Adaptive Mesh Refinement Based on a Posteriori Error Estimation

by

Martin Juhas

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Aerospace Engineering
University of Toronto

Copyright © 2014 by Martin Juhas

Abstract

Adaptive Mesh Refinement Based on a Posteriori Error Estimation

Martin Juhas

Master of Applied Science

Graduate Department of Aerospace Engineering

University of Toronto

2014

This thesis describes the application of a h-refinement, adjoint-based, *a posteriori* error estimation strategy with a solution-driven, block-based adaptive mesh refinement (AMR), finite-volume scheme to the solution of advection-diffusion problems in two dimension. Error-based criteria following from the adjoint-weighted computable correction of a functional, error in the computable correction for the functional, using either primal, dual, or both error formulations, and/or the total estimated error in the functional are all used to direct the mesh refinement in the block-based AMR scheme. The test cases considered here for the advection-diffusion equation illustrate the ability of the error-based AMR criteria to improve the accuracy of integrated functionals where physics-based AMR criteria may fail. The performance of the adjoint-based refinement criteria is compared with both gradient-based and uniform mesh refinement strategies.

Acknowledgements

I would like to use this opportunity to express my gratitude to my family for supporting me and giving me the opportunity to study and learn.

I would also like to thank my supervisor Dr. Groth for the chance to work in his group and for his financial support during my stay at University of Toronto.

For his support in programming and code development, I thank Dr. Scott Northrup.

Martin Juhas

Toronto, August 2014

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The importance of computational fluid dynamics (CFD) lies in its utility as a tool for scientific research and as an engineering design aid. Numerical methods can help guide the development of complex engineering designs from the initial stage to the final prototype.

As the global population increases, while our natural resources are constant, or being continually depleted, it is important for humanity to be able to increase the efficiency of all the devices that we use. This requires a greater understanding of basic physical processes and development of advanced design tools to meet the need for a sustainable development and social and environmental responsibility.

Numerical methods help design devices that are ever more efficient and play a crucial role in reducing the environmental foot print left by society. For example, finite element methods help designers produce cars that use fewer structural parts, while the safety standards improved to previously unseen level. CFD affects the shapes of all devices that move though fluid. Ships, submarines, cars, aircraft, helicopters, sporting goods, even clothes are designed to move though fluids efficiently. This means that CFD is directly responsible for lower energy needs of new transportation devices. The numerical simulations of combustion enhance the design of combustion engines. This allows new engines to operate more efficiently, produce lower emissions, operate more quietly and have more compact dimensions. The efficiency increases in all of the mentioned areas help

reduce energy needs of each individual, while making the production of usable energy more efficient. Thus, the need of natural resources per capita can be reduced.

The prediction of fluid flows and combustion processes occurring in real-world problems are currently very expensive using numerical methods available at the moment. The problems studied by scientists and engineers range from large domain problems of complex atmospheric processes on a planetary scale, to supersonic, turbulent, multi-phase, combustion flows in rocket engines to micro scale flows encountered by micro-electronic devices. The physical processes in such flows are complex and present a great challenge for scientists trying to produce efficient and accurate numerical simulations that describe such flows.

One way of obtaining more accurate results of complex numerical simulations is to use more computational resources. This approach is often financially too expensive, impractical or even impossible depending on the numerical problem, numerical algorithm and its implementation and hardware available. To solve these issues, there is an effort for the development of numerical methods that aims to provide useful engineering tools, which can use the existing computational resources, to produce better simulation results within an acceptable time frame. An ideal numerical algorithm would use most of the hardware resources available to solve the problem as efficiently as possible. Also, it is important for the algorithm to be flexible enough to adapt to new and coming technological changes in hardware design.

