat{p} = p + 1$; note this is necessary to ensure the enriched residual operator acting on a prolongated version of $\mathbf{u}_{h,p}$ to $\mathcal{V}_{h,\hat{p}}$ does not vanish. The DWR estimate employed in this work thus takes the form

$$\mathcal{E}_{\text{true}} \approx \mathcal{R}_{h,p}(\mathbf{u}_{h,p}, \boldsymbol{\psi}_{h,\hat{p}}). \tag{2.12}$$

This error can be localized over each element κ by restricting the evaluation of Eq. (2.12) to each element,

$$\eta_\kappa \equiv \left|\mathcal{R}_{h,p}(\mathbf{u}_{h,p}, \boldsymbol{\psi}_{h,\hat{p}}|_\kappa)\right| . \tag{2.13}$$

This gives an indication of the local elemental error and is useful in driving an adaptive process.

2.3 Metric construction

Now consider the problem of finding the optimal metric field to pass to the mesh generator of Chapter 3. The simplicial framework of Yano and Darmofal [58, 60], Mesh Optimization via Error Sampling and Synthesis (MOESS), is extended here to handle mixed-element meshes. First, the duality between a metric field and a mesh is discussed.

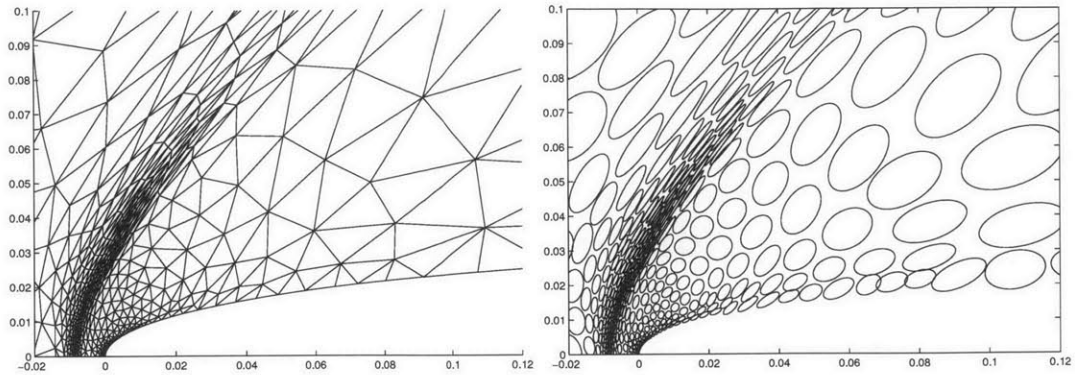
2.3.1 Mesh-metric duality

In this work, element sizes and orientations are encoded into a Riemannian metric field which is a continuously-defined set of symmetric positive-definite (SPD) tensors defined over Ω. A metric field simply transforms the notions of length and area, among other geometric properties.

Definition 2.1. (k-norm length of a vector under a metric field). *The k-norm length of a vector $\mathbf{e} \in \mathbb{R}^d$ in physical space under a continuous metric field, $\{\mathcal{M}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$, is given by*

$$\ell_{\mathcal{M}}(\mathbf{e}; k) = \int_0^1 \left(\sum_{i=1}^{d} \left[\sqrt{\lambda_i} \mathbf{q}_i^t \mathbf{e} \right]^k \right)^{1/k} \mathrm{d}s. \tag{2.14}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ and $\mathbf{q} \in \mathbb{R}^{d\times d}$ are the eigenvalues and eigenvectors of $\mathcal{M}(\mathbf{e}_0 + \mathbf{e}s)$, respectively. The value of k controls the *interpretation* of length under a metric field and will be discussed in a later chapter. For now, assume $k = 2$ unless stated otherwise.



(a) Mesh (b) Metric

Figure 2-1: Mesh-metric duality

Definition 2.2. (volume under a metric field). *The volume of a region in physical space, $\kappa \subset \mathbb{R}^d$, under a continuous metric field, $\{\mathcal{M}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$, is given by*

$$v_{\mathcal{M}}(\kappa) = \int_{\kappa} \sqrt{\det(\mathcal{M}(\mathbf{x}))}\, \mathrm{d}\mathbf{x}. \tag{2.15}$$

The above definitions alter the interpretation of edge lengths and element volumes with respect to a metric field, as shown in Fig. 2-1.

Metric-conforming mesh A tessellation, $\mathcal{T}_h$ is said to *conform* to a smoothly varying Riemannian metric field, $\{\mathcal{M}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ if $\mathcal{T}_h$ satisfies both an edge length condition [7, 36, 37],

$$\ell_{\min} \leq \ell_{\mathcal{M}}(\mathbf{e}) \leq \ell_{\max}, \quad \forall\, \mathbf{e} \in \mathrm{edges}(\mathcal{T}_h) \tag{2.16}$$

and a quality condition,

$$1 \leq q(\kappa) \leq q_{\max}, \quad \forall\, \kappa \in \mathcal{T}_h \tag{2.17}$$

where the quality measure is defined in this work as

$$q(\kappa) = \frac{1}{c_K} \frac{\sum_{\mathbf{e}\in edges(\kappa)} \ell^2_{\mathcal{M}}(\mathbf{e})}{(v_{\mathcal{M}}(\kappa))^{2/d}}, \tag{2.18}$$

where c_K is a constant used to assign a unit quality to the ideal reference element. For example, c_K for a unit-length equilateral triangle is $4\sqrt{3}$ whereas that for a unit square is 4.

Mesh-implied metric field Given a tessellation $\mathcal{T}_h$, the implied metric field of $\mathcal{T}_h$ is defined in two steps. First, a discontinuous metric field is computed by evaluating the implied metric of each element,

$$\mathcal{M}_\kappa(\mathbf{x}) = \frac{\int_\kappa \left([\mathbf{D}\phi_q(\mathbf{y})]^T[\mathbf{D}\phi_q(\mathbf{y})]\right)^{-1} \mathrm{d}\mathbf{y}}{\int_\kappa \mathrm{d}\mathbf{y}}, \quad \forall\, \mathbf{x} \in \kappa \tag{2.19}$$

where $\mathbf{D}\phi_q(\mathbf{y})$ is the Jacobian of the transformation from the master element to the curved element which is constant for straight-sided simplices. A continuous metric field is then constructed by averaging the metrics surrounding each mesh vertex. In the affine-invariant framework of [45], this average is computed as

$$\mathcal{M}_\nu = \underset{\mathcal{M}}{\operatorname{argmin}} \sum_{\kappa \in \diamondsuit(\nu)} \|\log\left(\mathcal{M}_\kappa^{-1/2}\mathcal{M}\mathcal{M}_\kappa^{-1/2}\right)\|_F^2 \tag{2.20}$$

where $\diamondsuit(\nu)$ denotes the *star* of vertex ν; that is, the collection of elements attached to ν. The metric tensor at $\mathbf{x}$ can be computed by first locating the element containing $\mathbf{x}$ and computing the weighted mean of the vertex metrics. In the affine-invariant framework, this is

$$\mathcal{M}(\mathbf{x}) = \underset{\mathcal{M}}{\operatorname{argmin}} \sum_{\nu \in \mathcal{V}(\kappa)} w_\nu(\mathbf{x}) \|\log\left(\mathcal{M}_\nu^{-1/2}\mathcal{M}\mathcal{M}_\nu^{-1/2}\right)\|_F^2 \tag{2.21}$$

where $\mathcal{V}(\kappa)$ is the set of vertices on element κ and $w_\nu(\mathbf{x})$ denotes the barycentric coordinate of $\mathbf{x}$ with respect to ν. The interested reader is referred to the gradient-descent algorithm of [45] for the evaluation of Eqs. (2.20) and (2.21).

2.3.2 Mesh optimization via error sampling and synthesis

The steps needed to construct a continuous metric field during each adaptation iteration are now presented. This algorithm is henceforth referred to as Mesh Optimization via Error Sampling and Synthesis (MOESS). The method was first introduced by Yano and Darmofal [58, 60] and in this work is extended to treat mixed-element meshes. The current analysis is limited to h-adaptation; the algorithm seeks the optimal tessellation, $\mathcal{T}_h^*$ of Ω, given some solution order, p, which minimizes the discretization error subject to a requested computational cost,

$$\mathcal{T}_h^* = \underset{\mathcal{T}_h}{\operatorname{arginf}}\ \mathcal{E}(\mathcal{T}_h), \quad \text{s.t.} \quad \mathcal{C}(\mathcal{T}_h) \leq \mathtt{N}, \tag{2.22}$$

where $\mathcal{E}(\cdot)$ and $\mathcal{C}(\cdot)$ denote the error and cost functionals, respectively. Since simultaneously optimizing the topology and vertex positions of $\mathcal{T}_h$ is, in general, intractable, the mesh-metric duality of Sec. 2.3.1 is used to apply a continuous relaxation of the optimization problem, as proposed by Loseille and Alauzet [36, 37]. As a result, the continuous metric field $\boldsymbol{\mathcal{M}} = \{\mathcal{M}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ is optimized and a metric-conforming tessellation is constructed:

$$\boldsymbol{\mathcal{M}}^* = \underset{\boldsymbol{\mathcal{M}}}{\operatorname{argmin}}\ \mathcal{E}(\boldsymbol{\mathcal{M}}) \quad \text{s.t.} \quad \mathcal{C}(\boldsymbol{\mathcal{M}}) \leq \mathtt{N}. \tag{2.23}$$

In this view, the error and cost functionals can be evaluated as

$$\mathcal{E}(\boldsymbol{\mathcal{M}}) = \sum_{\kappa\in\mathcal{T}_h} \eta_\kappa \tag{2.24}$$

and

$$\mathcal{C}(\boldsymbol{\mathcal{M}}) = \int_\Omega c_p \sqrt{\det(\mathcal{M}(\mathbf{x}))}\,\mathrm{d}\mathbf{x} = \sum_{\kappa\in\mathcal{T}_h} \int_\kappa c_{p,\kappa} \sqrt{\det \mathcal{M}(\mathbf{x})}\,\mathrm{d}\mathbf{x} \tag{2.25}$$

where η_κ is the local error indicator from Eq. (2.13). Note the constant $c_{p,\kappa}$ is dependent on the solution order of κ normalized by the size of the reference element, $\hat{\kappa}$.

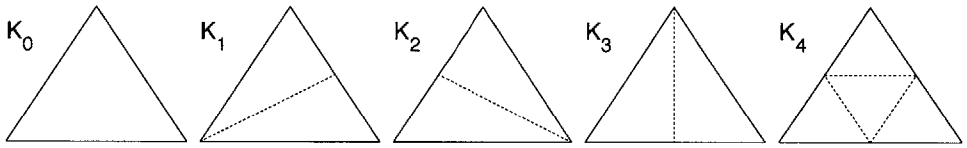
After integration, the cost functional reduces to [58]:

$$\mathcal{C}(\mathcal{M}) = \sum_{\kappa \in \mathcal{T}_h} \rho_\kappa \tag{2.26}$$

where $\rho_\kappa = \text{dof}(\hat{\kappa})$ is the degrees of freedom of the reference element, $\hat{\kappa}$. Since high cube-to-simplex ratio tessellations are sought, ρ_κ is set to $(p+1)^d$ in mixed-element meshes, regardless of the element shape.

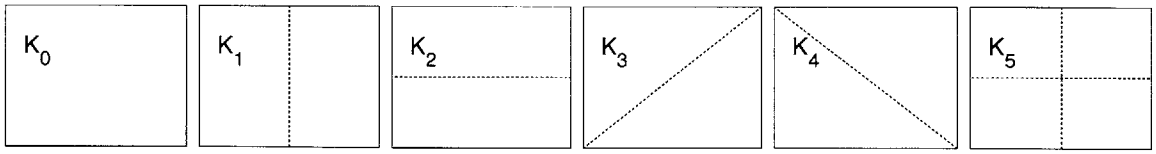
The MOESS algorithm consists of three essential stages. In Stage I, local elemental subdivisions gather metric-error pairs to be used in the construction of a metric-error model (Stage II). Stage III uses this model to optimize $\mathcal{M}$ which will be subsequently passed to the mesh generator of Chapter 3.

Stage I: Error sampling Eq. (2.13) gives a measure of the local error over an element $\kappa_0 \in \mathcal{T}_h$ due to the discretisation. Each element, κ_0, can be further split and the error over each child element can be recalculated. The split configurations used in this work are shown in Figs. 2-2 and 2-3.



(a) Original (b) Split 1 (c) Split 2 (d) Split 3 (e) Uniform

Figure 2-2: Triangle split configurations



(a) Original (b) Split 1 (c) Split 2 (d) Split 3 (e) Split 4 (f) Uniform

Figure 2-3: Quadrilateral split configurations

The implied metric associated with each split configuration is taken as the affine-invariant mean, Eq. (2.20), of the implied metrics of the child elements and the error

introduced by the child elements is summed to complete the set of metric-error pairs:

$$\{\mathcal{M}_i, \eta_{\kappa_i}\}, \quad i = 1, \ldots, n_s \tag{2.27}$$

where n_s denotes the number of samples used on κ_0.

The implied metric of each split element along with its recomputed error is used to build a list of metric-error samples. In the case of quadrilaterals, all five split configurations are used.

Stage II: Error model synthesis The metric-error pairs are then used to construct a model of the error as a function of the metric field. This model is constructed in terms of the *step tensor*, $\mathcal{S}$, which represents the vector difference between two metric tensors. In the context of the discrete set of sampled data, these tensors are computed from $\mathcal{M}_{\kappa_i}$ and $\mathcal{M}_{\kappa_0}$ as

$$\mathcal{S}_{\kappa_i} = \log\left(\mathcal{M}_{\kappa_0}^{-1/2}\mathcal{M}_{\kappa_i}\mathcal{M}_{\kappa_0}^{-1/2}\right), \quad i = 1, \ldots, n_s. \tag{2.28}$$

A linear model is then constructed from the n_s metric-error samples,

$$f_\kappa(\mathcal{S}) \equiv \log\left(\eta(\mathcal{S})/\eta_0\right) = \mathcal{R}_\kappa : \mathcal{S}, \tag{2.29}$$

where $\mathcal{R}_\kappa$ is referred to as the *rate tensor*. A linear regression of the sampled data is used to determine $\mathcal{R}_\kappa$. Note the importance of the diagonal splits of Fig. 2-3. For the case of parallelograms, the step tensor implied by the uniform refinement configuration is a linear combination of those implied by the two anisotropic splits, thus causing the regression to be underdetermined. The diagonal samples encode additional information into the error model synthesis and prevent such a failure.

Stage III: Metric optimization Equipped with an error model over each element, the continuous optimization problem for the optimal step tensor field can be solved

which yields the requested metric at each element vertex. Recent work by [28] has investigated the use of gradient-based techniques to solve Eq. (2.23). The original formulation is supplemented with constraints to limit the change in the mesh during each adaptation. Denoting the field of step tensors as $\boldsymbol{\mathcal{S}}$, the problem formulation is stated as follows:

$$\boldsymbol{\mathcal{S}}^* = \underset{\boldsymbol{\mathcal{S}}}{\operatorname{arginf}}\ \mathcal{E}(\boldsymbol{\mathcal{S}}) + \mathbf{p}^t \phi\left(\|\mathcal{S}_\nu\|_F^2 - 4\log^2 r\right),$$

subject to:

$$\begin{aligned} \mathcal{C}(\boldsymbol{\mathcal{S}}) &\leq \mathtt{N}, \\ |(\mathcal{S}_\nu)_{i,j}| &\leq 2\log r \quad i,j = 1,\ldots,d, \quad \forall\, \nu \in \mathcal{T}_h, \\ \sum_{\nu \in \mathcal{T}_h} \|\mathcal{S}_\nu\|_F^2 &\leq \alpha \times n_\nu \times 4\log^2 r, \end{aligned} \tag{2.30}$$

where the penalty term is defined as $\phi(z) = (z < 0)\,?\,0 : z^2$ and n_ν is the number of vertices in $\mathcal{T}_h$. The penalty vector, $\mathbf{p}$ is used to control edge length deviations and the user-specified refinement factor, r, is used to control the change in the mesh from one adaptation to the next. The optimization problem is solved as follows. First, $\mathbf{p}$ is initialized to some constant c and the initial problem is solved using the Method of Moving Asymptotes in the optimization library, `NLopt` [27]. For the resultant step tensors which do not satisfy the edge length condition, the corresponding entry in $\mathbf{p}$ is multiplied by some constant factor, β, to further penalize that step tensor. In the current work, the constants are set to $\alpha = .25$, $\beta = 2$ and $c = 10^{-5}$.

Chapter 3

Mesh generation

This chapter presents the algorithm used to generate metric-conforming, mixed-element meshes. Curvilinear mesh generation is also discussed.

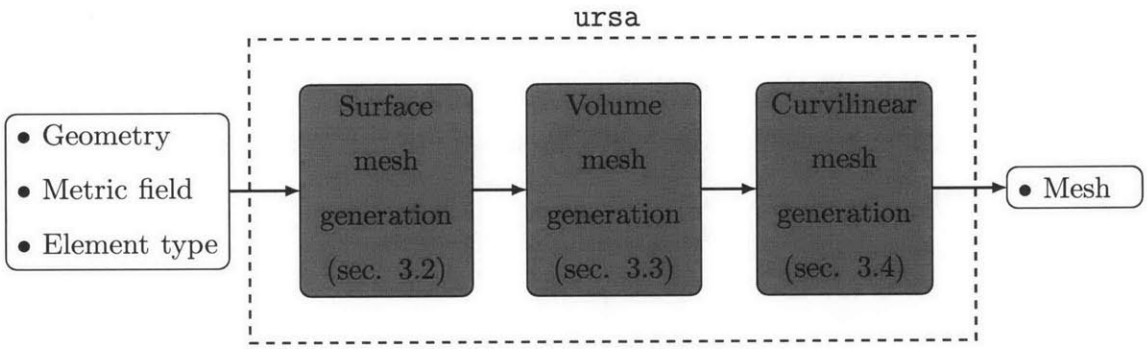


Figure 3-1: Mesh generation algorithm

3.1 Fundamentals

A basic outline of the mesh generation algorithm used in this work is shown in Fig. 3-1. The output mesh is generated from a given geometry and metric field in a dimension-by-dimension approach. The problem of generating metric-conforming surface meshes is first treated and followed by a volume tessellation algorithm. The straight-sided mesh is then curved, as required by the high-order finite-element solver. The developed mesh generation package will be henceforth referred to as `ursa` (unstructured

anisotropic). Some central ideas used throughout `ursa` are first discussed.

3.1.1 Geometry representation

The input geometry takes the form of a Non-Uniform Rational B-Spline (NURBS). Several geometric algorithms including evaluation, differentiation, projection and interpolation were drawn from *The $\mathcal{NURBS}$ Book* [48]. The basic form of a d-dimensional NURBS curve is

$$\mathbf{x}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \quad a \leq u \leq b. \tag{3.1}$$

where $\mathbf{P} \in \mathbb{R}^n \times \mathbb{R}^d$ is the vector of control points, $\mathbf{w} \in \mathbb{R}^n$ is the vector of weights and $\mathbf{N}_p \in \mathbb{R}^n$ is the p-th degree B-Spline basis functions, defined over some interval in the knot vector, $\tilde{\mathbf{u}}$. In the current implementation, a set of points along the true geometry is interpolated with cubic NURBS curves. In this setting, the knot vector is computed using the chord length method of [48] and the weights are set to 1, recovering cubic B-Splines.

3.1.2 Metric field evaluation

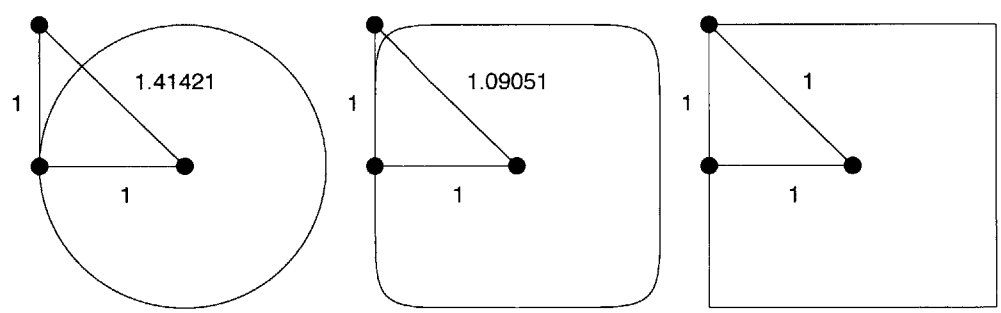
The input metric field can be described either analytically or, more practically, through the use of a background tessellation. Upon initial import, the input tessellation is split, such that no quadrilateral elements remain in the background mesh. This has little effect on the evaluation of $\mathcal{M}(\mathbf{x})$ since these are defined at the nodes of the background mesh. Evaluation of the metric tensor at $\mathbf{x}$ is done by first locating the element containing $\mathbf{x}$, computing the barycentric weights with respect to the three triangle vertices and interpolating the nodal metrics, as in Eq. (2.21). An efficiency gain is made by storing the background enclosing shape for each active node during the mesh generation algorithm. This reduces the triangle search time to approximately $\mathcal{O}(1)$ as the algorithm converges.

3.1.3 Right triangle mesh generation

The proposed algorithm for generating mixed-element tessellations first consists in creating a right-triangle mesh and further combining pairs of right triangles into quadrilaterals, when possible.

L^k norm

Traditional mesh generation algorithms, such as [24] and [42] aim for unit equilateral simplices. In particular Edge Primitive Insertion and Collapse (`EPIC`) [42] makes use of simple mesh operations, driven by edge length and element quality calculations, to generate a mesh.



(a) L^2 (b) L^8 (c) L^∞

Figure 3-2: Interpretation of a right triangle with respect to a unit L^k ball

The L^k norm modifies how lengths are interpreted in the reference metric space. Equilateral simplex mesh generation algorithms should be thought of as L^2-based algorithms. Here, the L^∞ norm is proposed to create right triangles; this idea was previously investigated by [41, 50, 15]. The length of an edge under a d-dimensional metric field can be computed in a k-norm from Eq. (2.14). Here, the two metric tensors defined at the end nodes of an edge are averaged in the affine-invariant sense to reduce the integration in Eq. (2.14) to a single evaluation. All three edge lengths of a right triangle approach unity as k approaches infinity, as depicted in Fig. 3-2. It should be noted that the quality of an element under $\mathcal{M}$ given by Eq. (2.18) is only affected by the constant c_K, which now takes the value of 6 for right triangles.

Recombination algorithms

Once a right triangle mesh is generated, pairs of triangles can be combined to form quadrilateral elements. Previous work includes the simple merging procedure of [8] or graph-matching algorithms; the latter is adopted as was done in [19, 50] and make use of the *Blossom IV* algorithm [11, 12]. The underlying principle of the recombination algorithm is that a triangulation is simply a graph, where each triangle is a graph node and each triangle-neighbour pair is an edge in the graph, as shown in Fig. 3-3.