The trend in recent years has been to use parallel, multi-processor computational platforms with distributed memory for solving numerical problems [1, 2]. This hardware constraint requires an efficient numerical method that is highly parallel and can be easily scaled. Any efficient and fast CFD algorithm must have an efficient algorithm for generating an appropriate numerical domain in which the problem can be solved. The accuracy of CFD algorithm is highly dependant on the quality of the numerical domain. Also, manual mesh generation is typically a very time consuming process that doesn't guarantee efficient use of computational resources, nor can it ensure that the solution will be accurate enough and further mesh adjustment will not be needed. In light of these issues with the traditional approach of having a separate mesh generation process and a separate CFD solver, it can be seen that an integral, automatic mesh generation algo-

rithm working with the CFD solver would make the algorithm much more user-friendly and time-effective. In addition, by using adaptive mesh refinement (AMR), it is also possible to reduce the overall numerical domain while providing better solution accuracy [2–6]. This can be achieved for flows where the solution quality depends on features with a large variation of spatial scales. This thesis aims to describe one possible means to guide the AMR algorithm so as to better predict a values of a solution-dependent functional of particular interest.

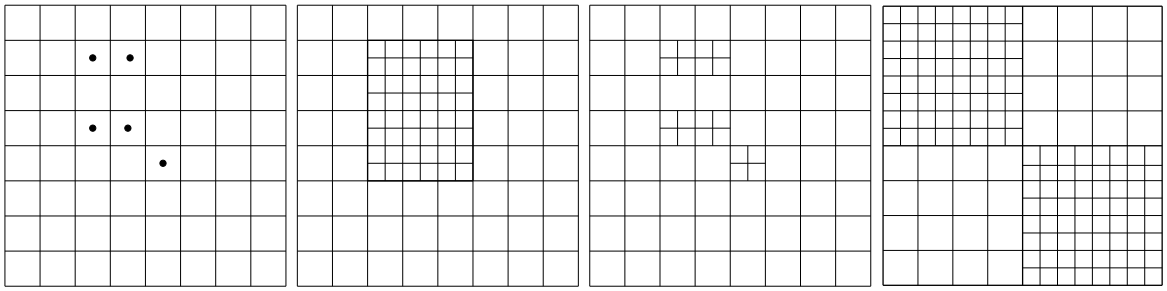
1.2 Parallel Adaptive Mesh Refinement

1.2.1 Adaptive Mesh Refinement Techniques

The effort of reducing the computational cost of solving the desired numerical problem naturally brings attention to the computational domain and its size. If the domain becomes smaller, the problem can be solved more quickly, or a larger problem can be solved using the same resources. Therefore, an algorithm that can better utilise the information at hand would be more efficient at providing an accurate solution. There is number of ways to make it possible to use the solution domain and the information in it to better estimate the calculated solution, or to adjust the domain in such way that the computation resources can be better focused on the most important part of the solution in a given domain.

The two classes of solution refinement can be classified based on the refinement of the mesh (h-refinement), or on the refinement of the order of accuracy of the solution scheme that is being applied (p-refinement). For Godunov-type finite-volume schemes, p-refinement is done by increasing the order of solution reconstruction before application of a slope limiter [7, 8]. This method uses more of the information contained in the mesh at each computational step to better estimate the solution. Conversely, the h-refinement approach refines part of the mesh that is deemed to be of higher importance to the solution accuracy than other parts of the mesh that remain at the original refinement level, or may even be coarsened to further save on computational resources.

Mesh refinement and/or coarsening works in two basic steps. First, a decision must



(a) Base Cartesian grid **(b)** Patch-based AMR **(c)** Cell-based AMR **(d)** Block-based AMR

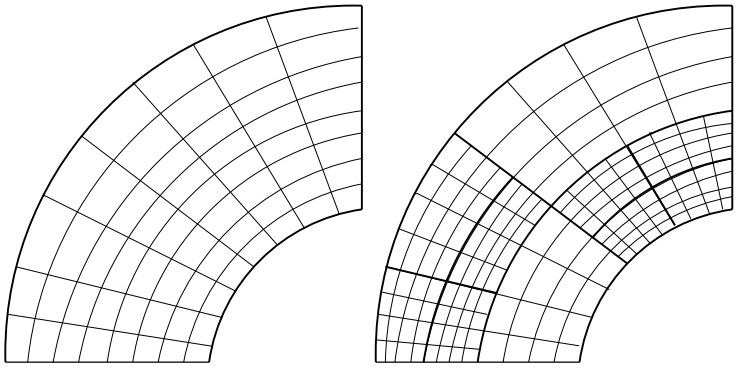
Figure 1.1: Illustration of (b) patch-based, (c) cell-based, and (d) block-based AMR techniques applied to a base Cartesian mesh (a) with cells flagged for refinement indicated by black dots.

be made, if and where the mesh should be refined, or coarsened. Then, the mesh needs to be adjusted accordingly and the solution information transformed to the new grid. There is a number of AMR types that have been successfully developed.