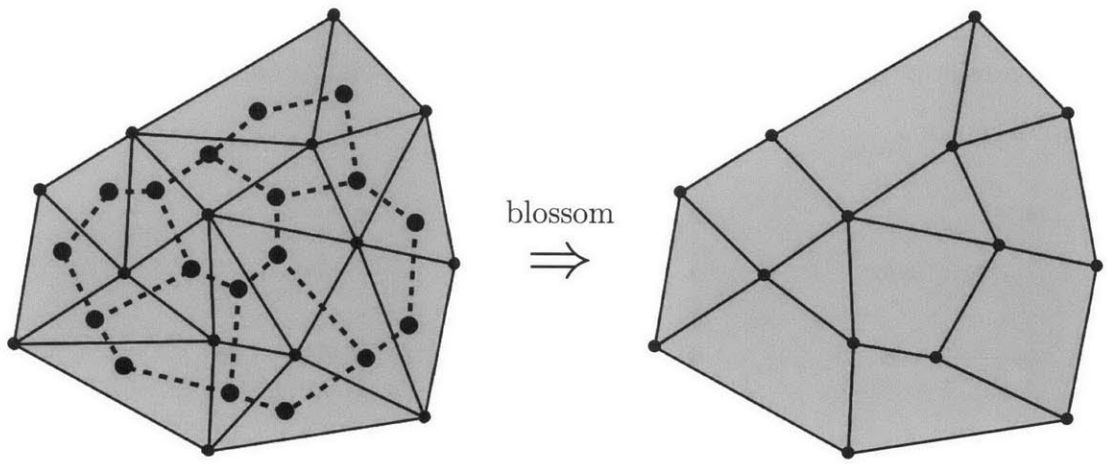


Figure 3-3: Graph-matching approach

The *Blossom IV* algorithm computes a minimum-weight perfect matching of an input graph and allows the assignment of weights, $\boldsymbol{\omega}$, to the triangle-neighbour pairs. Here, the weights are assigned as

$$\omega(\kappa_1, \kappa_2) = \sigma(\{\det \mathbf{D}\phi_{1,i}\}) \tag{3.2}$$

where κ_1 and κ_2 are the proposed pair of triangles, σ is the standard-deviation function and $\mathbf{D}\phi_{1,i}$ represents the Jacobian of the bilinear shape functions ($q = 1$) at some i-th sample point. Here, the coordinates of a tenth-order quadrature rule is used to sample the Jacobian of the proposed quadrilateral. This weighting function has been chosen to penalize elements with significantly-varying Jacobians. That is, parallelograms are sought because they have constant Jacobians and thus, constant implied metric tensor

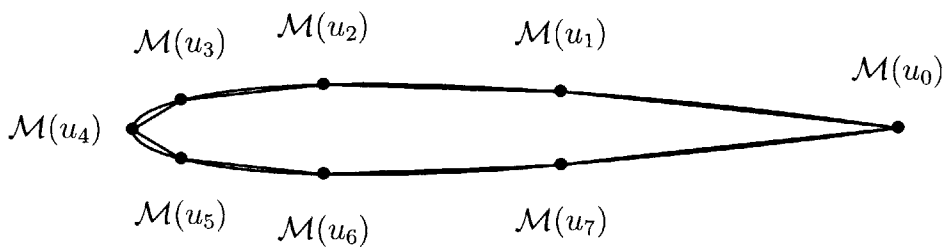


Figure 3-4: Curve metric description with true geometry

which is believed to enhance the fidelity of the error modeling stage of Sec. 2.3.2.

3.2 Surface mesh generation

The first step in generating a metric-conforming mesh is to tessellate the input geometry in forming $\mathcal{G}_h = \partial\mathcal{T}_h$. The background mesh provides nodal metrics along the domain boundaries, which are used to construct a one-dimensional background mesh in the parameter space of the corresponding NURBS curve, see Fig. 3-4.

The algorithm begins by integrating the metric along the geometry,

$$\ell_{\mathcal{M}}(\Gamma) = \sum_{i=1}^{n_e} \ell_{\mathcal{M}}(\gamma_i) = \sum_{i=1}^{n_e} \int_{u_{i-1}}^{u_i} \sqrt{\mathbf{e}_i^t \mathcal{M}(u) \mathbf{e}_i}\, \mathrm{d}u. \tag{3.3}$$

Note the use of the L^2 norm in this computation which aligns quadrilateral edges with the geometry instead of its diagonals. The number of edges needed to discretise the geometry is then $n_e' = \texttt{round}(\ell_{\mathcal{M}}(\Gamma))$. Each inserted edge should then have a length of

$$\ell_{\mathcal{M}}'(\gamma) = \frac{\ell_{\mathcal{M}}(\Gamma)}{n_e'}. \tag{3.4}$$

Parameter values are then determined through a linear interpolation of the accumulated edge lengths of Eq. (3.3); Eq. (3.1) is then evaluated to obtain the physical coordinates of the surface nodes.

3.3 Volume mesh generation

The volume mesh generation algorithm was influenced by the work of [15] and involves an iterative sequence of primitive mesh operations. In particular, edge split, edge collapse, node relocation and edge swap operators are used, as shown in Fig. 3-6. Operations are only performed if the worst quality of the affected (dark gray) triangles is improved by the operation, inspired by the work of [29]. This requirement is relaxed for the node relocation operator and element areas are simply checked to remain positive during the relocation.

The volume mesh generation algorithm is described in Alg. 1. Note that all length and quality computations are performed in $\boldsymbol{\mathcal{M}}$. The aspect ratio function is computed from the eigenvalues, λ, of $\mathcal{M}$ as, $\mathcal{AR} = \sqrt{\lambda_{\max}/\lambda_{\min}}$ and are sorted in decreasing order to target regions of high anisotropy first. The `relocate` function acts as a smoother during the iterative process. Similar to [9] and [15], the force on each node is computed from the L^k norm edge length:

$$\mathbf{f}(\nu) = \sum_{e \in \text{edges}(\nu)} \phi(\ell_M(e))\mathbf{q}^* \tag{3.5}$$

where $\mathbf{q}^*$ is the eigenvector of $\boldsymbol{\mathcal{M}}(\mathbf{x}(\nu))$ closest in direction to e and $\phi(z) = (1 - z^4)\exp(-z^4)$ which applies a tensile force to short edges and a compressive force for long edges as shown in Fig. 3-5. Note that this field is stronger for shorter edges rather than longer edges since edge splits are always valid whereas edge collapses can sometimes introduce inverted elements. Thus, a stronger force is needed to improve the possibility of elongating short edges. The `swaptimize` function seeks the optimal swap configuration about a node star or element neighbours which will improve the worst quality of the affected elements, if possible. The parameters used here are $l_{\min} = 0.70$, $l_{\max} = 1.41$, $n_1 = n_2 = 10$.

Algorithm: $\mathcal{T}_h \leftarrow (\mathcal{G}_h, \boldsymbol{\mathcal{M}})$

```
T_h ← triangulation of nodes on surface mesh, G_h.
while not converged do
    Tag Æ(M(x(ν))) at all nodes ν ∈ T_h and sort.
    for ν ∈ T_h do
        relocate(ν)
        e ← shortest edge attached to ν
        ν_1 ← opposite node to ν on e
        if ℓ_M(e) < l_min and collapse(ν, ν_1) improves worst quality then
            collapse(ν, ν_1)
            swaptimize(ν)
        else
            e ← longest edge attached to ν
            ν_1 ← opposite node to ν on e
            if ℓ_M(e) > l_max and split(ν, ν_1) improves worst quality then
                split(ν, ν_1)
                swaptimize(ν)
            end
        end
    end
    swaptimize(κ) ∀ κ ∈ T_h
end
Perform n_1 iterations of smoothing on all nodes.
Perform n_2 iterations of swaptimization on all nodes.
```

Algorithm 1: Volume mesh generation algorithm

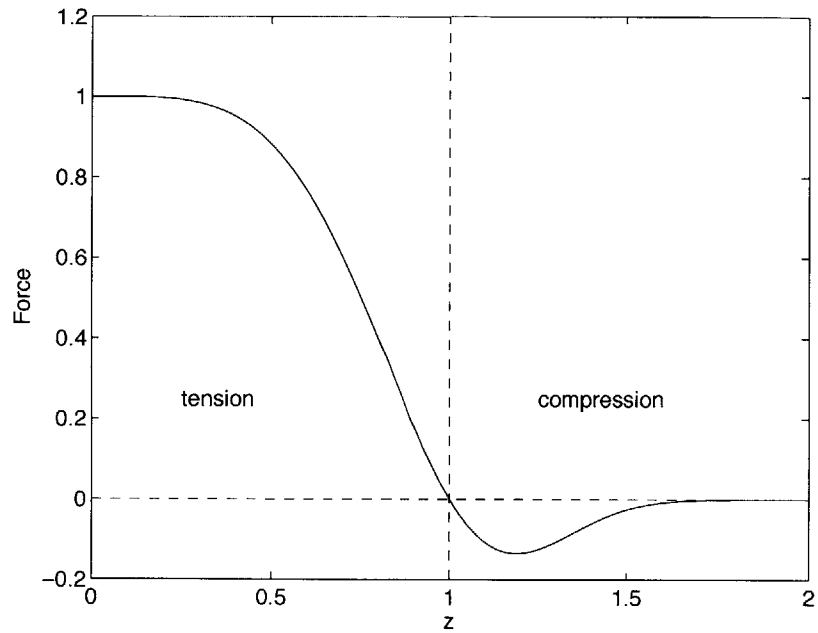


Figure 3-5: Force used to elongate or compress mesh edges according to their length under a metric field

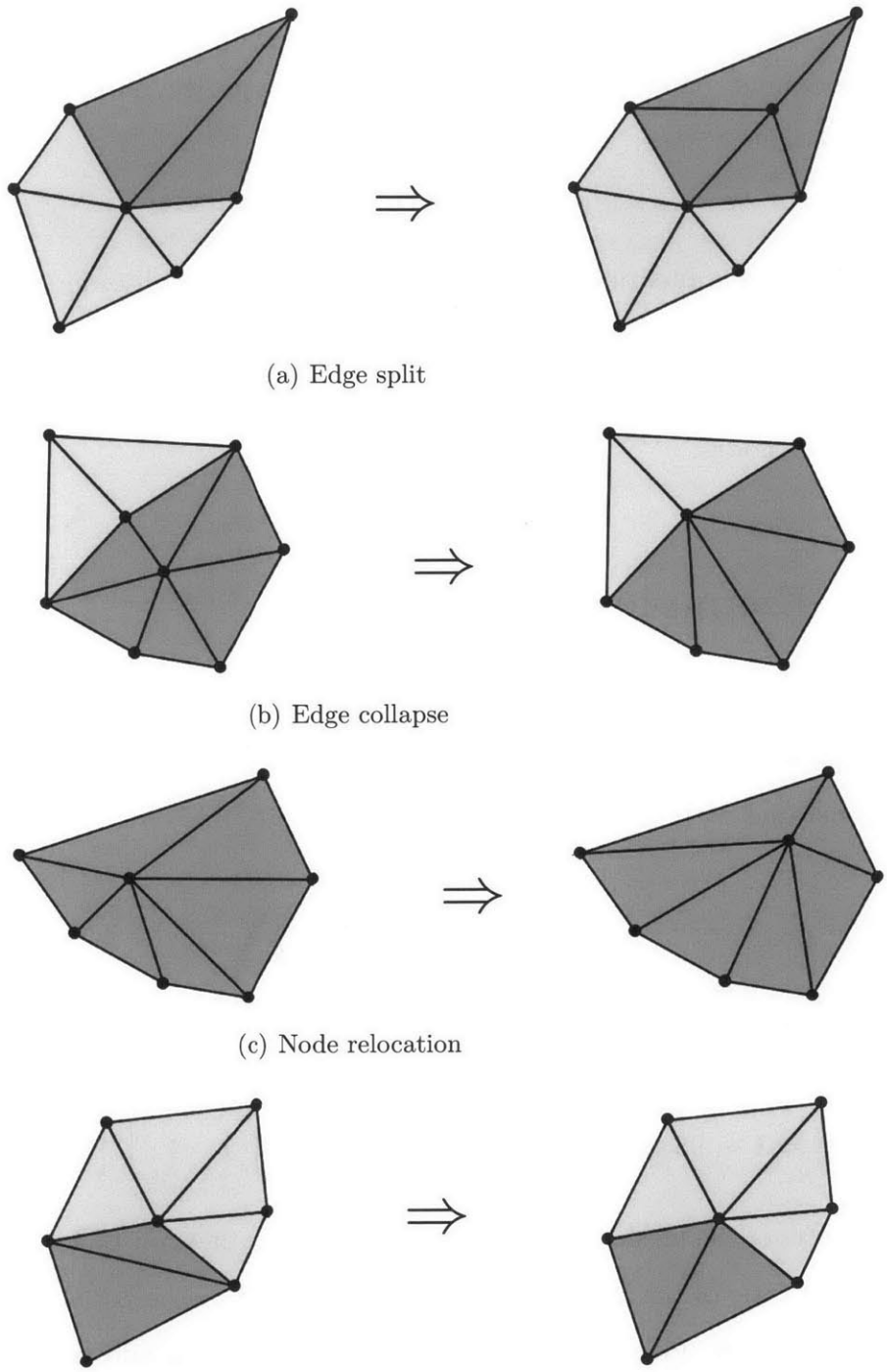
3.4 Curvilinear mesh generation

For problems with curved boundaries, a high-order representation of the geometry is required [4]. In this work, a nonlinear elastic model is used to generate curvilinear meshes [42, 47]. Denote the desired coordinates of the deformed, high-order nodes as $\mathbf{x}$ and those in the reference, undeformed, volume as $\mathbf{x}_0$. The sensitivity of the deformed volume with respect to the initial configuration is

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{x}_0}, \tag{3.6}$$

where $\mathbf{F}$ is the deformation gradient tensor. Imposing a force equilibrium on an arbitrary volume and applying the divergence theorem yields the equilibrium condition

$$\nabla \cdot \mathbf{P} = 0, \tag{3.7}$$



(a) Edge split

(b) Edge collapse

(c) Node relocation

(d) Edge swap

Figure 3-6: Primitive mesh operations

where $\mathbf{P}$ is the Piola-Kirchoff stress tensor, related to the traction vector $\mathbf{t}_0 = \mathbf{P}\mathbf{n}_0$. Here, a neo-Hookean elastic model is used to compute $\mathbf{P}$

$$\mathbf{P}(\mathbf{F}) = \left[\mathbf{F}^{\mathrm{t}}\mathbf{F} + \lambda \ln(\det \mathbf{F})\mathbf{I} - \mathbf{I}\right] \mathbf{F}^{-t} \tag{3.8}$$

where $\lambda = 3$ is the Lamé constant. Eq. (3.7) is solved by imposing Dirichlet conditions on the domain boundaries, corresponding to the projection of the high-order nodes onto the true surface:

$$\mathbf{x}(\mathbf{x}_0) = \texttt{proj}(\mathbf{x}_0, \texttt{CAD curve}) \tag{3.9}$$

where `proj` is a function which projects the inserted high-order node onto the true geometry given, for example, by Eq. (3.1) or some other CAD engine. Furthermore, a $\mathcal{C}^0$ finite-element discretisation is used to solve Eq. (3.7) for the coordinates of the high-order nodes.

3.5 Examples

Two examples now illustrate the developed mesh generation algorithm and validate the expected differences between the L^2- and L^∞-generated meshes. Here, L^2 meshes are generated with `ursa` using $k = 2$.

The geometry for the first example is simply a cylinder within a square domain. An analytic metric field is prescribed:

$$\left.\begin{aligned} h_1(x,y) &= 1 - .95\exp\left(-\tfrac{|x+y|}{\sqrt{2}}\right) \\ h_2(x,y) &= 1 - .95\exp\left(-\tfrac{|y-x|}{\sqrt{2}}\right) \end{aligned}\right\} \mathcal{M}(x,y) = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \frac{1}{h_1^2} & 0 \\ 0 & \frac{1}{h_2^2} \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}.$$

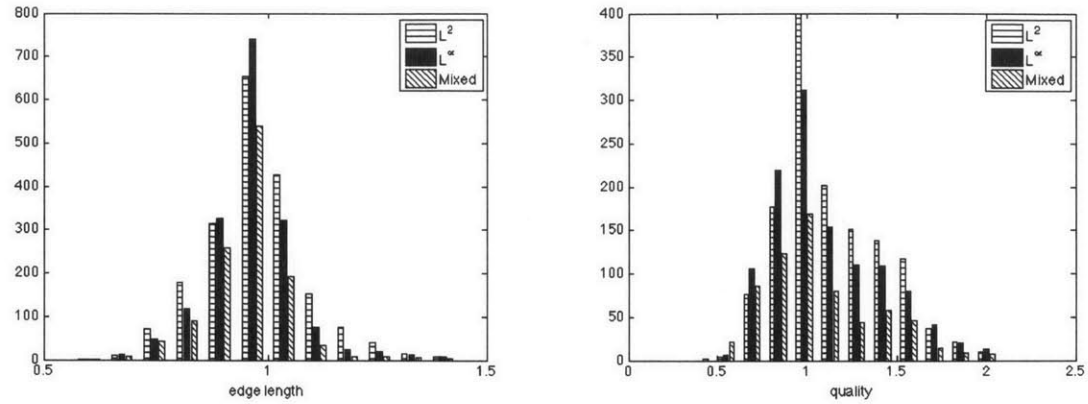
The results of the mesh generation are shown in the first row of Table 3.1 and Fig. 3-9. Note that ratio of the number of triangles generated by the L^∞ and L^2 norm methods is 0.877 which is close to the theoretical ratio of $\sqrt{3}/2$. In addition the quad cover for this case is about 80%. The distributions of the edge lengths and element qualities for this problem are given in Fig. 3-7 which shows that most of the generated mesh edges have a unit edge length under the background metric field. The element qualities, while exhibiting a spike near the $q = 1$ bin, are more evenly distributed which is a result of the decision-making process during the mesh generation algorithm. Since the worst quality elements are targeted during each mesh operation, this tends to

equidistribute the qualities of the resulting elements.

Example/Method	L^2	L^∞	Mixed
Analytic X-field, cylinder	t = 1336	t = 1172	q = 510, t = 152
Discrete field, 3-element MDA airfoil	t = 4839	t = 4167	q = 1627, t = 913

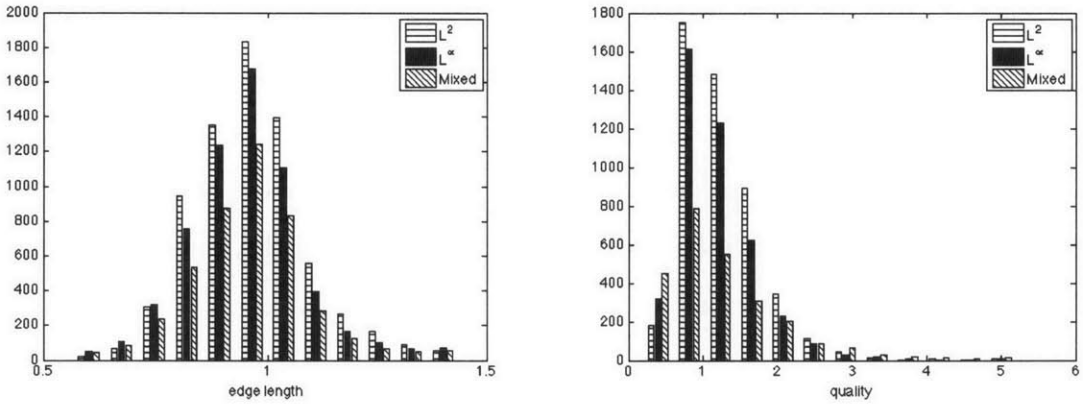
Table 3.1: Number of triangles (t) and quads (q) for the mesh generation examples

The second example considered is more practical in the context of the current adaptive simulation framework. The implied metric field of an input triangulation around a three-element MDA airfoil was used to test the discrete evaluation of a metric field. The resulting meshes are shown in Fig. 3-9. Again the ratio of L^∞ to L^2 triangles (0.861) is close to the expected value of $\sqrt{3}/2$; the quad cover is only 65% which is perhaps due to a combination of the anisotropy (about 1.4×10^2) and the geometry in this problem. Edge lengths are still clustered near unity for all mesh types (Fig. 3-8(a)), however, the spread is slightly larger than the analytic cylinder case due to difficulties in conforming to this metric request. Element qualities of Fig. 3-8(b) are still concentrated near unity, however, difficulties in conforming to the requested metric are evident in the existence of quality bins with values ranging from $3 - 5$.



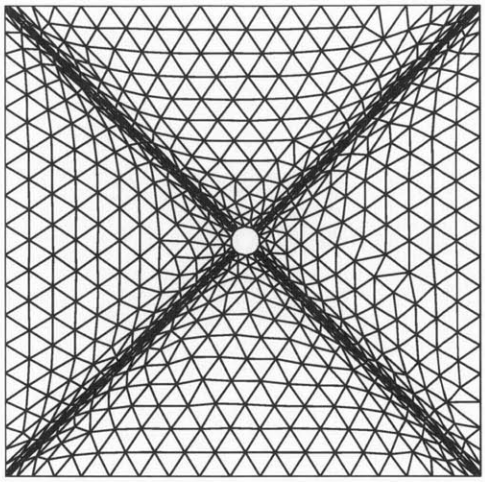
(a) Edge length (b) Quality

Figure 3-7: Edge length and element quality distributions for the cylinder example with an analytic metric field

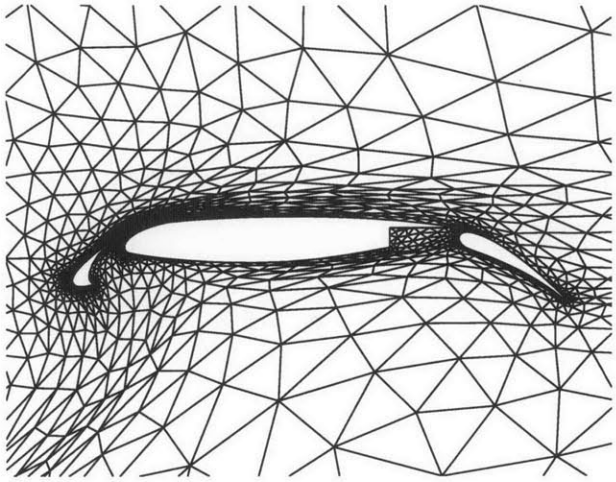


(a) Edge length (b) Quality

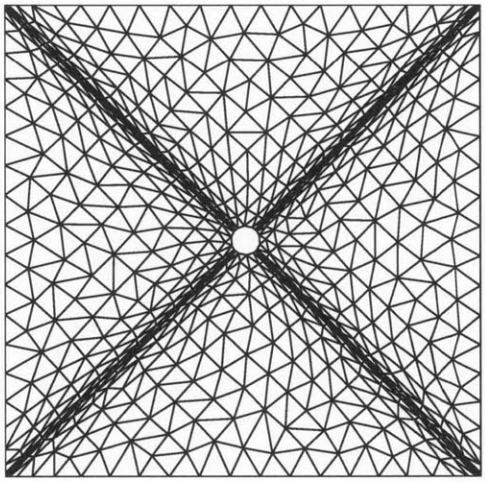
Figure 3-8: Edge length and element quality distributions for the three-element MDA airfoil example with a discrete metric field



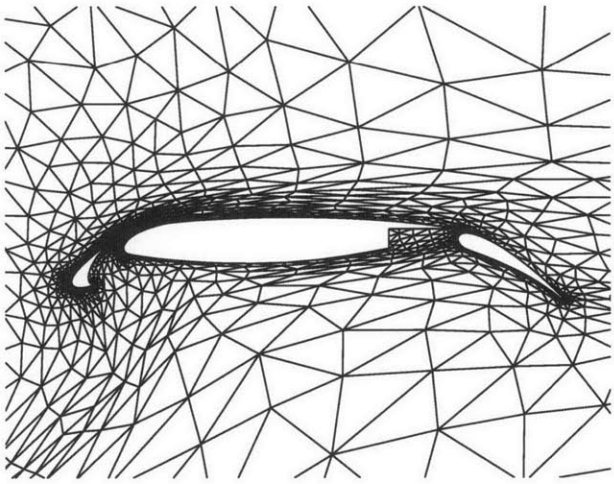
(a) L^2 cylinder



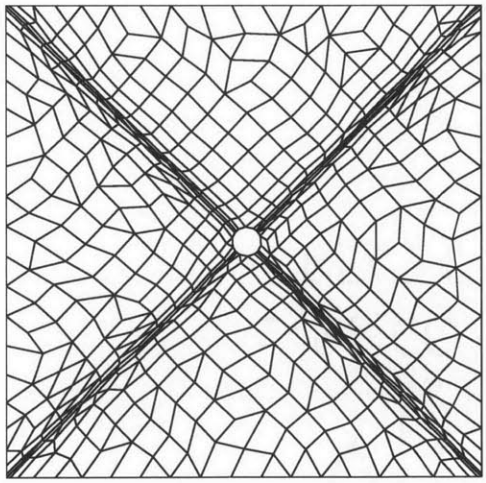
(b) L^2 MDA airfoil



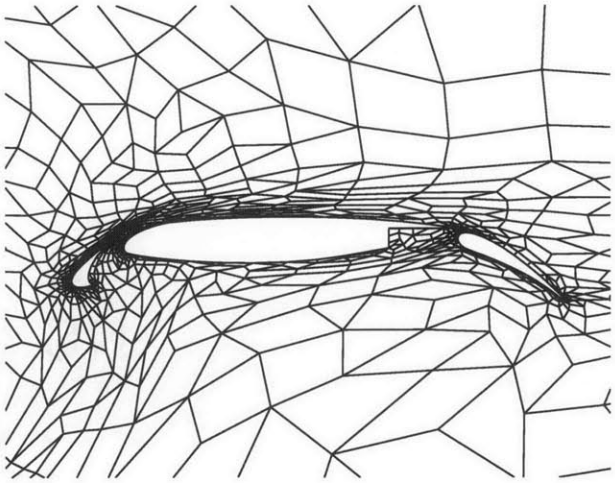
(c) L^∞ cylinder



(d) L^∞ MDA airfoil



(e) Mixed-element cylinder



(f) Mixed-element MDA airfoil

Figure 3-9: Meshes generated by analytic (cylinder) and discrete (three-element MDA airfoil) metric fields using equilateral (L^2), right (L^∞) and mixed-element mesh generation techniques

Chapter 4

Numerical results

The mixed-element mesh adaptation algorithm is now studied and compared to L^2- and L^∞-based methods. The algorithm is first applied to isotropic and anisotropic L^2 error control problems to ensure an optimal mesh grading is obtained. Conclusions are then drawn from these studies and the algorithm is further applied to aerodynamics problems.