As a review of the various AMR strategies, consider the mesh flagged for refinement shown in Figure 1.1a. The mesh can be refined using a patch-based AMR method, as depicted on Figure 1.1b. Such an approach was proposed by Berger and Collela [9–11]. In this approach, the part of the mesh flagged for refinement is overlapped with a denser mesh creating a *patch* of dense mesh containing the flagged cells of the original mesh domain. For a general case, it is possible that the finer patch also covers cells that were not flagged for refinement.

The alternative AMR method of Figure 1.1c, proposed by Powell *et. al* [12–14], Berger and Leveque [15], and Aftosmis *et al.* [16–18] refines only the cells flagged for refinement, thus potentially making the computational domain smaller than the patch-based refinement. An issue can arise when most of the domain is flagged for refinement. The complexity of the algorithm that tracks each individual mesh cell as unique, independently refined units can become computationally expensive.

Figure 1.1d shows a much simpler approach with respect to implementation and parallel scalability, presented by a blocks-based AMR approach, as described by Quirk [19], Berger [20] and Groth *et al.* [3,21–27]. Such an approach shall be considered here. In particular, the focus of this study is on automatic, solution-directed, multi-block, body-fitted (curvilinear) AMR, developed by Groth *et al.*, that uses quadrilateral (2D case)



(a) Body-fitted coarse grid **(b)** Refined body-fitted grid

Figure 1.2: Illustration of block-based adaptive mesh refinement on body-fitted grid topology showing original coarse grid (a) and refined grid (b).

computational cells illustrated by Figure 1.2. The latter uses an efficient quad-tree data structure to store the mesh hierarchy and block connectivity. The algorithm has been proven to be highly parallel on distributed memory architectures, scalable and flexible.

Lastly, hybrid block-based AMR techniques can be used. This approach involves using two types of the previously mentioned techniques to implement the desired features of each approach. The problem with this approach is that the algorithm does not inherit only the benefits from the mixed approach, but also complexity, higher computational requirements and potential issues when it comes to resolving and preserving the solution during the refinement and coarsening processes.

1.2.2 Adaptive Mesh Refinement Criteria

Any of the AMR techniques, no matter how general, or sophisticated, are only as good, as the logic behind the refinement flagging algorithm. There are different ways of using the solution information to select if and where to adjust the solution grid. The two basic approaches that make use of the solution error are those that use *a priori* knowledge from the solution and those that use *a posteriori* solution analysis.

The *a priori* solution error assessment is based on error estimates made before the solution is calculated. The solution errors are often difficult, or impossible to predict without some prior knowledge of the final solution.

AMR based on heuristic measures are also possible. In particular, values such as the gradient, curl, or divergence of the solution can be used to direct the mesh refinement. These physics-based refinement criteria are effective in capturing dominant flow features in the solution [3, 12, 19, 24–28]. The problem is that capturing shock waves, boundary layers, vortices and flame fronts does not guarantee that the calculated solution after refining grid would actually reduce the error of the solution due to discretization of the domain.

To address the problems with *a priori* refinement methods, new refinement techniques based on *a posteriori* solution analysis were developed. Venditti and Darmofal [29–31], and Rannacher *et al.* [32, 33] developed an adjoint-based error correction technique for this purpose. Nemec *et al.* [34–36] have also shown adjoint-based error estimates for adaptive mesh refinement used with embedded-boundary Cartesian meshes to be effective for increasing the accuracy of simulations of aerodynamic flows. Such methods use an adjoint-based error estimation that provides opportunity to flag the solution grid for refinement so that the refined grid would produce a more accurate solution. Moreover, since the adjoint problem can focus on various studied integral quantities, the refinement can be guided to lower the error in a specific solution functional. Therefore, the adjoint based AMR is of great interest for design optimisation.