4.1 L^2 error control

The goal of this section is to isolate the adaptation algorithm and study its behaviour when applied to isotropic and anisotropic problems. These problems are designed to ensure MOESS generates optimal mesh gradings. Instead of employing the DWR error estimator, the exact L^2 error arising from the projection of a prescribed solution onto a certain function space is used to drive the adaptation:

$$\mathbf{u}_{h,p} = \underset{\mathbf{v}_{h,p}\in\mathcal{V}_{h,p}}{\operatorname{arginf}} \|\mathbf{u}-\mathbf{v}_{h,p}\|^2_{L^2(\Omega)} = \int_\Omega (\mathbf{u}-\mathbf{v}_{h,p})^2 \,\mathrm{d}\mathbf{x}, \tag{4.1}$$

where $\mathbf{u}$ is the analytic solution. Solving Eq. (4.1) simply requires the inversion of the local element mass matrix since $\mathcal{V}_{h,p}$ is discontinuous across elements. The local

error is then given by

$$\eta_\kappa = \|\mathbf{u} - \mathbf{u}_{h,p}\|^2_{L^2(\kappa)} = \int_\kappa (\mathbf{u} - \mathbf{u}_{h,p})^2 \, \mathrm{d}\mathbf{x} \tag{4.2}$$

which is used to drive the adaptation process.

Following the work of Yano [58], the optimal metric tensor field is obtained by minimizing the L^2 approximation error subject to a target number of degrees of freedom. The implied metric field of the adapted mesh is then compared with these analytic mesh gradings. Quadrilateral elements are first divided into triangles and the implied metric of each triangle is then used in the correlation.

4.1.1 r^α-type corner singularity

MOESS is first applied to an isotropic problem with a corner singularity. The domain is set to $\Omega = [0,1]^2$ and the analytic solution is given by

$$u(r,\theta) = r^\alpha \sin[\alpha(\theta + \theta_0)], \quad r^2 = x^2 + y^2, \theta = \arctan(y/x). \tag{4.3}$$

The singularity strength is set to $\alpha = 2/3$ and the offset angle is set to $\theta_0 = \pi/2$. The optimal mesh grading is [58]

$$h(r) = cr^k, \qquad k = 1 - \frac{\alpha + 1}{p + 2}, \tag{4.4}$$

where c is a constant independent of r but dependent on the prescribed number of degrees of freedom.

Methodology Thirty adaptation iterations, using $p = 1$ and $p = 3$ discretisations, are used to obtain equilateral (L^2), right (L^∞) and mixed-element adapted meshes with 4,000 degrees of freedom. The element sizes are computed as $h_\kappa = [\det(\mathcal{M}_\kappa)]^{-1/4}$ where $\mathcal{M}_\kappa$ is the element-implied metric tensor given by Eq. (2.19); these sizes are plotted against the radial distance of the element centroid from the origin. A refine-

ment factor of $r = 2$ was used for all mesh types since this problem is essentially isotropic and the rate of the mesh grading is not very strong.

Results The error and degree of freedom convergence plots are shown in Fig. 4-1 for both $p = 1$ and $p = 3$ discretisations. In the $p = 1$ case, MOESS exhibits smooth convergence behaviour without significant oscillations after converging to an error on the order of 10^{-9}. The meshes obtained with the mixed-element adaptation seem to overshoot the requested degree of freedom count by 10 % which is likely a result of the inability to determine a priori how many quadrilaterals or triangles will be created by the mesh generation algorithm. Ideally, the resultant mesh would be composed entirely of quadrilaterals, however, realizing this while conforming to an arbitrary metric field is rarely achievable.

The $p = 3$ error and degree of freedom plots of Fig. 4-1 appear somewhat oscillatory over the adaptation iterations, however, this is likely due to precision errors at the resultant error levels.

The $p = 1$ and $p = 3$ meshes of Figs. 4-2 and 4-3 appear similar in structure across all element types. However, the mixed-element meshes produce the most scatter about the regression lines; this is likely due to the fact that this problem is better suited for L^2 triangles since the solution is expressed in polar coordinates. It is interesting to note, however, that the correct mesh grading away from the corner singularity ($k^* = 0.44$ for $p = 1$ and $k^* = 0.67$ for $p = 3$) is obtained with all mesh types and all solution orders; these rates are tabulated in Table 4.1. The element sizes in the mixed-element cases exhibit far more scatter than the L^2- and L^∞-generated meshes which is, again, attributable to difficulties on both metric optimization and mesh generation algorithms in agreeing upon the number of elements to generate as the adaptations progress.

Note the L^∞-generated $p = 3$ mesh exhibits the least scatter in the final mesh

Table 4.1: Mesh size correlation rates away from the origin for the L^2 error control applied to the corner singularity problem

	L^2	L^∞	Mixed
$p = 1$	0.45	0.45	0.44
$p = 3$	0.71	0.68	0.68

grading and converges closer to the requested degree-of-freedom count than its L^2 and mixed-element counterparts.

4.1.2 $2d$ boundary layer

MOESS is now applied to the L^2 error control of a two-dimensional boundary layer solution within $\Omega \in [0,1]^2$ with y being the streamwise direction. The problem is essentially one-dimensional, however, a regularization term is added to limit the anisotropy in the streamwise direction; the projected solution is given by

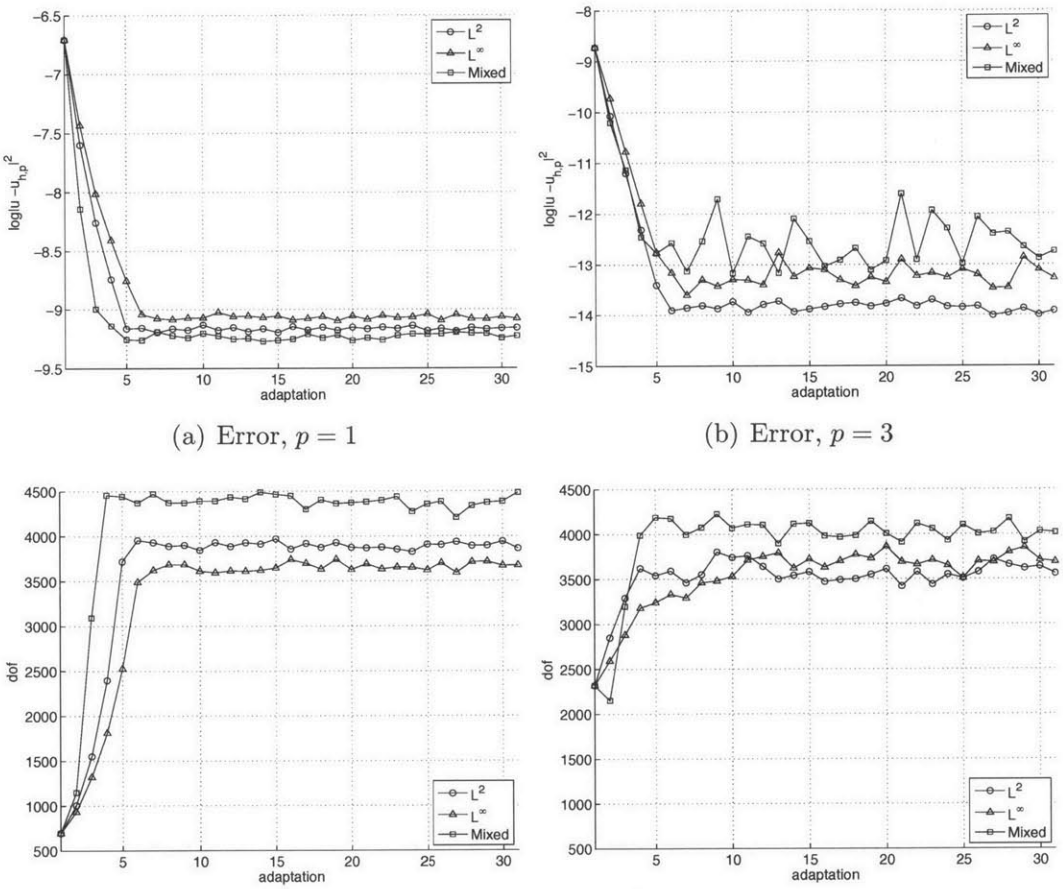
$$u(x,y) = \exp\left(-\frac{x}{\epsilon}\right) + \frac{\beta}{(p+1)!}y^{p+1} \tag{4.5}$$

where ϵ is a characteristic length and β is used to control the regularization in the streamwise direction. Solving for the optimal mesh grading and expected anistropy ratios perpendicular to the wall yields [58]

$$h_1 = c\,\exp(k_1 x), \quad \frac{1}{k_1} = \epsilon\left(p + \frac{3}{2}\right)\left(1 - \frac{1}{4p^2 + 12p + 9}\right), \tag{4.6}$$

$$Æ = Æ_0 \exp(k_{Æ} x), \quad Æ_0 = \frac{1}{\epsilon\beta^{p+1}}, \quad k_{Æ} = -\frac{1}{\epsilon(p+1)}. \tag{4.7}$$

Methodology Twenty-five adaptation iterations are used to generate $4,000$ degree-of-freedom meshes using all three approaches: L^2 (equilateral), L^∞ (right triangle) and mixed-element meshes at both $p = 1$ and $p = 3$ solution orders. The convergence behaviour of both the L^2 approximation error as well as the degrees of freedom are studied. In addition, all meshes should exhibit reasonable conformity with the analytic sizes and anisotropy ratios. Denote h_1 and h_2 as the mesh sizes perpendicular



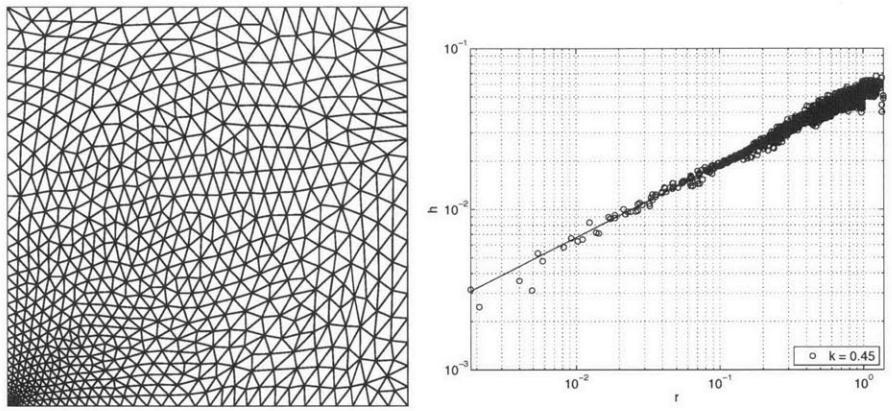
(a) Error, $p = 1$

(b) Error, $p = 3$

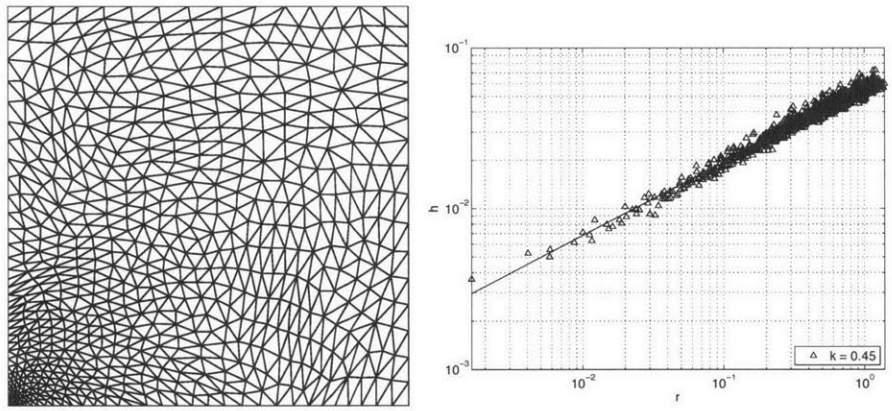
(c) dof, $p = 1$

(d) dof, $p = 3$

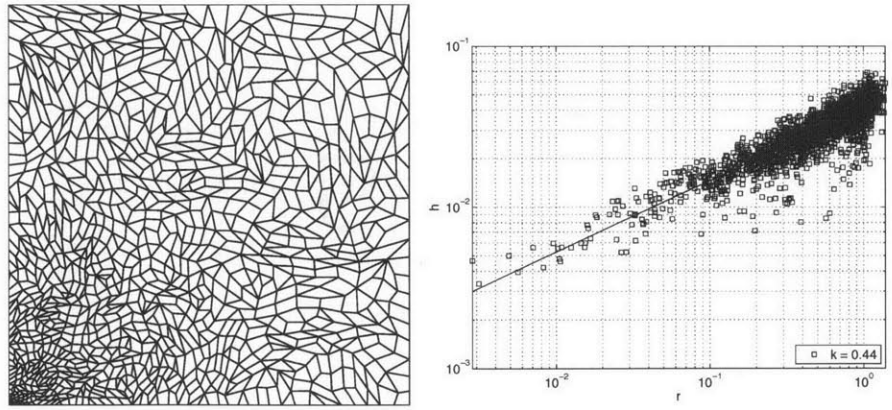
Figure 4-1: Error and degrees of freedom convergence for the corner singularity problem



(a) L^2

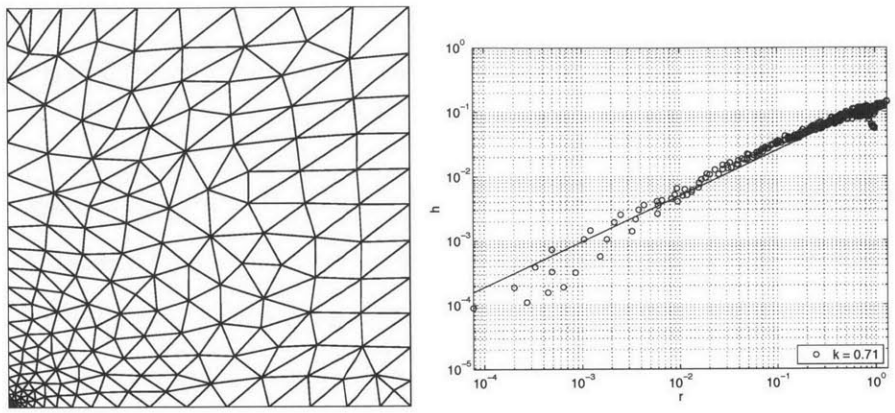


(b) L^∞

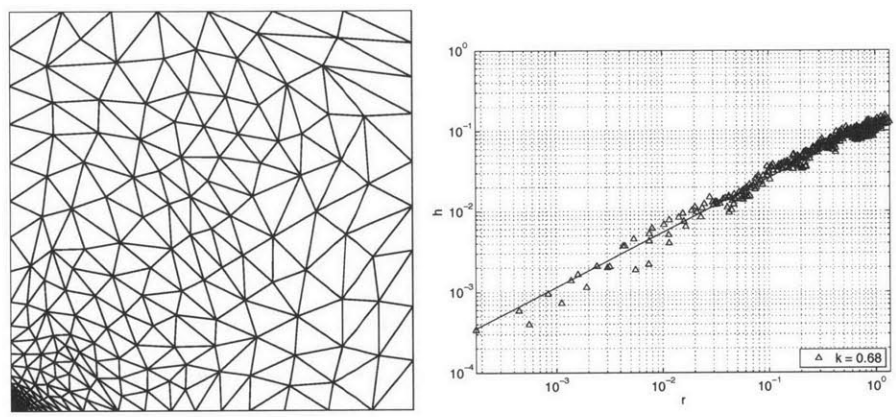


(c) Mixed

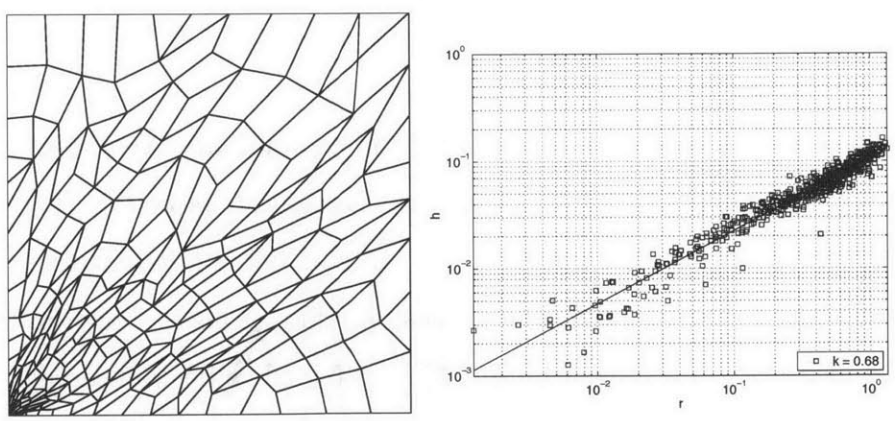
Figure 4-2: $p = 1$, dof = 4,000 optimized for the corner singularity problem, $k^* = 0.44$



(a) L^2



(b) L^∞



(c) Mixed

Figure 4-3: $p = 3$, dof = 4,000 optimized for the corner singularity problem, $k^* = 0.67$

and parallel to the wall ($x = 0$), respectively. These are extracted from the element implied metric tensor, Eq. 2.19, as $h_1 = (\mathcal{M}_\kappa)_{1,1}^{-1/2}$ and $h_2 = (\mathcal{M}_\kappa)_{2,2}^{-1/2}$. The aspect ratio is then given by $Æ = h_2/h_1$. The $p = 1$ optimized meshes should have $k_1 = 41.7$, $k_{Æ} = -50$ whereas the $p = 3$ optimized meshes should have $k_1 = 22.5$ and $k_{Æ} = -25$.

Results First consider the convergence behaviour of the adaptation algorithm, as shown in Fig. 4-4. In the $p = 1$ case, both the L^2- and L^∞-driven adaptations converge steadily to an L^2 error of about 10^{-8} and achieve the target number of degrees of freedom within a reasonable tolerance. The mixed-element meshes converge steadily to a slightly higher error, however, they overshoot the requested number of degrees of freedom by over 10%, in which case, the expected L^2 error should be lower than the L^2 and L^∞ meshes. The fact that the cost model only assumes quadrilateral elements yet the mesh generator returns both triangles and quadrilaterals makes it difficult for the metric optimization to converge upon a requested metric over the course of the adaptations. This phenomenon is further seen in the $p = 3$ case where the degree-of-freedom count oscillates and the algorithm never appears to converge. This also affects the final error of the mixed-element meshes, which can be seen to be two orders of magnitude larger than the L^2 and L^∞ meshes. It is important to note that the refinement factor, r in Eq. (2.30), was doubled in the mixed-element cases in order to capture the anisotropy present in the boundary layer. When $r = 2$, the algorithm initially detects the need for refinement in the boundary layer, however, since the error model contains information from the diagonal splits of Fig. 2-3, which do not effectively refine the original element, the error model is less sensitive to further refinement and fails to capture the correct mesh grading. This is shown in the $p = 3$, 4,000 degrees-of-freedom optimized mesh in Fig. 4-5. It has been observed, however, that the use of high refinement factors tend to create oscillations in the convergence of the output, which may also explain the behaviour observed in Fig. 4-4.

Despite some convergence issues created by problems with degree-of-freedom assumptions and higher refinement factors, size and aspect ratio rates are reasonable.

Table 4.2: Mesh size and aspect ratio correlation rates away from the wall for the L^2 error control applied to the $2d$ boundary layer problem

	L^2		L^∞		Mixed	
	$p=1$	$p=3$	$p=1$	$p=3$	$p=1$	$p=3$
h_1	41.84	19.22	39.71	20.86	36.40	18.51
$\mathcal{AR}$	-50.61	-20.36	-48.96	-25.38	-47.09	-18.52

The mixed-element adaptation algorithm performs better for $p = 1$ than for $p = 3$, however, the size and aspect ratio distributions exhibit far more scatter than the L^2 and L^∞ meshes. Of all three element types, the L^∞-generated meshes conform very well to the expected rates, especially in the $p = 3$ case. The use of this element type is proving itself quite useful in adding structure to the meshes of Figs. 4-6 (b) and 4-7 (b).

4.2 Compressible Navier-Stokes equations

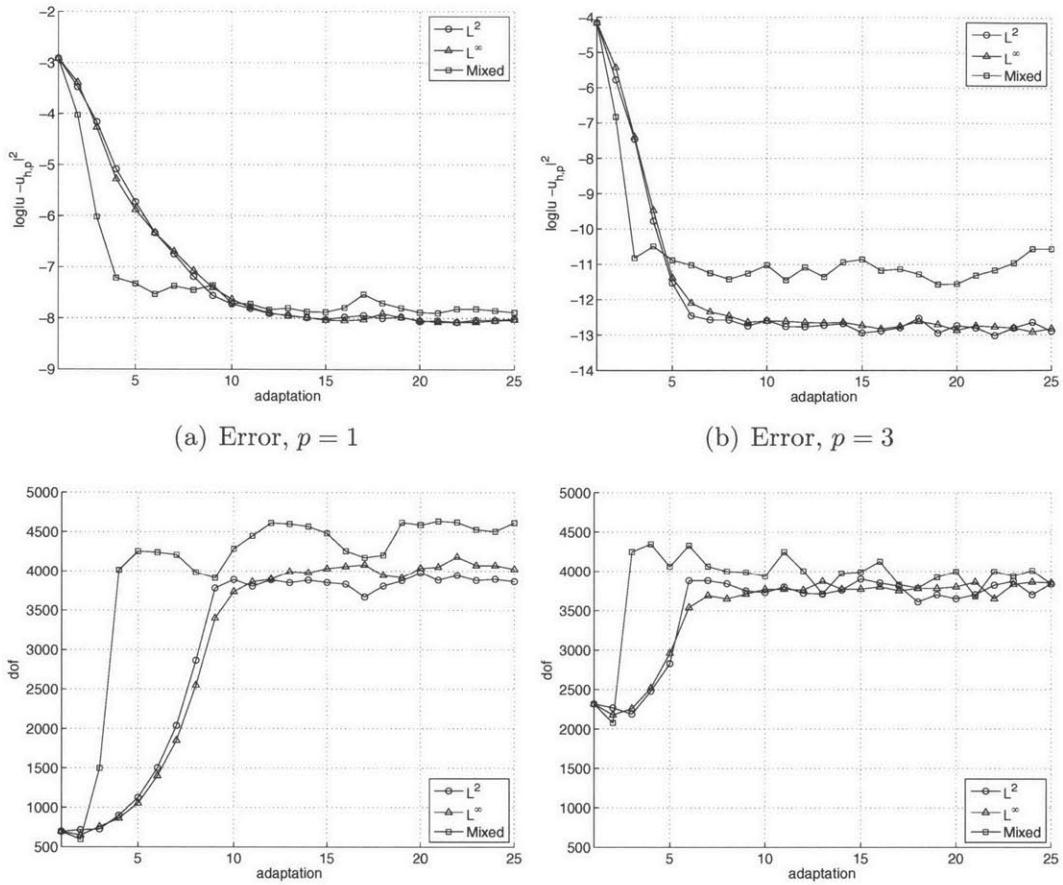
The mixed-element mesh adaptation algorithm is now applied to practical aerodynamics problems. First, the ability of MOESS to capture shock waves is observed by studying an inviscid, supersonic case over a wedge. Next, both subsonic and transonic viscous test cases are studied. These tests aim to demonstrate as well as compare the adaptation algorithm using different element types; in some cases, theoretical convergence rates for the output error are provided.

4.2.1 Euler equations

The conservative state vector and convective flux for the Euler equations are

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \end{bmatrix}, \quad \mathcal{F}_c = \begin{bmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\otimes\mathbf{v} + p\mathbf{I} \\ \rho H\mathbf{v} \end{bmatrix}.$$

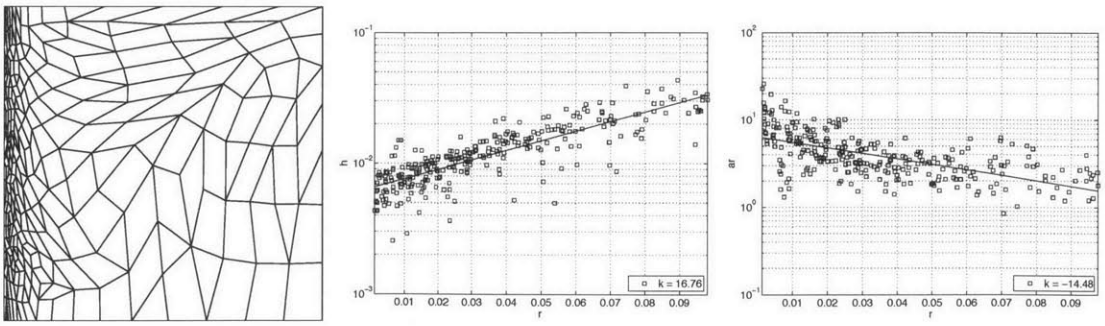
where p is the pressure, $p = (\gamma - 1)(\rho E - \|\mathbf{v}\|^2/2)$, and H is the total enthalpy, $H = E + p/\rho$. The ideal gas law, $p = \rho RT$, closes the system; R is the gas constant.



(a) Error, $p = 1$ (b) Error, $p = 3$

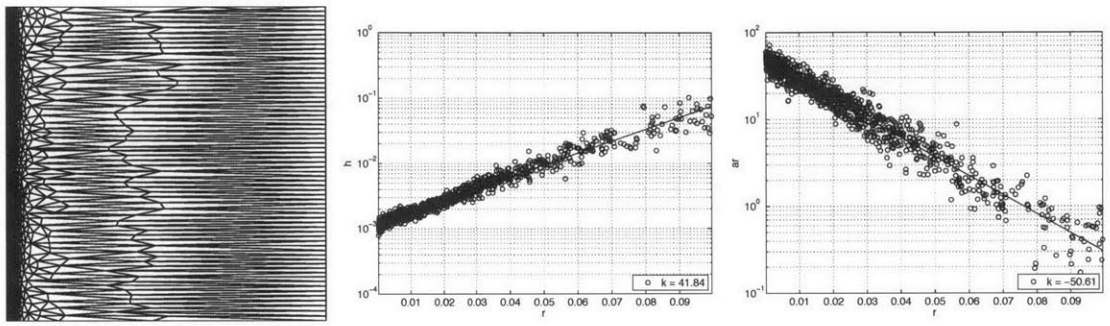
(c) dof, $p = 1$ (d) dof, $p = 3$

Figure 4-4: Error and degrees of freedom convergence for the $2d$ boundary layer problem. The refinement factor for triangulations is set to $r = 2$ whereas it is set to $r = 4$ for mixed-element meshes.

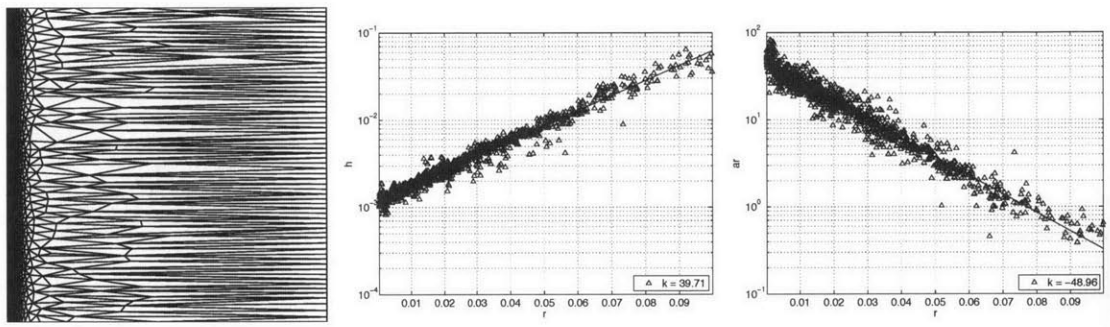


(a) Mesh (b) h_1 (c) $\mathcal{AR}$

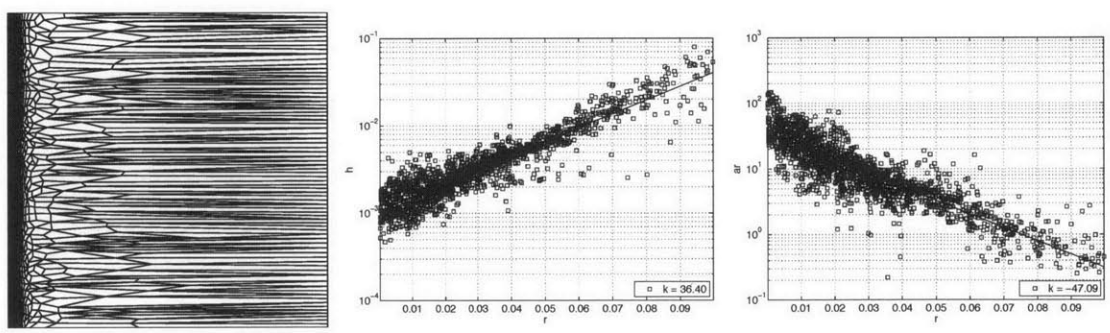
Figure 4-5: $p = 3$, dof $= 4{,}000$ optimized mixed element meshe along with corresponding size and aspect ratio distributions for the $2d$ boundary layer problem with a refinement factor of $r = 2$, $k^*_{h_1} = 22.5$, $k^*_{ar} = -25$



(a) L^2

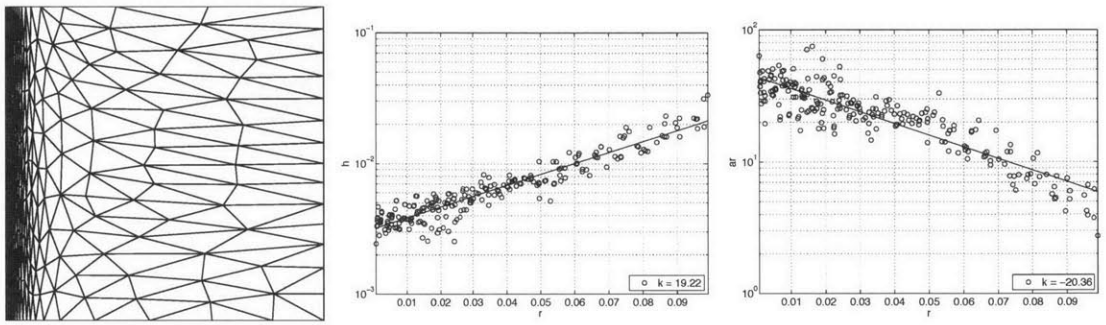


(b) L^∞

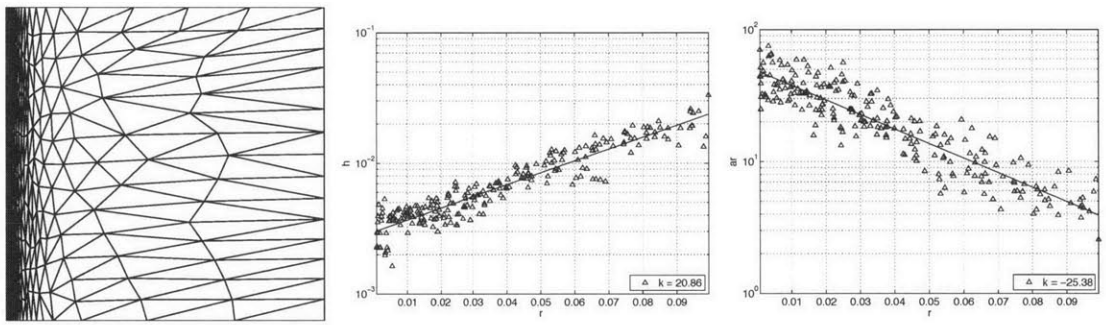


(c) Mixed

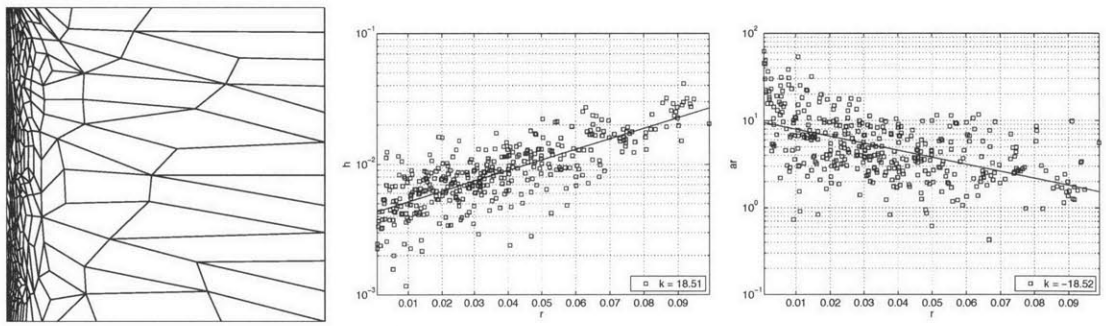
Figure 4-6: $p = 1$, dof = 4,000 optimized meshes along with corresponding size and aspect ratio distributions for the $2d$ boundary layer problem, $k^*_{h_1} = 41.7$, $k^*_{ar} = -50$. The refinement factor for triangulations is set to $r = 2$ whereas it is set to $r = 4$ for mixed-element meshes.



(a) L^2



(b) L^∞



(c) Mixed

Figure 4-7: $p = 3$, dof = 4,000 optimized meshes along with corresponding size and aspect ratio distributions for the $2d$ boundary layer problem, $k^*_{h_1} = 22.5$, $k^*_{ar} = -25$. The refinement factor for triangulations is set to $r = 2$ whereas it is set to $r = 4$ for mixed-element meshes.

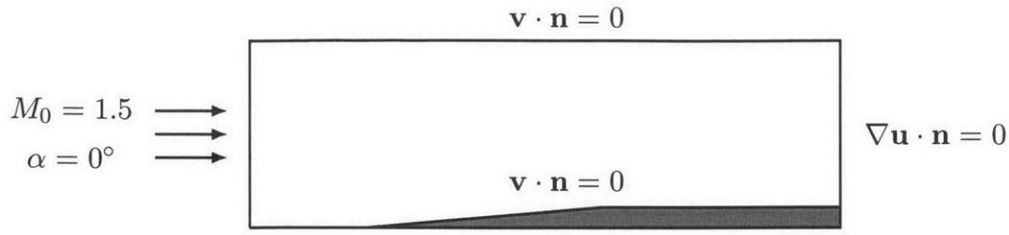


Figure 4-8: Supersonic wedge problem setup

The system is augmented with a PDE-based shock-capturing model for the artificial viscosity as developed by Barter [3] and later modified by Yano [58] to employ tensor-based element size information in the shock-capturing model.

Methodology Here the supersonic flow over a wedge is computed, as sketched in Fig. 4-8. Flow tangency boundary conditions are used for the wedge surface and the inlet is specified as $M_0 = 1.5$, $\alpha = 0°$. The output functional is the integrated force tangent to all domain boundaries except the inlet. Twenty adaptation iterations are first performed with both L^∞- and L^2-based adaptation techniques for $p = 1$ and $p = 3$ discretisation orders at 10,000, 20,000 and 30,000 target degree-of-freedom counts. Subsequently, the mixed-element adaptation algorithm is started from the optimized L^∞ meshes and ten adaptations are used to produce adapted meshes for both $p = 1$ and $p = 3$ solution orders. Note that a refinement factor of $r = 2$ is used for triangulations but $r = 3$ is used for mixed-element meshes. This is lower than the $r = 4$ value used in the L^2 error control of a $2d$ boundary layer since the mixed-element based adaptation is restarted from an L^∞-adapted mesh and is already near the optimal configuration. The reference output is obtained from a 100,000 degrees of freedom, $p = 3$ adaptive simulation.

Results The mixed-element meshes for both $p = 1$ and $p = 3$ solution orders, obtained following the procedure mentioned above, are shown in Fig. 4-9 and 4-10. Notice the $p = 1$ meshes capture additional features, present in both the primal and dual solution which is a result of the DWR error estimator influenced by both the residual at the $p+1$ prolongated state, as well as the $p+1$ adjoint solution. However,

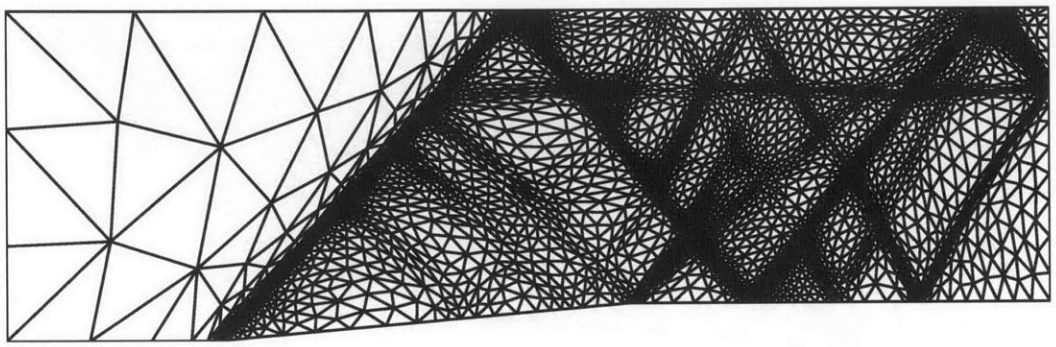
Table 4.3: Properties of the 30,000 degree-of-freedom adapted meshes for the supersonic wedge case

	L^2		L^∞		Mixed	
	$p=1$	$p=3$	$p=1$	$p=3$	$p=1$	$p=3$
# quad	0	0	0	0	6,986	1,786
# tri	10,061	3,113	10,241	3,235	1,915	592
$\mathcal{AR}$	2.64×10^1	23.3×10^1	3.36×10^1	2.41×10^1	3.29×10^1	3.09×10^2

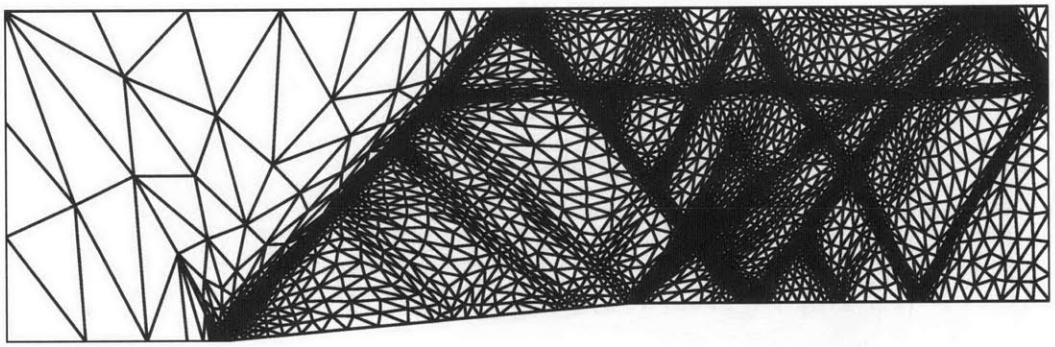
this phenomenon is less evident in the $p = 3$ adapted meshes since the richer polynomial space better captures these features. Some properties of the final mixed-element meshes are given in Table 4.3. Note the higher anisotropy present in the $p = 3$ mixed-element mesh. Neither anisotropy nor the linear geometry were problematic in this case; the quad cover was 78% for the $p = 1$ mesh and 75% for the $p = 3$ mesh.

The Mach number, total pressure and total temperature distributions across a horizontal line through the domain at $y = 0.4$ are shown in Fig. 4-11 where the solutions are taken from the 30,000 degree-of-freedom adapted simulations. The $p = 3$ solution better captures the jump in the Mach number across shocks than the $p = 1$ solution for all element types in Figs. 4-11(a) and (c). In addition, observe the total temperature remains essentially constant as it should for an inviscid flow in Figs. 4-11(e) and (f). The $p = 1$ meshes exhibit jumps in the total pressure across shocks but remains fairly constant in regions of uniform flow. However, the $p = 3$, L^2 mesh exhibits some variation in total pressure near the shock at $x = 1.03$ whereas the mixed-element mesh captures the expected constant total pressure. This is likely due to the artificial viscosity added by the shock-capturing model in this region. In Fig. 4-12, notice the L^2 mesh contains slightly more artificial viscosity in the shock regions. In particular, the artificial viscosity distribution near $x = 1.03$ is slightly greater for the L^2 mesh and is likely due to a poorer resolution of the shock.