1.3 Thesis Objective

This thesis aims to both develop and implement an *a posteriori* adjoint based refinement criteria to an existing CFD numerical framework developed by Groth *et al.* [3,21–27]. The newly developed AMR criteria are based on a functional of interest that will be at the core of the refinement strategy. The aim is to reduce the functional error by making the mesh more dense in the areas where the functional and solution residual(s) produce the largest error in the solution. By guiding the AMR technique in this way, the computational resources are effectively focused on the most critical part of the solution with respect to the functional solution accuracy. The refinement criteria is applied to an existing, two dimensional, isotropic AMR within a body-fitted, curvilinear mesh used by finite-volume advection-diffusion solver. The test cases aim at highlighting the performance

of the proposed adjoint-based AMR refinement in situations where *a priori* refinement methods fail or have poor performance.

1.4 Scope of Research

The thesis work presented here concentrates on implementing the *a posteriori* error estimation based on the previous work of Venditti and Darmofal [29, 30] with the Computational Framework for Fluids and Combustion (CFFC) developed by CFD and Propulsion Group at the University of Toronto Institute for Aerospace Studies (UTIAS). The key to this method is to calculate the solution error estimates for a specific functional. The implementation required development of new parts to the CFFC code.

For an effective implementation of the adjoint-based refineemnt criteria, the errors in the functional need to be analysed. There are number of ways the estimated functional errors can be used for adaptive mesh refinement purposes. This study looks at the different refinement criteria options and evaluates their potential for the adaptive mesh refinement method. The AMR based on the *a posteriori* error estimates is compared to gradient-based AMR and uniform mesh refinement. The relative performance of the different mesh refinement techniques are assessed.

The thesis is divided into multiple sections that describe the theoretical background, results and comment on the findings of this study. Chapter 2 describes the governing equations that were used to analyse the adjoint-based AMR criteria. It contains description of advection-diffusion equation and the adjoint-based error formulations. Numerical methods used to solve the governing equations are described in Chapter 3. The description of AMR strategies and some of the implementation details are described in Chapter 4. This chapter also includes the details of how the *a posteriori* adjoint-weighted errors are used for the guidance of the AMR process. The numerical results are shown in Chapter 5 and the findings of this study are summarized in Chapter 6.

Chapter 2

Governing Equations

2.1 Advection-Diffusion Equation

The advection-diffusion equation is a convenient mathematical model to use in the initial development of new CFD algorithms and solution methods. The transport mechanisms of real flows rely on advection for particle transport along the stream and on diffusion for transport of flow quantities from high concentration to regions of lower concentration. The advection-diffusion equation captures these two essential transport mechanisms and their properties in a single partial differential equation (PDE).

The unsteady advection-diffusion equation with a source term given by

$$\underbrace{\frac{\partial u}{\partial t}}_{\text{Transient Term}} + \underbrace{\vec{\nabla} \cdot (\vec{V} u)}_{\text{Advection Term}} = \underbrace{\vec{\nabla} \cdot \kappa \vec{\nabla} u}_{\text{Diffusion Term}} + \underbrace{\phi(x, y, u)}_{\text{Source Term}}, \tag{2.1}$$

contains a single solution variable u and four terms, so the model is relatively easy to solve compared to Navier-Stokes equations and other, more complex, systems of equations. Moreover, unlike the Navier-Stokes equations or other more complex systems of partial differential equations, the advection-diffusion equation has more readily available analytical solutions making the solution accuracy easier to analyse.

The *transient term* allows the time rate of change of the solution, u, due to hyperbolic and elliptical fluxes as well as the influence of the *source term*.

The hyperbolic part of the Equation (2.1) represents the advection of the solution

content. A simple one dimensional partial differential equation of this type is represented by

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0\,. \tag{2.2}$$

This purely hyperbolic PDE contains no diffusion, so the initial conditions are transported with velocity determined by a real-valued coefficient a. The advection coefficient in Equation (2.1) is represented by the velocity vector $\vec{V}$.

A simple one dimensional elliptical PDE can be represented by Poisson's equation

$$\bigtriangledown^2 u = \phi(x, y, u)\,. \tag{2.3}$$

This equation represents pure diffusion. The source term $\phi(x, y, u)$ can be used to control dynamic changes of local concentrations in the system. If the source term is dependent on the solution u, the term becomes non-linear. The rate of diffusion in Equation (2.1) is determined by a real coefficient κ in the *diffusive term*.