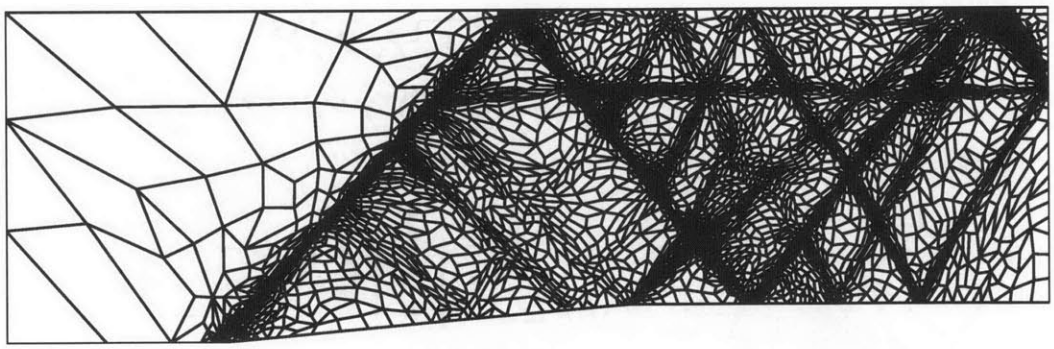
The convergence rate of the estimated drag error versus the characteristic mesh size, $h \sim (\text{dof})^{-1/2}$, for all mesh types are plotted in Fig. 4-13. Due to the presence of shock waves and the first-order upwinding scheme used to capture them, optimal



(a) L^2

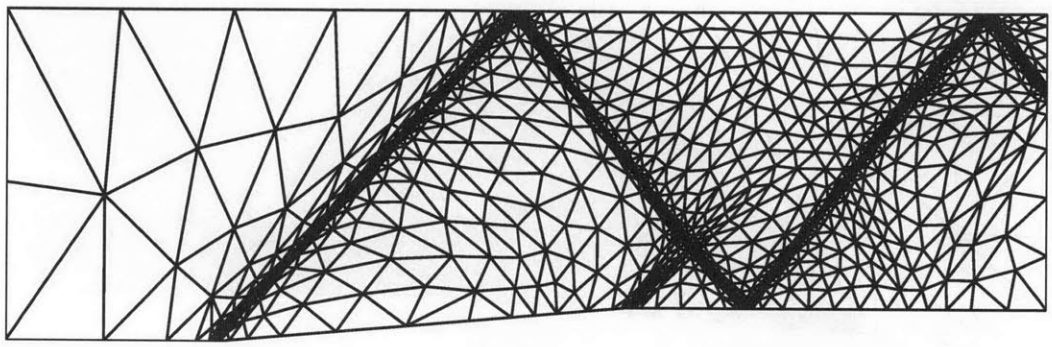


(b) L^∞

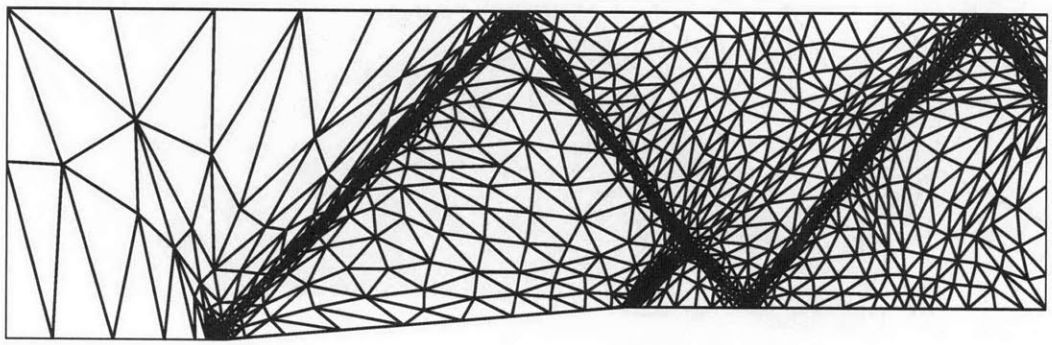


(c) Mixed

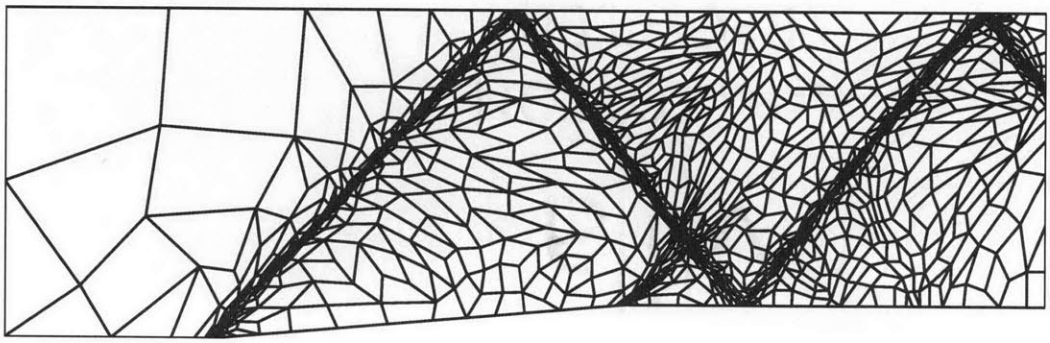
Figure 4-9: $p = 1$, 30,000 degree-of-freedom optimized meshes for the supersonic wedge case



(a) L^2

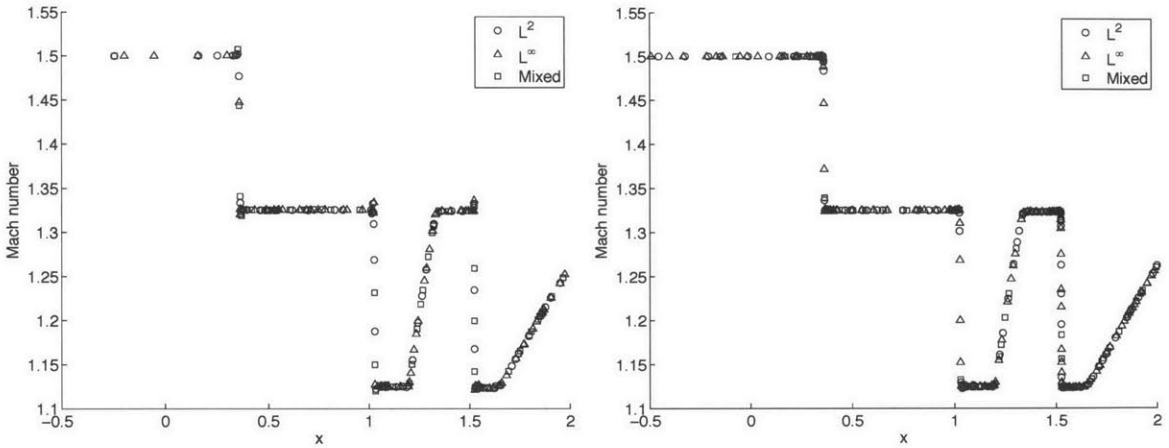


(b) L^∞



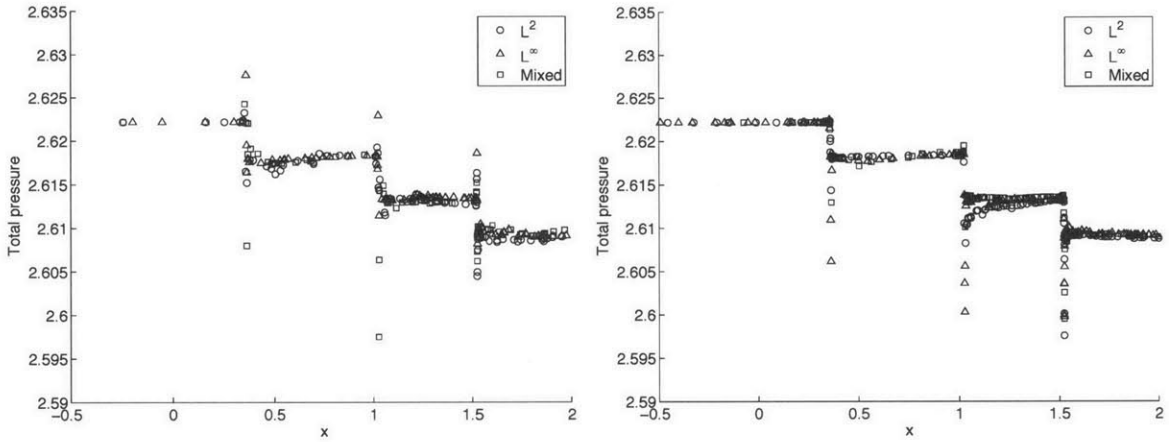
(c) Mixed

Figure 4-10: $p = 3$, 30,000 degree-of-freedom optimized meshes for the supersonic wedge case



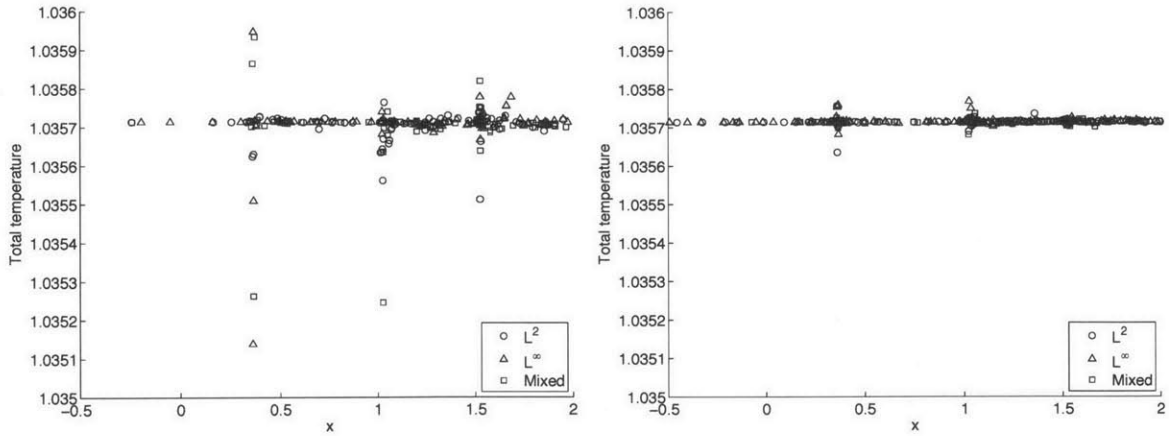
(a) Mach number, $p = 1$

(b) Mach number, $p = 3$



(c) Total pressure, $p = 1$

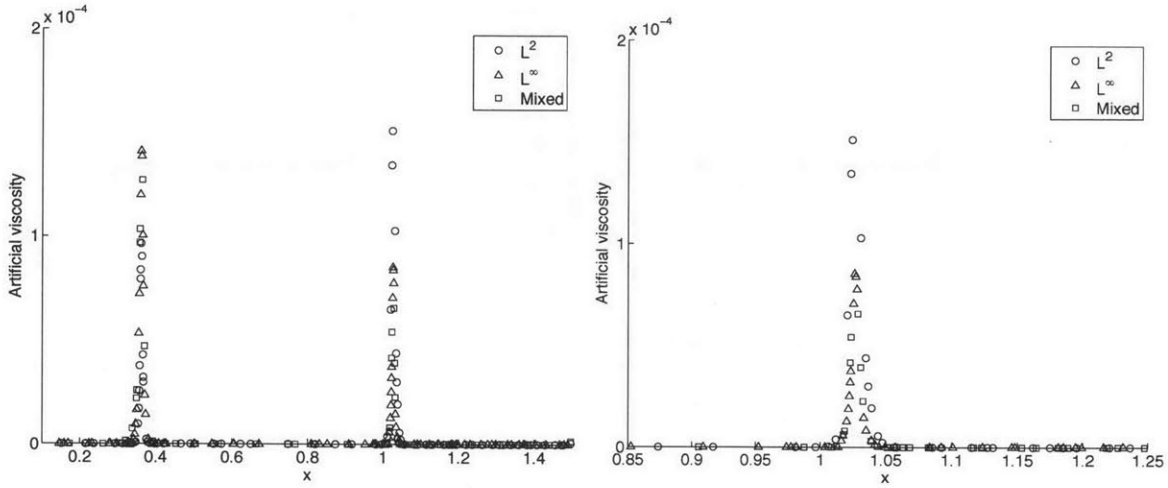
(d) Total pressure, $p = 3$



(e) Total temperature, $p = 1$

(f) Total temperature, $p = 3$

Figure 4-11: Mach number, total pressure and total temperature distributions across a horizontal line at $y = 0.4$ for the supersonic wedge case for all mesh types with $p = 1$ and $p = 3$ discretisations



(a) Artificial viscosity peaks near shocks

(b) Artificial viscosity near shock at $x = 1.03$

Figure 4-12: Artificial viscosity added near shocks for the $p = 3$ solution to the supersonic wedge case with L^2, L^∞ and mixed-element meshes

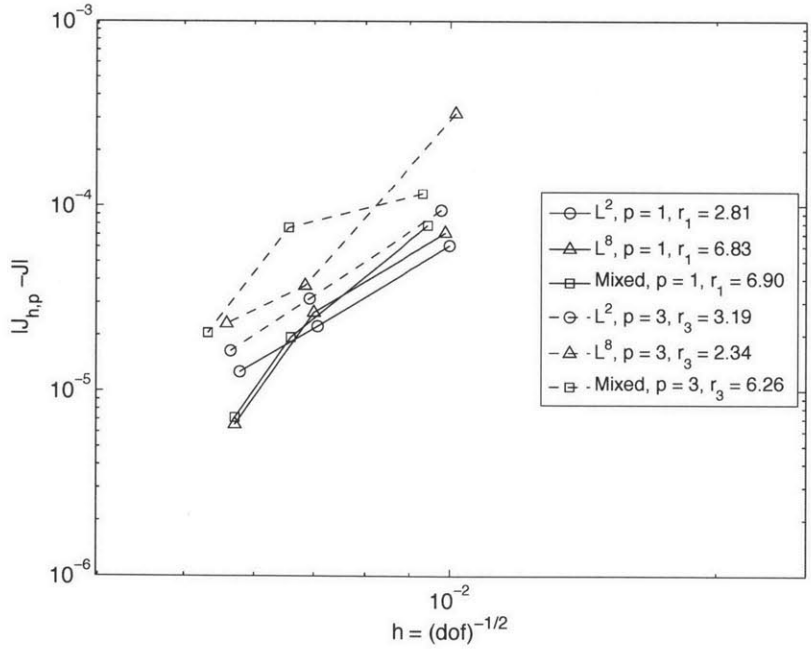


Figure 4-13: Convergence rates for $p = 1$ and $p = 3$ adaptations for the supersonic wedge case with L^2, L^∞ and mixed-element meshes

h^{2p} [49] convergence rates cannot be expected. The estimated error on the mixed-element meshes is larger than both L^2 and L^∞ methods, despite the mixed-element meshes usually overshooting the requested number of degrees of freedom. Again, this is likely due to the inability of MOESS to predict the number of quads or triangles returned by the mesh generator upon each adaptation iteration which is used in the cost model to constrain the metric optimization problem, conversely affecting the convergence of the error estimate. The results, nonetheless, demonstrate the ability of the adaptation algorithm to handle mixed-element meshes.

4.2.2 Reynolds-averaged Navier-Stokes equations

The compressible Reynolds-averaged Navier-Stokes (RANS) equations are obtained by decomposing the flow variables into mean and fluctuating components. The Favre averaging procedure gives rise to apparent stress gradient terms and requires the introduction of additional equations to close the system. In this work, the Spalart-Allmaras (SA) turbulence model, modified for negative values of the working variable $\tilde{\nu}$ is used [2]. The conservative state vector is given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \\ \rho\tilde{\nu} \end{bmatrix}$$

where $\tilde{\nu}$ is the working variable for the SA turbulence model, related to the eddy viscosity via

$$\mu_t = \begin{cases} \rho\tilde{\nu} f_{v1}, & \tilde{\nu} \geq 0 \\ 0, & \tilde{\nu} < 0 \end{cases}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}, \tag{4.8}$$

and $\nu = \mu/\rho$ is the kinematic viscosity. The dynamic viscosity is computed from Sutherland's law:

$$\mu = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{1.5} \frac{T_{\text{ref}} + T_s}{T + T_s}.$$

The convective flux, diffusive flux and source term are

$$\mathcal{F}_c = \begin{bmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \otimes \mathbf{v} + p\mathbf{I} \\ \rho H \mathbf{v} \\ \rho \tilde{\nu} \mathbf{v} \end{bmatrix}, \quad \mathcal{F}_d = \begin{bmatrix} 0 \\ \boldsymbol{\tau}^{\text{rans}} \\ \boldsymbol{\tau}^{\text{rans}} \mathbf{v} + \kappa_T^{\text{rans}} \nabla T \\ \frac{1}{\sigma} \eta \nabla \tilde{\nu} \end{bmatrix},$$

$$\mathcal{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \rho \left(\tilde{P} - \tilde{D} + \frac{c_{b2}}{\sigma} \|\nabla \tilde{\nu}\|^2 \right) + \frac{1}{\sigma} \left(\nu + \tilde{\nu} \right) \nabla \rho \cdot \nabla \tilde{\nu} \end{bmatrix},$$

where $\eta = \mu(1 + f_n \chi)$ and the RANS shear stress tensor, $\boldsymbol{\tau}^{\text{rans}}$, and thermal conducvtivity, κ_T^{rans}, are given by

$$\boldsymbol{\tau}^{\text{rans}} = (\mu + \mu_t) \left[\nabla \mathbf{v} + (\nabla \mathbf{v})^t + \lambda \mathbf{I} \nabla \cdot \mathbf{v} \right],$$

$$\kappa_T^{\text{rans}} = c_p \left[\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t} \right],$$

with the specific heat $c_p = \gamma R/(\gamma - 1)$, the specific heat ratio, $\gamma = 1.4$ for air and Pr is the Prandtl number.

In the recent modifications of the SA model [2], $f_n(\chi)$ modifies the diffusion coefficient to account for $\tilde{\nu} < 0$ through

$$f_n = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}.$$

Similarly the modified production and destruction terms are

$$\tilde{P} = \begin{cases} c_{b1}(1-f_{t2})\tilde{S}\tilde{\nu}, & \tilde{\nu} \geq 0 \\ c_{b1}(1-c_{t3})S\tilde{\nu}, & \tilde{\nu} < 0 \end{cases}, \quad \tilde{D} = \begin{cases} \left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left[\frac{\tilde{\nu}}{d}\right]^2, & \tilde{\nu} \geq 0 \\ -c_{w1}\left[\frac{\nu}{d}\right]^2, & \tilde{\nu} < 0 \end{cases}$$

where the modified vorticity is computed, such that it remains positive, from

$$\tilde{S} = \begin{cases} S + \overline{S}, & \overline{S} \geq -c_{v2}S \\ S + \frac{S(c_{v2}^2 S + c_{v3}\overline{S})}{(c_{v3}-2c_{v2})S - \overline{S}}, & \overline{S} < -c_{v2}S \end{cases}, \quad \overline{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2},$$

where S is the magnitude of the vorticity $S = \|\nabla \times \mathbf{v}\|$ and d is the distance to the nearest wall. The remaining terms are given by

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, \quad f_w = g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6}, \quad g = r + c_{w2}(r^6 - r), \quad r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}$$

$$f_{t2} = c_{t3}\exp\left(-c_{t4}\chi^2\right)$$

and the constants used in this work are: $c_{b1} = 0.1355$, $\sigma = 2/3$, $c_{b2} = 0.622$, $\kappa = 0.41$, $c_{w1} = c_{b1}/\kappa^2 + (1+c_{b2})/\sigma$, $c_{w2} = 0.3$, $c_{w3} = 2$, $c_{v1} = 7.1$, $c_{v2} = 0.7$, $c_{v3} = 0.9$, $c_{t3} = 1.2$, $c_{t4} = 0.5$ and $Pr_t = 0.9$. Also note the omission of the trip terms in this work; for further details, the interested reader is referred to the recent work of Allmaras [2].

4.2.3 Zero pressure gradient turbulent flat plate

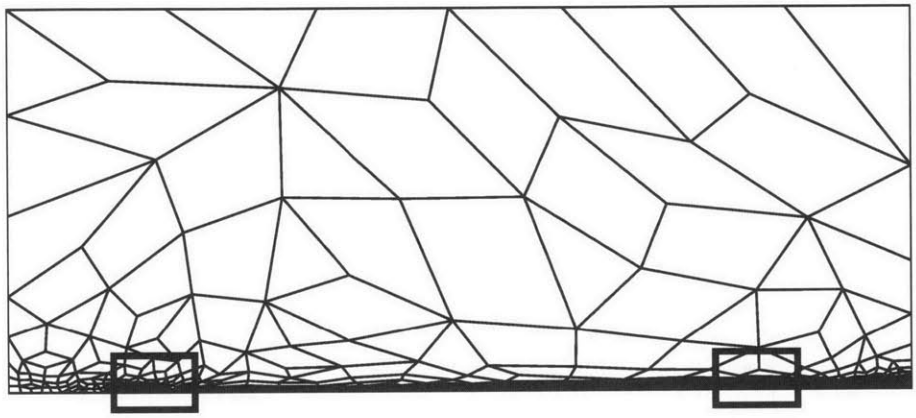
Methodology The subsonic flow over a flat plate is now computed; the freestream conditions are: $M_\infty = 0.2$, $\alpha = 0°$, $Re = 5 \times 10^6$ based on the plate length of 2 m. The total temperature and pressure are specified at the inlet whereas static pressure is specified on the upper and exit boundaries. An adiabatic no-slip condition is used on the plate, $x \in [0, 2]$. This test case is motivated by NASA's Turbulence Modeling Resource. Thirty adaptation iterations, starting with a NASA-provided quadrangulation with 875 nodes and 816 quadrilaterals, were first used to obtain L^∞- and L^2- optimized meshes at target degrees-of-freeom of 2,000, 6,000, 12,000 and 20,000 for both $p = 1$ and $p = 2$ discretisation orders. The mixed-element

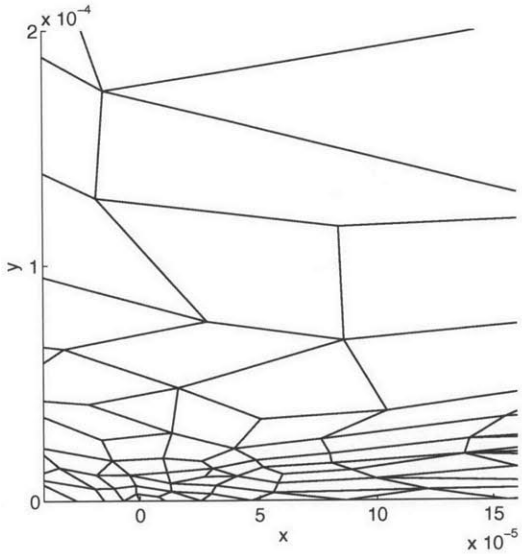
mesh adaptation algorithm was then restarted from the L^∞-optimized meshes for ten adaptation iterations; this restarting tactic is used to provide a good starting point for the mixed-element based adaptation. Again, the refinement factor is set to $r = 2$ for triangulations and $r = 3$ for mixed-element meshes for the same reasons given in the supersonic wedge case. The output of interest is the drag on the plate which is expected to converge at the rate $|J - J_{h,p}| \sim h^{2p}$ [49]. The reference drag coefficient was adaptively computed using a $p = 3$ discretisation with 100,000 degrees of freedom.

Results The 20,000 degree-of-freedom adapted meshes are shown in Figs. 4-14 and 4-16 along with magnified views of the leading edge and 90 % length section, where the mesh exhibits the highest anisotropy. A visual comparison of the leading edge section shows that the $p = 2$ mesh concentrates more elements near the leading edge. Similarly, the strength of the mesh grading at 90 % length is much stronger in the $p = 2$ case since the increased polynomial order better captures the smoothness in the boundary layer. The $p = 2$ algorithm thus has more elements at its disposal to cluster near the leading edge.

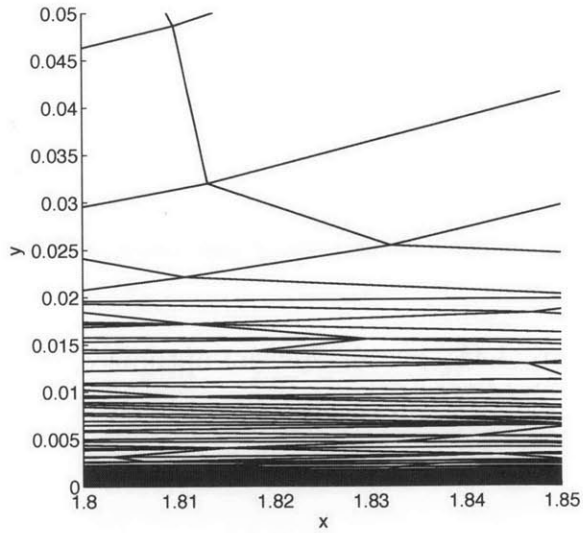
The error estimate of Fig. 4-18(a) converges at the optimal rate of h^{2p} whereas the exact error of Fig. 4-18(b) is polluted by the precision of the reference solution, which converged to within 7-8 digits. As such, these rates were computed using the first two dof-error pairs which still shows approximate h^{2p} convergence. Note also that the mixed-element meshes exhibit higher error estimates than the L^2 and L^∞ meshes. In fact, the L^∞-based adaptation appears to perform just as well as the L^2-based adaptation. In all cases, the $p = 2$ discretisation achieves a much lower error than the $p = 1$ case, despite using the same number of degrees of freedom.

For reference, the L^2 and L^∞ adapted triangulations are provided in Figs. 4-15 and 4-17 in which there is no recognizable difference with the adapted mixed-element meshes.



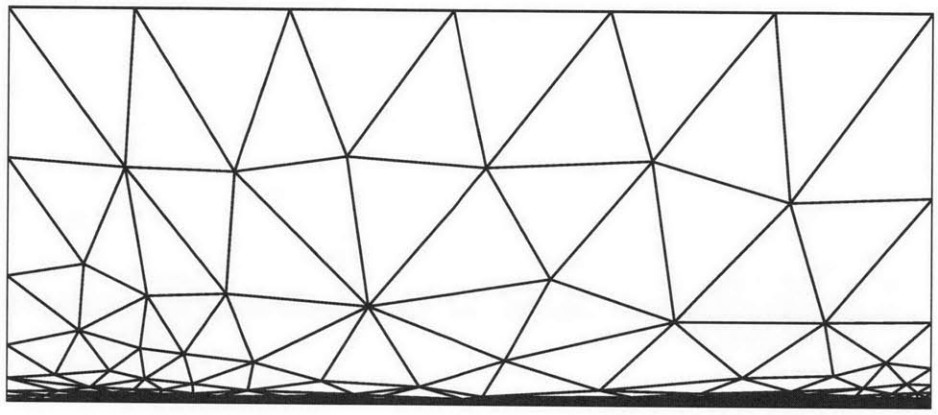


(b) Leading edge

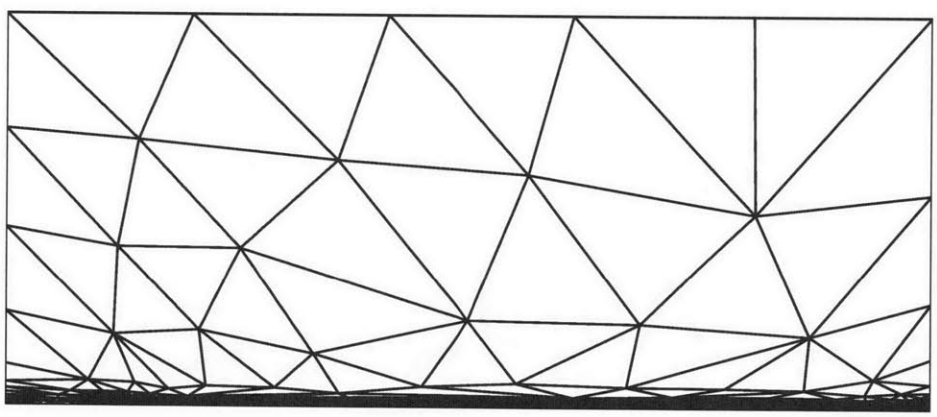


(c) 90% length

Figure 4-14: $p = 1$, 20k dof-optimized mixed-element mesh for the zero pressure gradient flat plate case

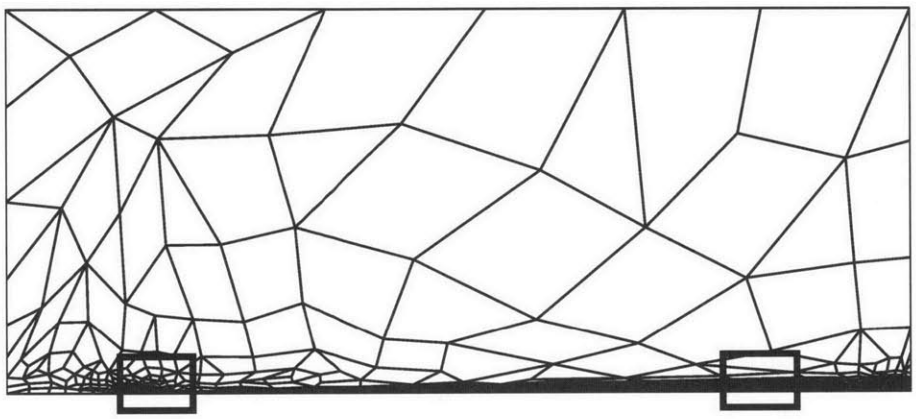


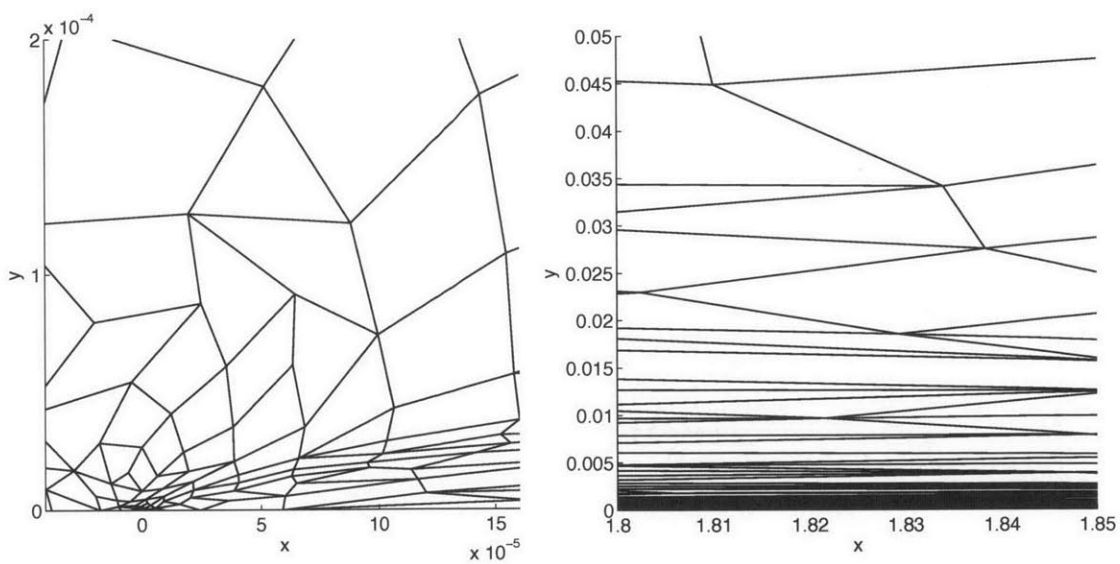
(a) L^2



(b) L^∞

Figure 4-15: $p = 1$, 20k dof-optimized triangulations for the zero pressure gradient flat plate case

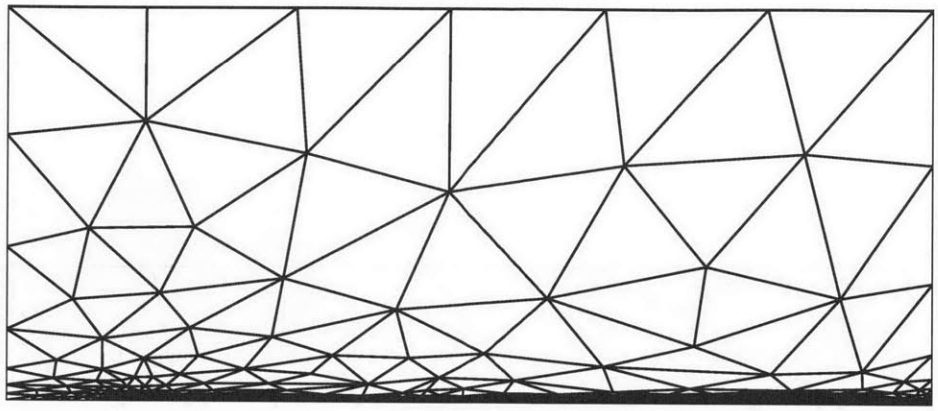




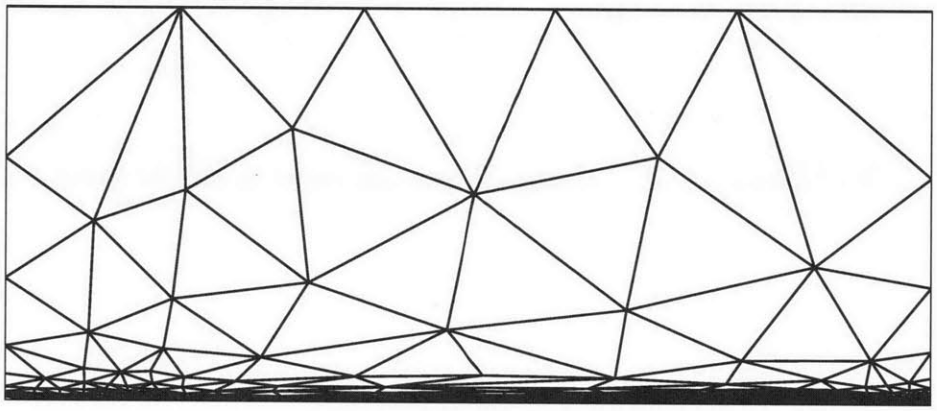
(b) Leading edge

(c) 90% length

Figure 4-16: $p = 2$, 20k dof-optimized mixed-element mesh for the zero pressure gradient flat plate case

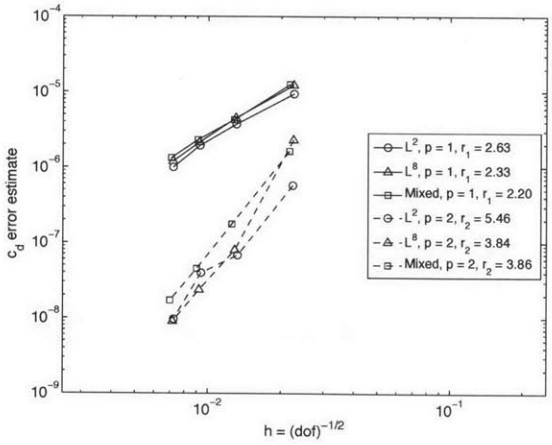


(a) L^2

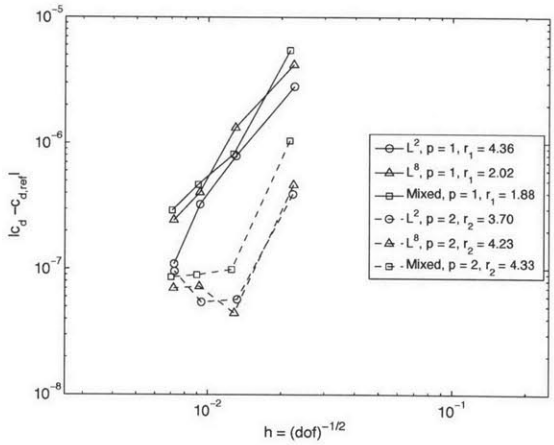


(b) L^∞

Figure 4-17: $p = 2$, 20k dof-optimized triangulations for the zero pressure gradient flat plate case



(a) c_d error estimate



(b) $|c_d - c_{d,\mathrm{ref}}|$

Figure 4-18: Convergence rates for $p = 1$ and $p = 2$ adaptations for the zero pressure gradient flat plate case with all element types

4.2.4 Turbulent transonic RAE2822 airfoil

Methodology Having studied problems independently exhibiting shock waves and boundary layers, the viscous flow over an RAE2822 airfoil at transonic flight conditions is now computed. The freestream conditions are: $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$. Riemann invariants to the freestream state are applied at the farfield whereas an adiabatic no-slip condition is used on the airfoil surface. This problem also tests the nonlinear elasticity method of Sec. 3.4 for curving the linear mesh output by the mesh generator; cubic elements are used throughout the mesh.

The L^2, L^∞ and mixed-element based mesh adaptation algorithms are applied to $p = 1$ and $p = 2$ solution orders. While the $p = 2$ meshes are optimized for a target degree-of-freedom count of 40,000, the $p = 1$ meshes are optimized for 45,000 degrees of freedom to better capture some of the solution features. Similar to the previous cases, the optimized mixed-element meshes are obtained by first running thirty adaptation iterations using the L^∞ mesh generation technique; this is followed by fifteen adaptation iterations using the mixed-element adaptation algorithm. Since the mixed-element based adaptation was started from an L^∞-adapted mesh, a refinement factor of $r = 3$ is used for mixed-elements whereas a factor of $r = 2$ is used for triangulations as in the previous cases. The obtained meshes and final errors are compared with the traditional L^2 mesh generation method. The output of interest, here, is the drag on the airfoil; the reference solution is taken from an adapted $p = 2$ discretisation with 100,000 degrees of freedom.

Results The meshes obtained with all element types and discretisation orders are shown in Fig. 4-19. The $p = 2$ meshes better capture the shock wave on the upper surface of the airfoil whereas the $p = 1$ meshes reach a compromise between adapting to the shock as well as the stem-like feature exhibited by the adjoint solution of Fig. 4-21(b). The $p = 1$ meshes also seem to cluster more elements in the boundary layer which is consistent with the subsonic flat plate results of the previous section.

Some properties of the adapted meshes are given in Tables 4.4 and 4.5 for $p = 1$ and $p = 2$ solution orders, respectively. The overall quad cover in both cases is about 50%, however, the local quad cover is lowest within the boundary layer where the element aspect ratios are highest. Again, MOESS with mixed-element meshes overshoots the requested degree-of-freedom count which explains why the error on the $p = 1$ mixed-element mesh is lower than the triangulations.

For the $p = 1$ discretisation, the final drag error estimate is about 10^{-5} on the triangulations whereas those for the $p = 2$ discretisation converge to an error estimate of 10^{-6}. Despite having fewer degrees of freedom at its disposal, the $p = 2$ based algorithm converges to a lower error than its $p = 1$ counterpart, demonstrating the benefits of performing high-order adaptive simulations.

Fig. 4-22(a) shows the convergence of the error estimate over the course of the mixed-element mesh adaptations. Note the dip in degrees of freedom as the mixed-element adaptations began in Fig. 4-22(b). This dip is more prominent in the $p = 1$ case which explains the larger error at the onset of the mixed-element adaptations of Fig. 4-22(a). Again observe the overshoot in the generated degrees of freedom in both $p = 1$ and $p = 2$ discretisations due to the inability to predict a priori the number of quadrilaterals created by the graph matching approach. This phenomenon motivates further improvements in the cost model to remedy this degree-of-freedom discrepancy since this directly affects the metric optimization problem.

The results of the L^∞-based adaptations, however, seem very promising. The final $p = 1$ and $p = 2$ L^∞ meshes with recombined quadrilaterals are shown in Fig. 4-20 which show finer resolution of both the shock and boundary layer. Properties of these meshes are also provided in Tables 4.4 and 4.5 under the L^∞-R row. The computed error for $p = 1$ and $p = 2$ recombined meshes, respectively, are 2.01×10^{-5} and 1.19×10^{-7}. Note the errors on these meshes are on the same order as the error in the original triangulations despite having a lower degree of freedom count after

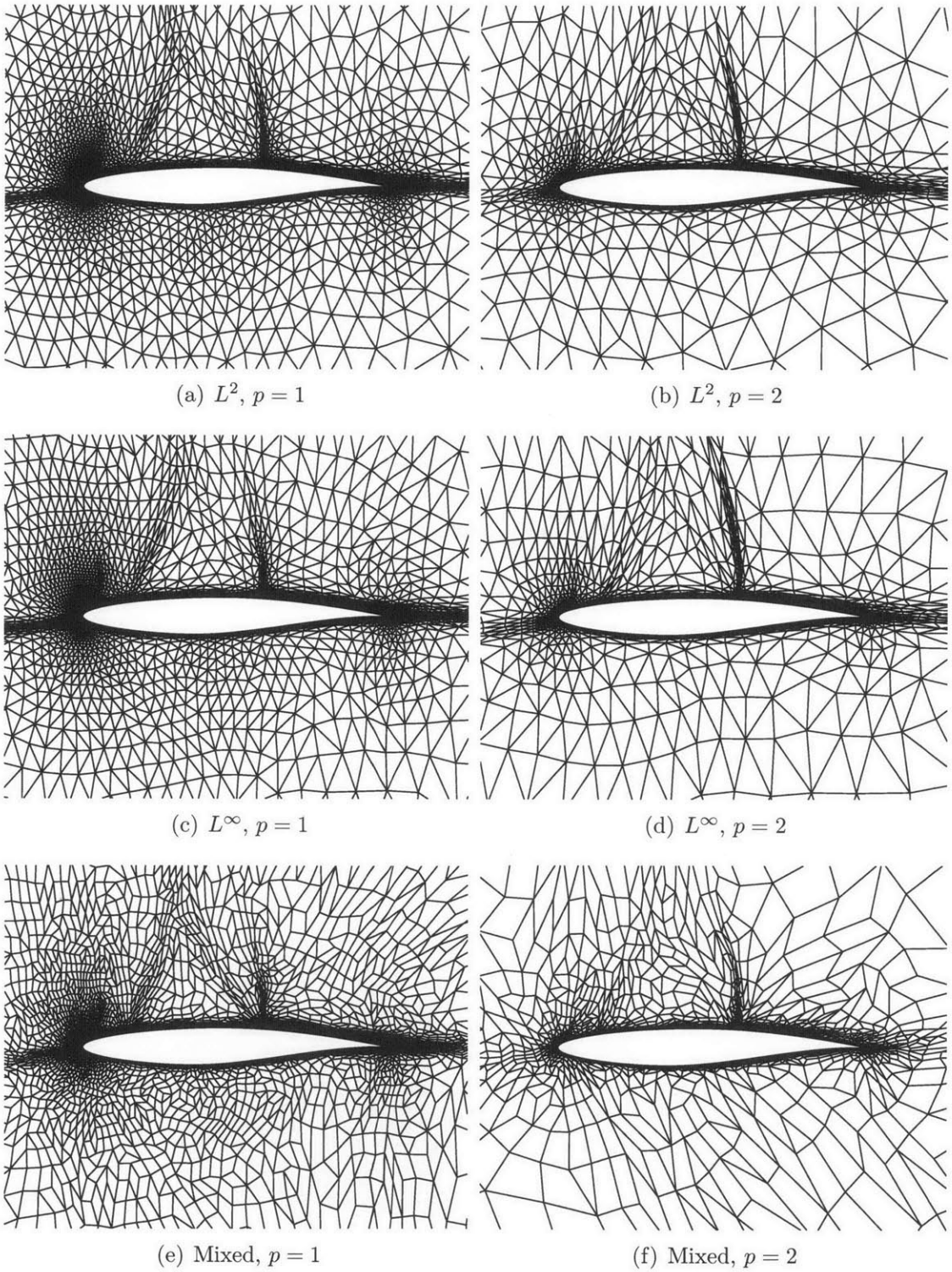
Table 4.4: Properties of the adapted meshes for the turbulent, transonic RAE2822 case, $p = 1$. L^∞-R stands for L^∞-Recombined.

Method	# quad	# tri	max(Æ)	dof	$\lvert c_d - c_{d,ref} \rvert$
L^2	0 (0%)	15,148 (100%)	2.01×10^3	45,444	1.50×10^{-5}
L^∞	0 (0%)	14,964 (100%)	3.87×10^3	44,892	1.11×10^{-5}
Mixed	6,892 (49%)	7,160 (51%)	1.22×10^3	49,048	2.11×10^{-6}
L^∞-R	3,869 (35%)	7,226 (65%)	3.88×10^3	37,154	2.01×10^{-5}

Table 4.5: Properties of the adapted meshes for the turbulent, transonic RAE2822 case, $p = 2$. L^∞-R stands for L^∞-Recombined.

Method	# quad	# tri	max(Æ)	dof	$\lvert c_d - c_{d,ref} \rvert$
L^2	0 (0%)	6,679 (100%)	8.51×10^2	40,074	5.34×10^{-7}
L^∞	0 (0%)	6,773 (100%)	9.09×10^2	40,638	1.01×10^{-7}
Mixed	2,944 (49%)	3,066 (51%)	1.06×10^3	44,892	8.61×10^{-6}
L^∞-R	1,731 (34%)	3,311 (66%)	9.09×10^2	35,446	1.19×10^{-7}

recombination. This demonstrates the potential of mixed-element based simulations; however, it also suggests work is needed in MOESS to improve the adaptation portion of the algorithm.



(a) L^2, $p = 1$

(b) L^2, $p = 2$

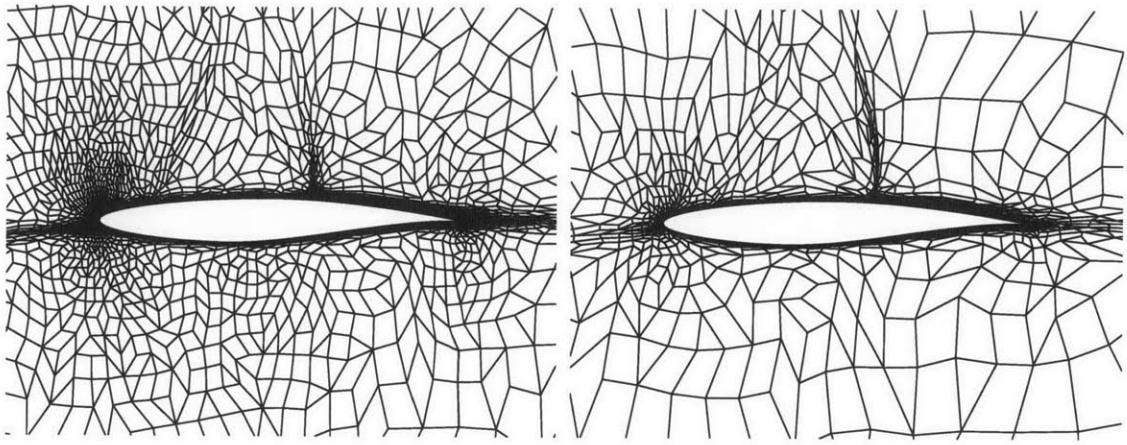
(c) L^∞, $p = 1$

(d) L^∞, $p = 2$

(e) Mixed, $p = 1$

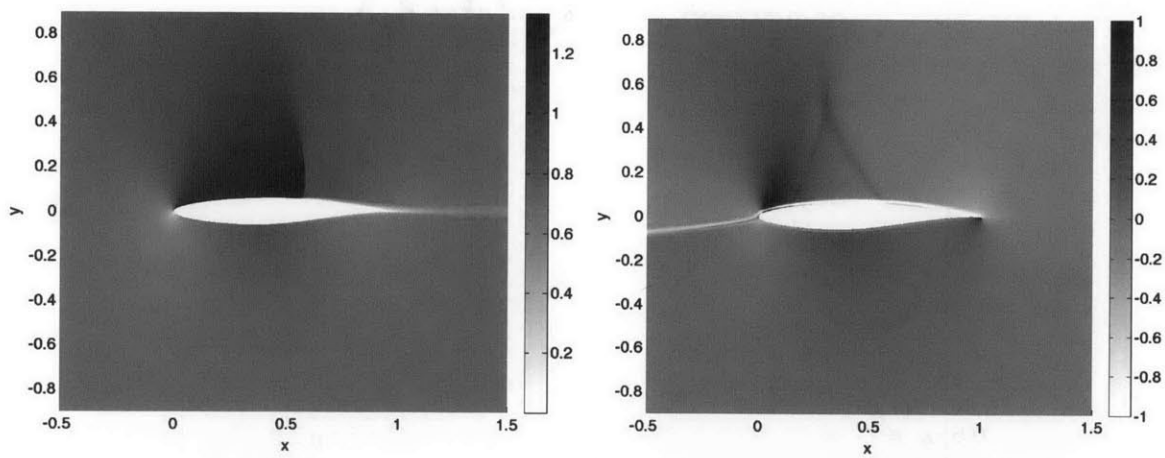
(f) Mixed, $p = 2$

Figure 4-19: Adapted meshes for the turbulent, transonic RAE2822 case at $p = 1$ and $p = 2$ solution orders