The dominance of advection or diffusion in the Equation (2.1) can be assessed by calculating a Péclet number defined as

$$Pe = aL/\kappa\,. \tag{2.4}$$

The coefficient L represents a typical length scale of the problem; other coefficients are as described before. Problems with Péclet numbers between zero and one corresponds to problems dominated by diffusion, while the problems dominated by advection have Péclet numbers greater than unity. Purely diffusive problem corresponds to the Péclet number of zero. A purely advective problem has Péclet number approaching infinity.

2.2 Adjoint-Based Error Estimation

The goal of *a posteriori* error analysis is to minimise the error of a particular scalar functional of interest. This is done by assessing the impact of mesh refinement on the functional value and its error. Therefore, by studying the functional error, it is possible to concentrate the computational resources on parts of the domain that have the greatest

effect on the solution accuracy for the particular functional. The adjoint problem and definition of adjoint-based error estimates for a desired functional is described in the paper by Venditti and Darmofal [30, 37] and is now summarized in this section for the sake of completeness.

Following Venditti and Darmofal [30, 37], the functional sensitivity to the mesh refinement is assessed using two levels of mesh refinement. The adjoint problem is first solved on the original, or coarse, grid Ω_H, with grid spacing H. The solution vector, $\mathbf{U}$, for the problem which becomes a scalar quantity, u, for the advection-diffusion equation, is used to estimate the functional of interest $f(\mathbf{U})$. This approximation of the functional on the coarse grid using coarse grid solution, is represented as $f_H(\mathbf{U}_H)$. The coarse grid solution is already available and the functional is cheap to compute, but may not be accurate enough for the desired purpose. Hence, a solution of the problem on a fine grid, Ω_h is considered. The computational cost of such solution is naturally much higher than the solution on the coarse grid. Calculation of functional $f_h(\mathbf{U}_h)$ using the actual fine mesh would make the AMR approach pointless since the expensive computations have been already performed. For any AMR method to be able to produce a higher quality solution with respect to the observed integral quantity compared to the uniform mesh, the AMR method must estimate the fine grid solution.

For the method to be effective, the estimation of the fine grid solution must be much less expensive than simply performing a uniform mesh refinement and solving the problem on a fine grid. The error in the functional solution is estimated using the solution residual as represented by the semi-discrete form of the spatially discretized differential equations of interest and given by

$$\mathbf{R}(\mathbf{U}) = \frac{d\mathbf{U}}{dt}\,. \tag{2.5}$$

For a steady state, the solution without any error has $\mathbf{R}(\mathbf{U}) = 0$. The fine grid solution $f_h(\mathbf{U}_h)$ can be expanded using Taylor expansion about the coarse grid

$$f_h(\mathbf{U}_h) = f_h(\mathbf{U}_h^H) + \left.\frac{\partial f_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H} (\mathbf{U}_h - \mathbf{U}_h^H) + \dots\,, \tag{2.6}$$

where $[\partial f_h/\partial \mathbf{U}_h]|_{\mathbf{U}_h^H}$ represents the linear sensitivities of the functional with respect to the solution on the fine grid. The term $\mathbf{U}_h^H$ represents the coarse grid solution that has

been prolonged onto the fine grid. The prolongation operator, $\mathbf{I}_h^H$, is defined as:

$$\mathbf{U}_h^H \equiv \mathbf{I}_h^H \mathbf{U}_H \,. \tag{2.7}$$

It is assumed that the residual operator on the fine grid satisfies

$$\mathbf{R}_h(\mathbf{U}_h) = 0 \,. \tag{2.8}$$

For a converged, steady state, solution, after linearisation, Equation (2.8) becomes

$$\mathbf{R}_h(\mathbf{U}_h) = \mathbf{R}_h(\mathbf{U}_h^H) + \left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H} (\mathbf{U}_h - \mathbf{U}_h^H) + \ldots \,. \tag{2.9}$$

Solving Equation (2.9) for the approximate error vector results in

$$(\mathbf{U}_h - \mathbf{U}_h^H) \approx \left[\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H}\right]^{-1} \mathbf{R}_h(\mathbf{U}_h^H) \,. \tag{2.10}$$