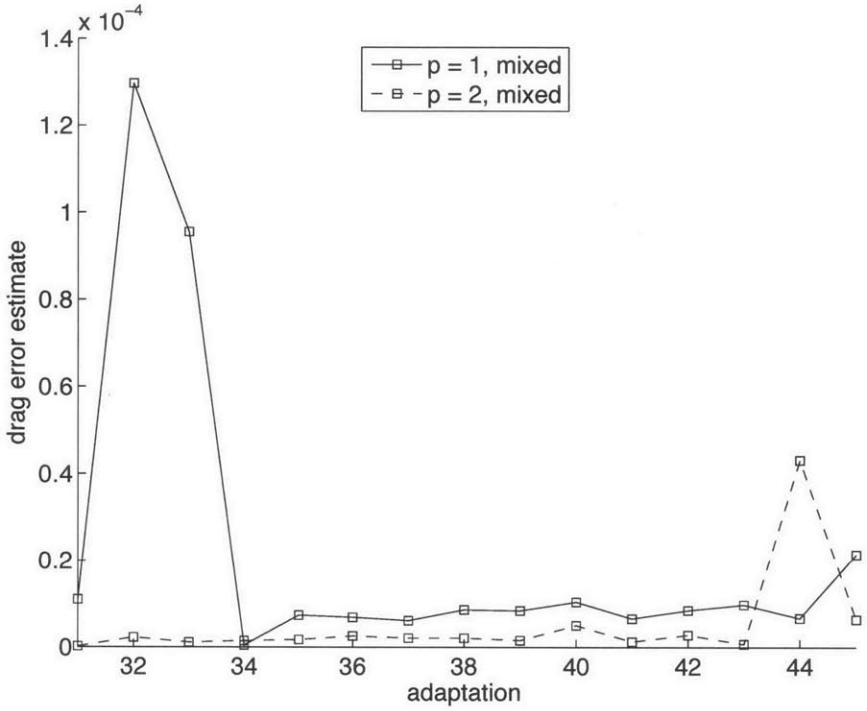
(a) $p = 1$ (b) $p = 2$

Figure 4-20: Recombined L^∞ meshes for the turbulent, transonic RAE2822 case at $p = 1$ and $p = 2$ solution orders

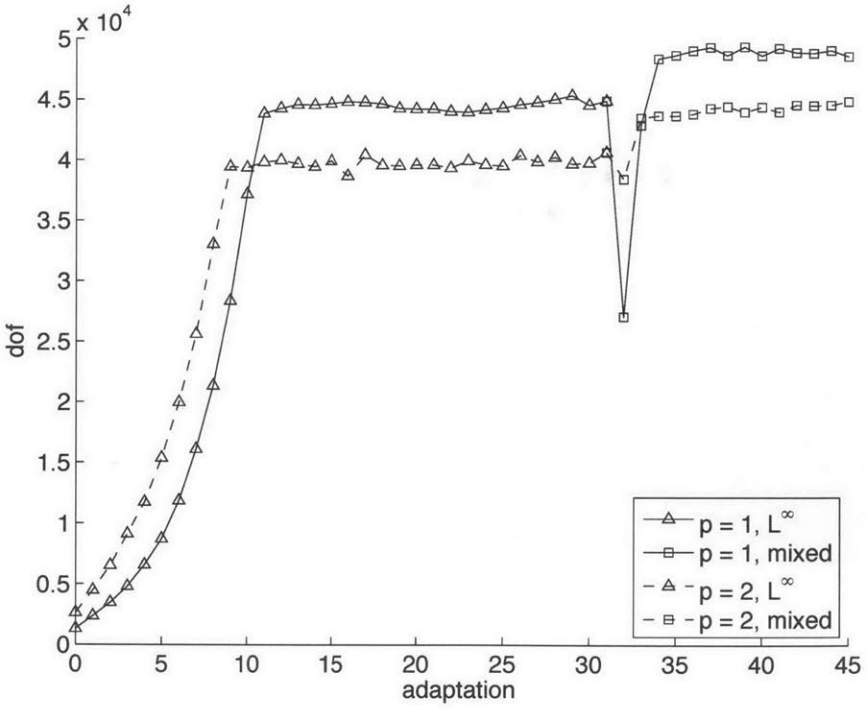


(a) Mach number (b) Mass adjoint

Figure 4-21: Mach number and mass adjoint for the turbulent, transonic RAE2822 case



(a) Error



(b) Degrees of freedom

Figure 4-22: Drag error estimate and degree-of-freedom convergence for the turbulent, transonic RAE2822 case

Chapter 5

Conclusions

5.1 Summary

This work investigated the use of mixed-element meshes in a mesh-adaptive solution process to high-order discretisations of partial differential equations. This required contributions in the fields of mesh adaptation and mesh generation. Mesh adaptation involved the extension of the Mesh Optimization via Error Sampling and Synthesis (MOESS) framework [58] to mixed-element meshes, which mostly involved modifications to the local error and cost models. As such, some preliminary work was done to improve the local error modeling procedure. The second fundamental component in this work is the development of a mixed-element, metric-driven mesh generator which was achieved by decomposing the original problem into that of a triangulation problem in which the L^∞ norm was used to generate right-triangles in the reference metric space. Right triangles were then combined to form quadrilaterals via a graph-matching approach. This method proved quite effective in generating metric-conforming meshes about complex geometries. Straight-sided meshes were then curved by employing a nonlinear elastic model to propagate surface displacements to the interior high-order geometry nodes.

The adaptation algorithm was first tested on L^2 error control problems to ensure an optimal mesh grading was obtained. First, an isotropic problem with a corner sin-

gularity demonstrated the ability of the algorithm to deduce the correct mesh grading using all element types. The convergence of the exact L^2 error proved to be more oscillatory in the mixed-element case than the equilateral (L^2) and right (L^∞) triangle cases for the $p = 3$ discretisation. In addition, the mixed-element size distribution displayed much more scatter about the regression line which is a result of the lower fidelity error model over quadrilateral elements which influences the optimized metric passed to the mesh generator. A two-dimensional boundary layer problem then revealed the mixed-element based adaptation algorithm necessitates a higher refinement factor to obtain the optimal anisotropy in the vicinity of the boundary layer. This is a result of the sample types used to construct the local error model. Some of the splits, notably those along the quadrilateral diagonals, cannot be considered as refinements since mesh edges are not split during the procedure. In fact, these samples increase the reference mesh size by the triangle inequality. As such, the correct behaviour of the local error as a function of the implied metric tensor is poorly captured.

Next, the algorithm was applied to select aerodynamic cases. First, an inviscid, supersonic problem demonstrated the ability to capture multiple shock waves at both $p = 1$ and $p = 3$ solution orders. Next, the flow over a flat plate at subsonic, turbulent conditions was considered. Optimal convergence rates of the error estimate for both $p = 1$ and $p = 2$ solution orders were observed in all adaptated mesh types. Finally, the ability of the algorithm to capture both shocks and boundary layers was tested in the transonic, turbulent flow over an RAE2822 airfoil using $p = 1$ and $p = 2$ solution orders. In general, the errors obtained on adapted mixed-element meshes were higher than those on triangulations; there was hardly a difference in the errors obtained with the L^∞- and L^2-based methods. The L^∞-based algorithm was effective in introducing structure to the highly anisotropic meshes which was observed to improve the convergence of the nonlinear solver, however, further work is needed to quantify these findings. An analysis similar to [55, 25] would be a good starting point.

While some interesting ideas may still be pursued in the field of mixed-element

based adaptation, the L^∞-based adaptation was impressive in maintaining the simplicity of simplex-based methods while improving solver robustness. The use of the L^∞ norm in three-dimensional applications is worth considering in the future.

5.2 Future work

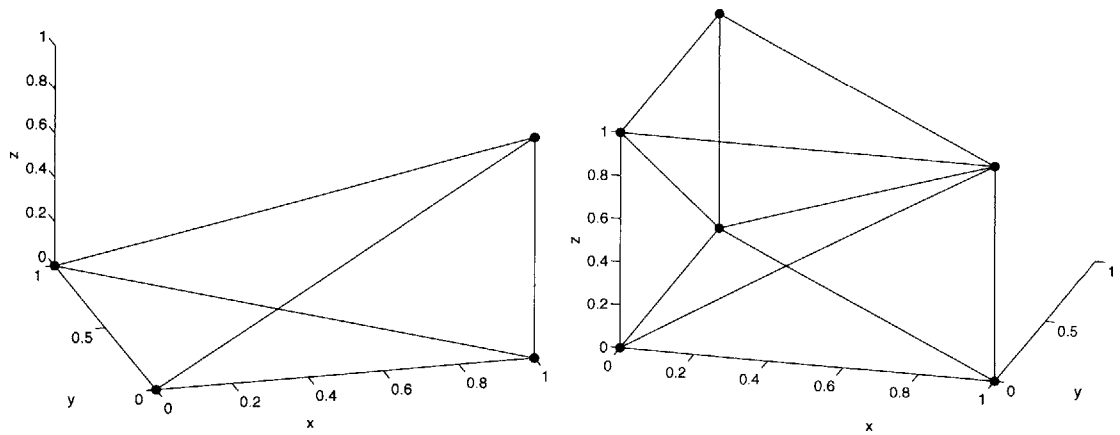
5.2.1 Improving the local error model

Throughout this work, comments were made about the local error and cost models. One possibility for improving the error model is to interpret the sampled data differently. Instead of averaging split element metrics, all split element data can be used to compute the rate tensor. Some preliminary work is given in Appendix A. Similarly, this data may be used to construct a higher-order error model.

5.2.2 Extension of the mesh generation algorithm to higher dimensions

The L^k norm introduced by Eq. (2.14) can still be used to generate right-simplicial meshes suitable for recombination into cubes. Setting $k = \infty$ in three dimensions allows the creation of the ideal tetrahedron shown in Fig. 5-1 (a). Three of these tetrahedra can be stacked to form the triangular-based prism of Fig. 5-1 (b). It can be shown that these tetrahedra are, in fact, identical and are merely affine transformations of one another.

Combining these tetrahedra into prisms falls into the realm of 3-uniform hypergraph matching, an NP-hard problem [31]. Lee et al. use reweighted random walks to compute an approximate solution to the hypergraph matching problem [31] and compare their proposed method with the tensor-matching approach of Duchenne et al. [14]. Once the triangular-based prisms are formed, they can be further combined into hexahedra using the same blossoming approach considered in this work.



(a) Ideal tetrahedron (b) Ideal prism

Figure 5-1: Ideal tetrahedron and triangular-based prism suitable for recombination into hexahedra

5.2.3 Mesh curving

This work employed a nonlinear elastic model to generate curved meshes. Another approach was initially investigated for generating curved meshes, based on the idea of optimizing the element Jacobians such that the resulting high-order elements have a positive Jacobian determinant throughout the element. The algorithm was introduced by Gargallo-Peiró and Roca [16, 57] and minimizes the distortion between the physical (curved) element and an ideal prescribed one, as shown in Fig. 5-2. The authors introduce a distortion measure which becomes infinite as the Jacobian tends to zero. To remedy this situation, they introduce a regularization parameter, used only when an invalid element is detected. This regularization parameter is dependent on the initial element aspect ratios of the input straight-sided mesh. For the highly anisotropic meshes considered in this work (aspect ratios as high as 10^6), the influence of this regularization parameter on the distortion measure vanishes which causes the optimizer to fail. A possible solution to this problem would be to use an adaptive precision library such as [51] to compute the regularized distortion measure exactly and avoid such errors.

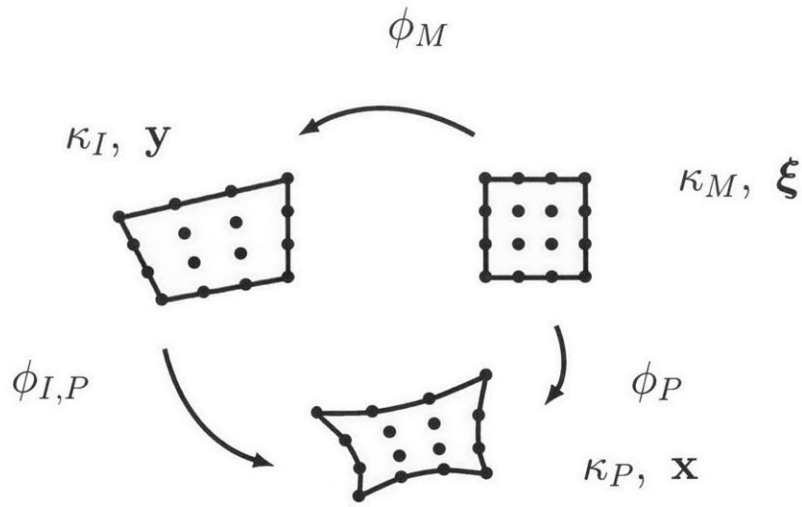


Figure 5-2: Mapping between ideal, master and physical elements for a $q = 3$ quadrilateral

5.2.4 Alternative discretisations

This work purely employed the discontinuous Galerkin method for solving partial differential equations. It is worth pursuing alternative discretisations, such as the hybridizable (HDG) or embedded (EDG) discontinuous Galerkin methods, as introduced in Chapter 1 and study the effect of element shape with those discretisations.

5.3 Concluding remarks

While the ability of the adaptive algorithm to generate adapted mixed-element meshes, the use of the L^∞ norm received considerable attention throughout this work due to its ability to introduce structure in problems with high anisotropy. In some cases, solver robustness *was* improved by the use of the L^∞-generated triangulations; however, further work is needed to investigate when and why this might be true.

On that note, we leave the reader with an L^∞-triangulation, generated from the isotropic metric field deduced from a grayscale image. Specifically, this mesh was generated from a picture of the original Ursa from which the developed mesh generator inherits its name.

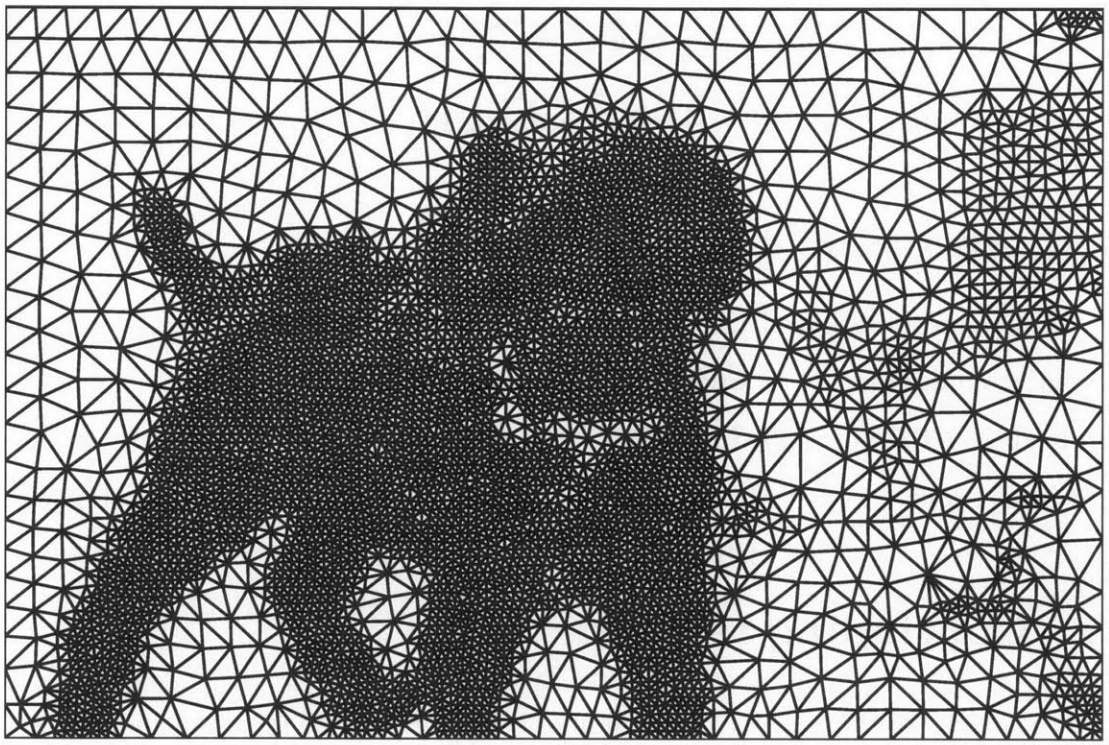


Figure 5-3: Mesh of Ursa generated by converting a grayscale image to an isotropic metric tensor field within a rectangular domain

Appendix A

Improving the local error model

A.1 Error modeling techniques

After sample metric-error pairs are taken, various methods can be used to interpret the data and construct a local error model over each element, κ_0, with implied metric, $\mathcal{M}_{\kappa_0}$ and original error, η_{κ_0}. This chapter outlines some preliminary work in improving this local error model to enhance the robustness of the adaptation algorithm.

Model 1: Average metric, accumulated errors The first method takes the implied metric of each element resulting from a split configuration and averages those to create a single implied metric of the configuration, $\overline{\mathcal{M}}_i$. The associated error of the split configuration is then the sum of the error contributions of each split element. Take for example, the first anisotropic split configuration of Fig. 2-2(b), with two split elements, κ_1 and κ_2, with associated metrics $\mathcal{M}_{\kappa_1}$ and $\mathcal{M}_{\kappa_2}$ and errors η_{κ_1} and η_{κ_2}. The implied metric of the configuration is $\overline{\mathcal{M}} = \text{mean}\{\mathcal{M}_{\kappa_1}, \mathcal{M}_{\kappa_2}\}$ and the error in the configuration is $\bar{\eta} = \eta_{\kappa_1} + \eta_{\kappa_2}$.

This method is advantageous because it agglomerates the samples into a single metric-error pair for each split configuration and, thus, reduces the size of the least-squares regression used to form the rate tensor. However, it may not capture the local error behaviour over the element since the individual contributions of the split

elements are masked by the single accumulated error.

Model 2: Sampled metric, weighted errors The second method considered here consists in using information from each split element obtained from each sample configuration. For example, with all split configurations from Figs. 2-2 and 2-3, there would be a total of 10 sample points for triangles and 12 for quadrilaterals. The implied metric of the sample is simply taken as the implied metric of the individual split element. However, the error invoked by the sample cannot simply be taken as the error computed over the split element. The error should be representative of the error induced by the discretisation over the *entire* domain. As a result, define a local *error density* induced by the split, $e_i = \eta_{\kappa_i}/|\kappa_i|$. Then the error induced in the entire domain is $\eta_{\kappa_i} = e_i|\kappa_0|$.

A.2 Error model rates

Each error model is now analyzed by observing the aggressiveness, or sensitivity of the model to changes in the mesh size. That is, in the context of the local error model,

$$\eta(\mathcal{S}) = \eta_0 \exp\left[\mathcal{R}_\kappa : \mathcal{S}\right], \tag{A.1}$$

a higher $\mathcal{R}_\kappa$ imposes more sensitivity of the error model to the step tensor, $\mathcal{S}$, and is, thus, more aggressive.

Without loss of generality, the analysis is reduced to a one-dimensional problem; consider a single split configuration over an element, as shown in Fig. A-1.

Figure A-1: $1d$ split configuration

For this one-dimensional case, both the rate and step tensors are reduced to scalar

quantities and the error model takes the form

$$\eta(s) = \eta_0 \exp\left[r_\kappa s\right]. \tag{A.2}$$

Since $s = -2\log(h/h_0)$, the error model can be expressed as

$$\eta(h) = \eta_0 \left(\frac{h}{h_0}\right)^{-2r_\kappa} \tag{A.3}$$

which provides a more intuitive interpretation of the relationship between the error and mesh size, as affected by the model rate.

Model 1 In order to evaluate the first error model, the rate, r_κ, needs to be computed. This first requires the average metric of the split elements.

Claim A.1. *The implied metric of the split configuration in Fig. A-1 is given by* $\overline{m} = \alpha^{-1}(1-\alpha)^{-1}h_0^{-2}$.

Proof. The implied metric tensors of each split element are clearly $m_1 = \alpha^{-2}m_0$ and $m_2 = (1-\alpha)^{-2}m_0$. The implied metric of the configuration is the minimum-distance tensor from both m_1 and m_2 in the affine-invariant space. That is

$$\overline{m} = \underset{m}{\operatorname{argmin}}\ d(m_1, m)^2 + d(m_2, m)^2$$

where the distance is $d(u,v) = \log(u^{-1/2}vu^{-1/2})$. For optimality,

$$\frac{\log[\alpha^2 m_0^{-1}m]}{m} + \frac{\log[(1-\alpha)^2 m_0^{-1}m]}{m} = \frac{\log[\alpha^2(1-\alpha)^2(m/m_0)^2]}{m} = 0 \tag{A.4}$$

which yields the solution $\overline{m} = \alpha^{-1}(1-\alpha)^{-1}m_0$. □

The total error on this split mesh is $\bar{\eta} = \eta_1 + \eta_2$ and the step from m_0 to $\overline{m}$ is $\bar{s} = -\log[\alpha(1-\alpha)]$. The error model rate is then

$$r_\kappa = -\frac{\log[(\eta_1+\eta_2)/\eta_0]}{\log[\alpha(1-\alpha)]} = -\frac{\log[z_1+z_2]}{\log\alpha(1-\alpha)} \tag{A.5}$$

with the definition $z_i \equiv \eta_i/\eta_0$.

Model 2 Here, the error model rate is computed by performing a linear regression of the area-weighted data obtained from both split elements:

$$r_\kappa = \underset{r}{\operatorname{argmin}}\ (\log[\bar{\eta}_1/\eta_0] - s_1 r_\kappa)^2 + (\log[\bar{\eta}_2/\eta_0] - s_2 r_\kappa)^2. \tag{A.6}$$

where $s_1 = -2\log\alpha$ and $s_2 = -2\log(1-\alpha)$ and the area-weighted errors are $\bar{\eta}_1 = \eta_1/\alpha$ and $\bar{\eta}_2 = \eta_2/(1-\alpha)$. Solving this yields the rate

$$r_\kappa = -\frac{1}{2}\frac{\log[z_1/\alpha]\log\alpha + \log[z_2/(1-\alpha)]\log(1-\alpha)}{\log^2\alpha + \log^2(1-\alpha)} \tag{A.7}$$

where z_i retains the same definition as above. This expression can be manipulated into

$$r_\kappa = -\frac{1}{2}\left[\frac{\log z_1 \log\alpha + \log z_2 \log(1-\alpha)}{\log^2\alpha + \log^2(1-\alpha)} - 1\right]. \tag{A.8}$$

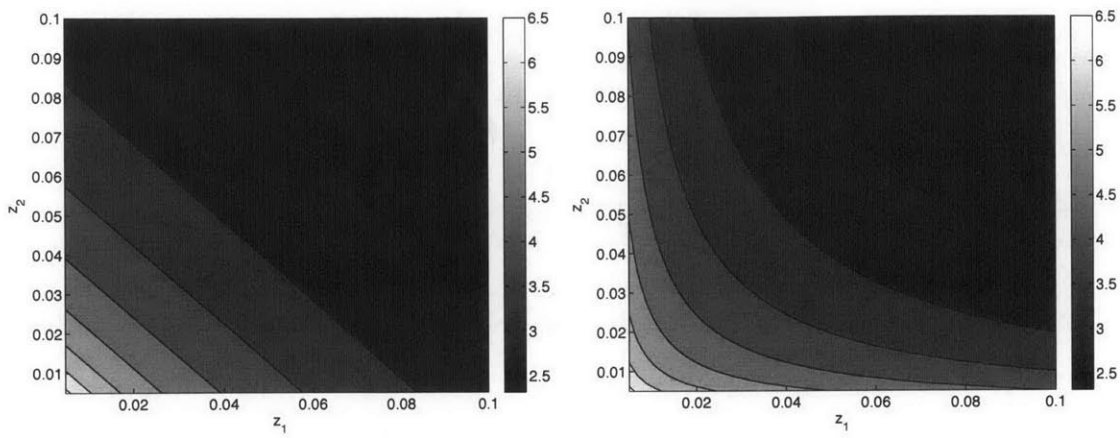
Note a slight issue with this model: in the event the error lies *exactly* within one split element and is zero in the other one, the model tends to infinity. However, in practice this never occurs since the error distribution within an element is *usually* well-distributed.

Comparison of both error models Since $0 \le \alpha \le 1$, assume $0 \le z_1, z_2 \le 1$; that is, the error always decreases in the direction of smaller elements.

Consider the $\alpha = .5$ case which is often the case in the MOESS sampling stage. The rates are

$$r_{\kappa,1} = \frac{\log(z_1+z_2)}{2\log 2} \qquad r_{\kappa,2} = \frac{1}{2}\left[1 + \frac{\log z_1 + \log z_2}{2\log 2}\right] \tag{A.9}$$

which are plotted over $[z_1, z_2] \in (0,1]^2$ in Fig. A-2. The second model better captures the higher rate needed from low error samples and can further direct the metric optimization in the direction of the associated implied metric of these samples.



(a) $-2r_{\kappa,1}$ (b) $-2r_{\kappa,2}$

Figure A-2: Effective exponent in error model of Eq. (A.3) obtained using both error modeling technqiues

A.3 Application to L^2 error control

As in Sec. 4.1, the L^2 error of an analytic function projected onto a function space is studied. The function in question is one-dimensional and is given by $u(x) = \sin(\pi x)$. Recall the projection problem is given by

$$\mathbf{u}_{h,p} = \underset{\mathbf{v}_{h,p} \in \mathcal{V}_{h,p}}{\operatorname{arginf}} \|\mathbf{u} - \mathbf{v}_{h,p}\|^2_{L^2(\Omega)} = \int_\Omega (\mathbf{u} - \mathbf{v}_{h,p})^2 \, \mathrm{d}\mathbf{x},$$

and the local error is

$$\eta_\kappa = \|\mathbf{u} - \mathbf{u}_{h,p}\|^2_{L^2(\kappa)} = \int_\kappa (\mathbf{u} - \mathbf{u}_{h,p})^2 \, \mathrm{d}\mathbf{x}.$$

In the context of the $1d$ error model of Eq. (A.3), optimal $p+1$ convergence of the exact error is observed for $\sqrt{\eta(h)}$. That is, the rates obtained with both methods introduced in this chapter should be compared with the exact error model as

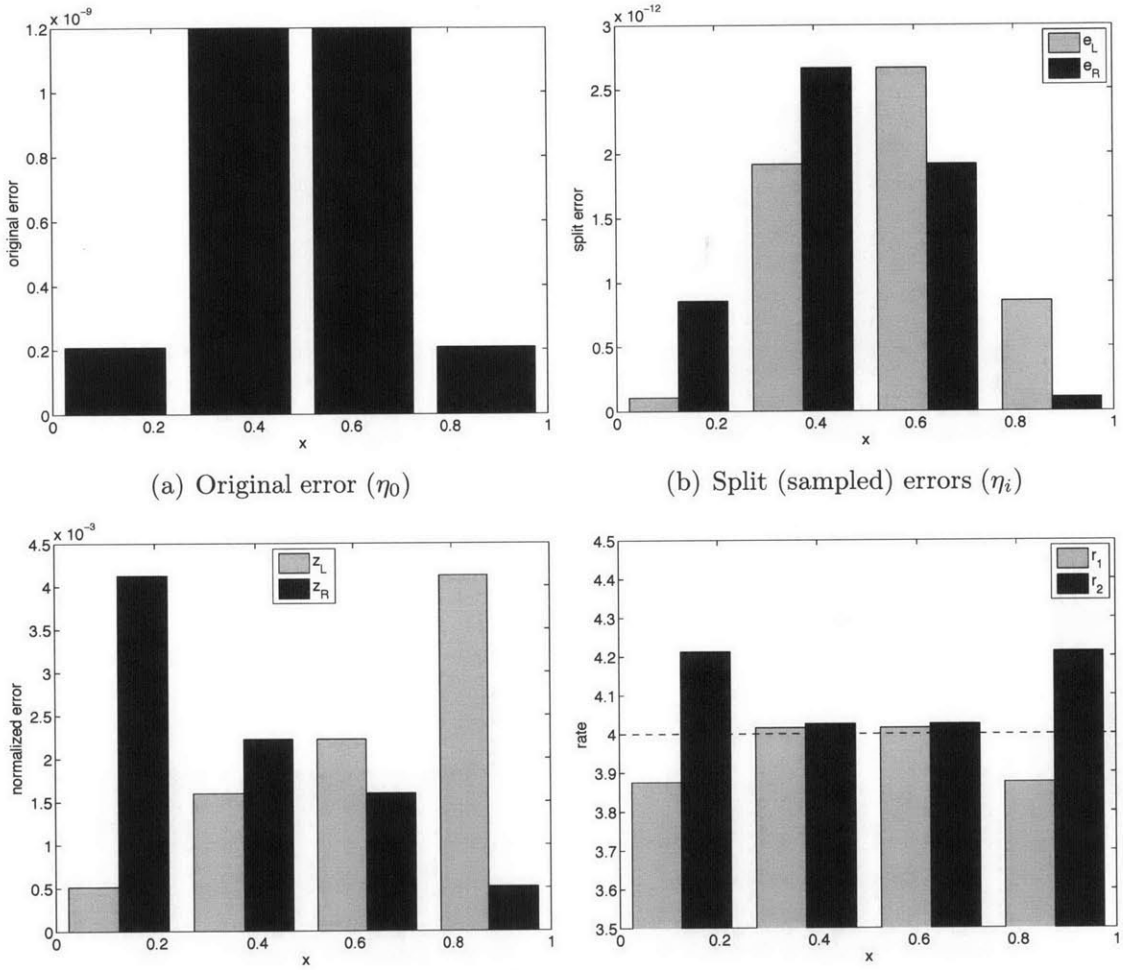
$$\eta(h) = \eta_0 \left(\frac{h}{h_0}\right)^{-r_\kappa} \quad \rightarrow \quad \eta(h) = \eta_0 \left(\frac{h}{h_0}\right)^{p+1}. \tag{A.10}$$

The purpose, here, is to first ensure both models can capture the asymptotic convergence rate of $p+1$. Next, the rates obtained in regions of low and high error will be

compared. The solution order is set to $p = 3$ and a uniform mesh in $\Omega \in [0, 1]$ is set up with 4 and 20 elements to study the behaviour on coarse and fine meshes. The element-wise errors and split errors are plotted across the domain. In addition, the rates obtained with both models are given for each element.

For clarity, the subscripts L and R refer to the left and right elements obtained during the error sampling stage. As shown in Fig. A-3(d), Method 2 always captures the highest of the normalized split errors (z_i) and produces a higher effective model rate. The same is true for the rates obtained with a fine discretisation of Fig. A-4. Note, however, that both models converge to the correct rate of $p + 1$ in the center of the domain, where the error is highest. This is more evident with a fine discretisation; a slight deviation from the asymptotic rate is observed on the coarse mesh. As expected from Fig. A-2, the second modeling procedure produces a higher rate in regions of high error.

In regions of low error, however, the model rates deviate from the asymptotic rate of $p + 1$. In the context of convergence, the adaptation algorithm should eventually stop optimizing the mesh. Model 1 essentially undershoots the optimal rate whereas Model 2 overshoots the rate. As such, Model 2 can yield additional refinement in these regions of low error, which can be problematic for the convergence over adaptation iterations. It is, however, important to note that Model 2 will detect high errors on the split elements whereas Model 1 only detects a cumulative error over the split configuration. This phenomenon can be important when adapting near boundaries. Additional work is needed to improve the local error model to achieve a compromise between the advantages of Models 1 and 2.



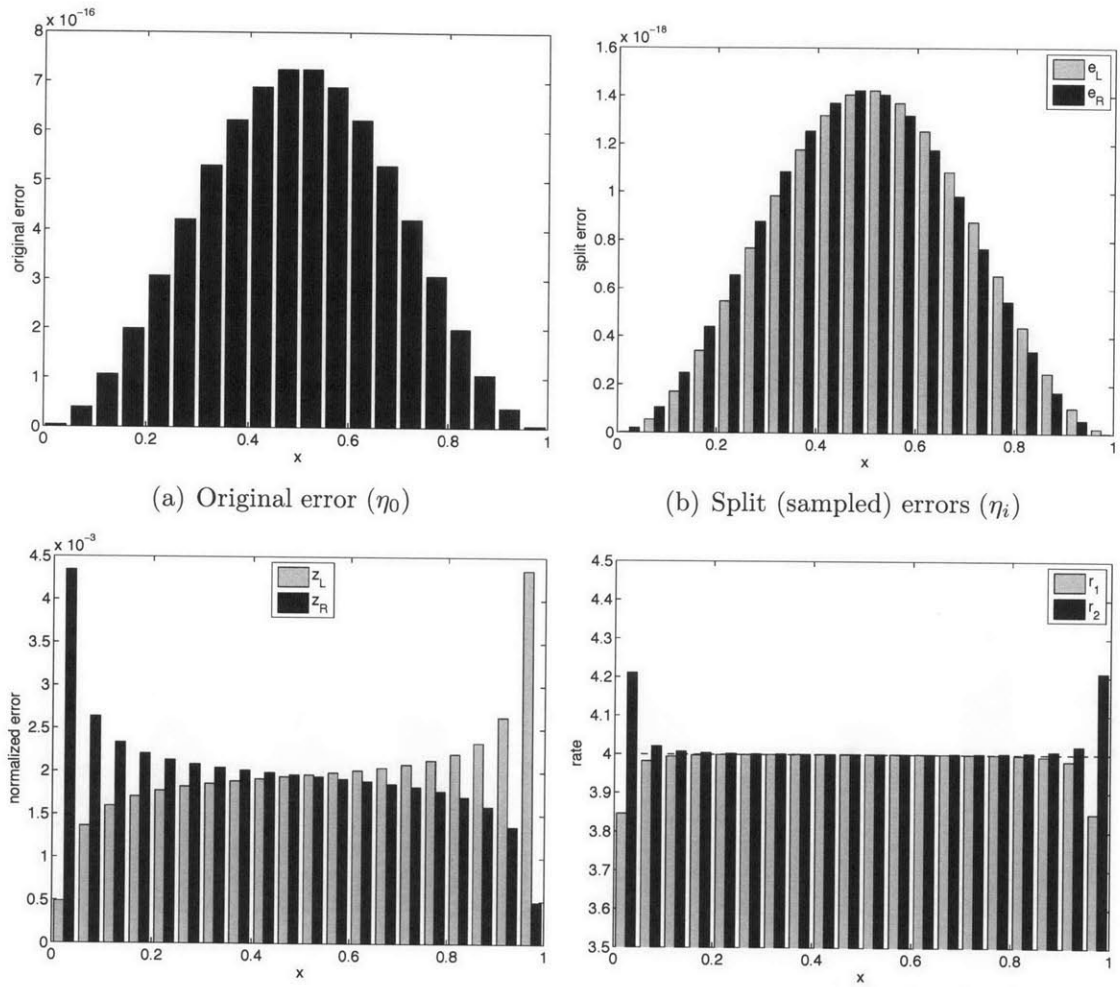
(a) Original error (η_0)

(b) Split (sampled) errors (η_i)

(c) Normalized split errors (z_i)

(d) Rates

Figure A-3: Errors and rates obtained with both models on a coarse mesh, $p = 3$, $n = 4$



(a) Original error (η_0)

(b) Split (sampled) errors (η_i)

(c) Normalized split errors (z_i)

(d) Rates

Figure A-4: Errors and rates obtained with both models on a fine mesh, $p = 3$, $n = 20$

Appendix B

Connectivity tables for the embedded discontinuous Galerkin method

This chapter provides the connectivity tables used to compute the number of non-zero entries in the Jacobian matrix for the embedded discontinuous Galerkin (EDG) method, given in Tables 1.1 and 1.3. Recall the factors $l_{i,d}$ of Table 1.2 represent the local number of d-dimensional entities attached to an i-dimensional one.

Table B.1: Connectivity factor, $c_{i,j}$, for triangles using the EDG method

$i \backslash j$	0	1
0	$l_{0,0} + l_{0,1}$	$l_{0,1} + l_{0,2}$
1	$l_{1,0} + l_{1,2}$	$l_{1,1} + 2l_{1,2}$

Table B.2: Connectivity factor, $c_{i,j}$, for quadrilaterals using the EDG method

$i \backslash j$	0	1
0	$l_{0,0} + l_{0,1} + l_{0,2}$	$l_{0,1} + 2l_{0,2}$
1	$l_{1,0} + 2l_{1,2}$	$l_{1,1} + 3l_{1,2}$

Table B.3: Connectivity factor, $c_{i,j}$, for tetrahedra using the EDG method

$i \backslash j$	0	1	2	3
0	$l_{0,0} + l_{0,1}$	$l_{0,1} + l_{0,2}$	$l_{0,2} + l_{0,3}$	0
1	$l_{1,0} + l_{1,2}$	$l_{1,1} + 2l_{1,2} + l_{1,3}$	$l_{1,2} + 2l_{1,3}$	0
2	$l_{2,0} + l_{2,3}$	$l_{2,1} + 3l_{2,3}$	$l_{2,2} + 3l_{2,3}$	0
3	0	0	0	0

Table B.4: Connectivity factor, $c_{i,j}$, for hexahedra using the EDG method

$i \backslash j$	0	1	2	3
0	$l_{0,0} + l_{0,1} + l_{0,2} + l_{0,3}$	$l_{0,1} + 2a_{0,2} + 3l_{0,3}$	$l_{0,2} + 3l_{0,3}$	0
1	$l_{1,0} + 2l_{1,2} + 2l_{1,3}$	$l_{1,1} + 3l_{1,2} + 5l_{1,3}$	$l_{1,2} + 4l_{1,3}$	0
2	$l_{2,0} + 4l_{2,3}$	$l_{2,1} + 8l_{2,3}$	$l_{2,2} + 5l_{2,3}$	0
3	0	0	0	0

Bibliography

[1] Frederic Alauzet and David Marcum. Anisotropic goal-oriented mesh adaptation for time dependent problems. In Josep Sarrate and Matthew Staten, editors, *Proceedings of the 22nd International Meshing Roundtable*, pages 241–261. Springer Berlin Heidelberg, 2013.

[2] Steven R. Allmaras, Forrester T. Johnson, and Philippe R. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. Seventh international conference on computational fluid dynamics, Big Island, Hawaii, 2012.

[3] Garrett E. Barter. *Shock Capturing with PDE-Based Artificial Viscosity for an Adaptive, Higher-Order, Discontinuous Galerkin Finite Element Method.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2008.

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

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

[6] Ted D. Blacker and Michael B. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32(4):811–847, 1991.

[7] H. Borouchaki, P. George, F. Hecht, P. Laug, and E Saltel. Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes. INRIA-Rocquencourt, France. Tech Report No. 2741, 1995.

[8] Houman Borouchaki, Pascal, J. Frey, and Key Words. Adaptive triangular-quadrilateral mesh generation. *Intl. J. Numer. Methods Eng*, pages 915–934, 1998.

[9] Frank J. Bossen and Paul S. Heckbert. A pliant method for anisotropic mesh generation. In *5th Intl. Meshing Roundtable*, pages 63–74, Oct. 1996.

[10] Marco Ceze and Krzysztof J. Fidkowski. Output-driven anisotropic mesh adaptation for viscous flows using discrete choice optimization. AIAA 2010–170, 2010.

[11] William Cook. The blossom iv code for min-weight perfect matching, available at: `http://www.math.uwaterloo.ca/ bico//blossom4/`, 1999.

[12] William Cook and Andre Rohe. Computing minimum-weight perfect matchings. *INFORMS J. on Computing*, 11(2):138–148, February 1999.

[13] C. Dobrzynski and P. Frey. Anisotropic delaunay mesh adaptation for unsteady simulations. In *17th Intl. Meshing Roundtable*, pages 177–194, Oct 2008.

[14] Olivier Duchenne, Francis Bach, In-So Kweon, and Jean Ponce. A tensor-based algorithm for high-order graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(12):2383–2395, December 2011.

[15] Ko foa Tchon and Ricardo Camarero. Quad-dominant mesh adaptation using specialized simplicial optimization. In *In Proc. 15th International Meshing Roundtable*, pages 21–38, 2006.

[16] Abel Gargallo-Peiro, Xevi Roca, and Josep Sarrate. A surface mesh smoothing and untangling method independent of the cad parametrization. *Comput. Mech.*, 2013.

[17] Emmanuil H. Georgoulis, Edward Hall, and Paul Houston. Discontinuous Galerkin methods for advection-diffusion-reaction problems on anisotropically refined meshes. *SIAM J. Sci. Comput.*, 30(1):246–271, 2007.

[18] Emmanuil H. Georgoulis, Edward Hall, and Paul Houston. Discontinuous Galerkin methods on *hp*-anisotropic meshes II: A posteriori error analysis and adaptivity. *Appl. Numer. Math.*, 59:2179–2194, 2009.

[19] Christophe Geuzaine and Jean-Franois Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[20] R. Hartmann. Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 51:1131–1156, 2006.

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

[22] Ralf Hartmann and Paul Houston. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. *J. Comput. Phys.*, 183(2):508–532, 2002.

[23] Ralf Hartmann and Paul Houston. Error estimation and adaptive mesh refinement for aerodynamic flows. In H. Deconinck, editor, *VKI LS 2010-01: 36th CFD/ADIGMA course on hp-adaptive and hp-multigrid methods, Oct. 26-30, 2009*. Von Karman Institute for Fluid Dynamics, Rhode Saint Genèse, Belgium, 2009.

[24] F. Hecht. Bamg: Bidimensional anisotropic mesh generator, 1998. `http://www-rocq1.inria.fr/gamma/cdrom/www/bamg/eng.htm`.

[25] David L. Darmofal Huafei Sun and Robert Haimes. On the impact of triangle shapes for boundary layer problems using high-order finite element discretization. *J. Comput. Phys.*, 231:541–557, 2012.

[26] Antonio Huerta, Aleksandar Angeloski, Xevi Roca, and Jaime Peraire. Efficiency of high-order elements for continuous and discontinuous galerkin methods. *International Journal for Numerical Methods in Engineering*, 96(9):529–560, 2013.

[27] Steven G. Johnson. The NLopt nonlinear-optimization package, available at: `http://ab-initio.mit.edu/nlopt`.

[28] Kudo Jun. Robust adaptive high-order rans methods. Master's thesis, Mass. Inst. of Tech., CDO, 2014.

[29] Bryan Matthew Klingner and Jonathan Richard Shewchuk. Agressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, Seattle, Washington, October 2007.

[30] Denis Kovacs, Ashish Myles, and Denis Zorin. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 137–146, New York, NY, USA, 2010. ACM.

[31] Jungmin Lee, Minsu Cho, and Kyoung Mu Lee. Hyper-graph matching via reweighted random walks. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1633–1640, Washington, DC, USA, 2011. IEEE Computer Society.

[32] Y. K. Lee and C. K. Lee. A new indirect anisotropic quadrilateral mesh generation scheme with enhanced local mesh smoothing procedures. *International Journal for Numerical Methods in Engineering*, 58(2):277–300, 2003.

[33] A. Loseille, A. Dervieux, and F. Alauzet. Fully anisotropic goal-oriented mesh adaptation for 3d steady Euler equations. *J. Comput. Phys.*, 229:2866–2897, 2010.

[34] Adrien Loseille and Frédéric Alauzet. Continuous mesh model and well-posed continuous interpolation error estimation. INRIA RR-6846, 2009.

[35] Adrien Loseille and Frédéric Alauzet. Optimal 3D highly anisotropic mesh adaptation based on the continuous mesh framework. In *Proceedings of the 18th International Meshing Roundtable*, pages 575–594. Springer Berlin Heidelberg, 2009.

[36] Adrien Loseille and Frédéric Alauzet. Continuous mesh framework part I: Well-posed continuous interpolation error. *SIAM J. Numer. Anal.*, 49(1):38–60, 2011.

[37] Adrien Loseille and Frédéric Alauzet. Continuous mesh framework part II: Validations and applications. *SIAM J. Numer. Anal.*, 49(1):61–86, 2011.

[38] Adrien Loseille and Rainald Löhner. On 3d anisotropic local remeshing for surface, volume and boundary layers. In *Proceedings of the 18th International Meshing Roundtable*, pages 611–630. Springer Berlin Heidelberg, 2009.

[39] Adrien Loseille and Rainald Löhner. Anisotropic adaptive simulations in aerodynamics. AIAA 2010-169, 2010.

[40] Bruno Lvy and Nicolas Bonneel. Variational anisotropic surface meshing with voronoi parallel linear enumeration. 2013.

[41] Bruno Lvy and Yang Liu. Lp centroidal voronoi tesselation and its applications. *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*, 2010. patent pending - FR 10/02920 (filed 07/09/10).

[42] Todd Michal and Joshua Krakos. Anisotropic mesh adaptation through edge primitive operations. AIAA 2012-159, 2012.

[43] S. J. Owen, M. L. Staten, S. A. Canann, and S. Saigal. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 44(9):1317–1340, 1999.

[44] Steven J. Owen and Sunil Saigal. H-morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1-2):289–312, 2000.

[45] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *Int. J. Comput. Vision*, 66(1):41–66, 2006.

[46] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *J. Comput. Phys.*, 72:449–466, 1987.

[47] Per-Olof Persson and Jaime Peraire. Curved mesh generation and mesh refinement using Lagrangian solid mechanics. AIAA 2009-0949, 2009.

[48] Les Piegl and Wayne Tiller. The nurbs book, 1997.

[49] Niles A. Pierce and Michael B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Rev.*, 42(2):247–264, 2000.

[50] J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Bchet, E. Marchandise, C. Geuzaine, and T. Mouton. A frontal delaunay quad mesh generator using the lnorm. *International Journal for Numerical Methods in Engineering*, 94(5):494–512, 2013.

[51] Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. Technical report, Carnegie Mellon University, 1996.

[52] Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In Ming C Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, pages 203–222. Springer-Verlag, 1996.

[53] Hang Si. Tetgen: A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator. Weierstrass Institute for Applied Analysis and Stochastics, 2005. `http://tetgen.berlios.de`.

[54] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences. CRDL, 2013.

[55] Huafei Sun. Impact of triangle shapes using high-order discretizations and direct mesh adaptation for output error. Master's thesis, Massachusetts Institute of Technology, Computation for Design and Optimization Program, 2009.

[56] Huafei Sun. *A Robust Simplex Cut-Cell Method for Adaptive High-Order Finite Element Discretizations of Aerodynamics and Multi-Physics Problems.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, August 2013.

[57] Abel Gargallo-Perio Xevi Roca and Josep Sarrate. Defining quality measures for high-order planar triangles and curved mesh generation. In *Proceedings of the 20th International Meshing Roundtable*, pages 265–383. Springer Berlin Heidelberg, 2011.

[58] Masayuki Yano. *An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2012.

[59] Masayuki Yano and David Darmofal. An optimization framework for anisotropic simplex mesh adaptation: Application to aerodynamic flows. AIAA 2012–0079, January 2012.

[60] Masayuki Yano and David L. Darmofal. An optimization-based framework for anisotropic simplex mesh adaptation. *J. Comput. Phys.*, 231(22):7626–7649, September 2012.

[61] Masayuki Yano, James M. Modisette, and David Darmofal. The importance of mesh adaptation for higher-order discretizations of aerodynamic flows. AIAA 2011–3852, June 2011.