Substituting Equation (2.10) into Equation (2.6), yields an estimate of the functional

$$f_h(\mathbf{U}_h) \approx f_h(\mathbf{U}_h^H) - (\,\boldsymbol{\Psi}_h|_{\mathbf{U}_h^H})^T \mathbf{R}_h(\mathbf{U}_h^H) \,. \tag{2.11}$$

The term $\boldsymbol{\Psi}_h|_{\mathbf{U}_h^H}$ is the discrete adjoint solution satisfying

$$\left[\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H}\right]^T \boldsymbol{\Psi}_h|_{\mathbf{U}_h^H} = \left(\left.\frac{\partial f_h}{\partial \mathbf{U}_h}\right|_{\mathbf{U}_h^H}\right)^T \,. \tag{2.12}$$

The estimate of $f_h(\mathbf{U}_h)$ in Equation (2.11) is exact for linear residuals and functionals. However, this equation still requires the adjoint problem to be solved on the fine grid, which should be avoided. Therefore the term $\boldsymbol{\Psi}_h|_{\mathbf{U}_h^H}$ is interpolated from the coarse grid solution

$$\boldsymbol{\Psi}_h^H \equiv \mathbf{J}_h^H \boldsymbol{\Psi}_H \,, \tag{2.13}$$

where the $\mathbf{J}$ operator is a prolongation or projection operator (such as operator $\mathbf{I}$ used previously in Equation (2.7)). The coarse grid adjoint, $\boldsymbol{\Psi}_H$, is found by solving the

following system

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{U}_H}\right]^T \mathbf{\Psi}_H = \left(\frac{\partial f_H}{\partial \mathbf{U}_H}\right)^T . \tag{2.14}$$

Substituting the interpolated coarse grid adjoint to Equation (2.11), the functional estimate takes a computable form

$$\tilde{f}_h(\mathbf{U}_H) = f_h(\mathbf{U}_h^H) - (\mathbf{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H) . \tag{2.15}$$

Using the Equation (2.11), it is possible to estimate the functional if a fine grid solution was calculated. Since this is only an estimate, it is desirable and useful to approximate the error of this estimate. The difference between a true fine grid solution functional and a functional estimated on fine gird using the coarse solution can be expressed as

$$f_h(\mathbf{U}_H^h) - f_h(\mathbf{U}_h) \approx \underbrace{(\mathbf{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{Computable Correction}} + \underbrace{(\mathbf{\Psi}_h|_{\mathbf{U}_h^H} - \mathbf{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{Error in Computable Correction}} . \tag{2.16}$$

Equation (2.16) represents the total error estimation for the fine grid solution.

Both parts of the total functional error estimate can be used as criteria for AMR method as described in Chapter 4. This equation can be rewritten as

$$f_h(\mathbf{U}_H^h) - f_h(\mathbf{U}_h) \approx \underbrace{(\mathbf{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{Computable Correction}} \underbrace{- \{\mathbf{R}_h^\Psi(\mathbf{\Psi}_h^H)\}^T \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\bigg|_{\mathbf{U}_h^H}\right]^{-1} \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{Error in Computable Correction}} , \tag{2.17}$$

where $\mathbf{R}_h^\Psi$ is the adjoint residual operator defines as

$$\mathbf{R}_h^\Psi(\Psi) \equiv \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\bigg|_{\mathbf{U}_h^H}\right]^T \Psi - \left(\frac{\partial f_h}{\partial \mathbf{U}_h}\bigg|_{\mathbf{U}_h^H}\right)^T . \tag{2.18}$$

A dual form of the error in computable correction described by Equation (2.16) can be derived using Equation (2.9) and Equation (2.18). This form is defined as

$$f_h(\mathbf{U}_H^h) - f_h(\mathbf{U}_h) \approx \underbrace{(\mathbf{\Psi}_h^H)^T \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{Computable Correction}} + \underbrace{\{\mathbf{R}_h^\Psi(\mathbf{\Psi}_h^H)\}^T (\mathbf{U}_h - \mathbf{U}_h^H)}_{\text{Error in Computable Correction}} . \tag{2.19}$$

The duality on the functional error estimate arise from the ability to compute the error using either adjoint residual with difference in fine grid solutions, or the solution residual with the difference in fine grid adjoint solutions. The two errors should be similar assuming the numerical schemes used to calculate both errors are accurate enough.

The error estimates described above can be used to analyse the functional value and error. Also, since it is possible to compare the coarse and estimated fine grid functionals, it is possible to asses the sensitivity of the functional value to the mesh resolution.

Chapter 3

Numerical Solution Methods

3.1 Finite-Volume Method

3.1.1 Godunov-Type Finite-Volume Method

Computational fluid dynamics uses partial differential equations (PDEs) to model and study fluid flow problems. Finite-volume methods are well established in solving such problems. The block-based adaptive mesh refinement studied in this work has been implemented to work with a Godunov-type finite-volume method.

The advection-diffusion equation of interest discussed previously can be rewritten and expressed as

$$\frac{\partial u}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = \phi(x, y, u)\,. \tag{3.1}$$

The terms u and $\vec{\mathbf{F}}$ represent the solution variable and its flux, respectively. The term, ϕ represents a source term. The flux term contains elliptical (diffusive) and hyperbolic (advective) parts. This scalar PDE has to be solved on a discrete numerical grid using an appropriate numerical method. Finite-volume methods use each mesh cell as a finite control volume. The differential from of a conservation equation is then integrated over this control volume, $\mathcal{V}$, to obtain a set of discrete, average solution values, $\bar{u}_{i,j}$, at each computational cell. For a two dimensional problem, the cells are labelled by two indices, (i, j), that correspond to the discrete cell location in each direction of the solution domain.

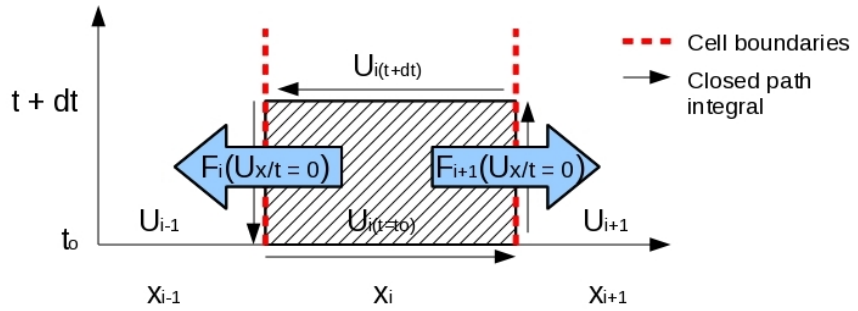


Figure 3.1: Closed-path integral used by the Godunov's method for a 1D problem shown on space-time plot

The Integral form of Equation (3.1) is described by

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} u(x,y)\,d\mathcal{V} + \int_{\mathcal{V}} \vec{\nabla}\cdot\vec{\mathbf{F}}\,d\mathcal{V} = \int_{\mathcal{V}} \phi(x,y,u)\,d\mathcal{V}\,, \tag{3.2}$$

where the average solution value for each control volume is then defined by

$$\bar{u}_{i,j} = \frac{1}{\mathcal{V}_{i,j}}\int_{\mathcal{V}} u_{i,j}(x,y)\,d\mathcal{V}\,. \tag{3.3}$$

In the finite-volume discretization, the control volume's average solution value, stored at the centroid of each cell, is balanced with the fluxes across the cell boundaries. Each time step, the solution is adjusted based on the imbalance of the fluxes adding and removing quantities to/from the control volume. A steady state is achieved when the solution is no longer changing and the incoming and outgoing fluxes are balanced.

The great importance of Godunov's work lies in the monotonicity and reliability of his method when it comes to dealing with discontinuities presented by Riemann problems for hyperbolic PDEs [38]. His conservative finite-volume-type scheme for hyperbolic systems is first order accurate and monotonic. Godunov showed that higher order schemes, despite producing better smooth solution representation, lose their monotonicity at discontinuities. This means that only piecewise-constant solution reconstruction within each cell guarantees the monotonicity of the solution.

The core of the Godunov's finite volume scheme lies in the integral depicted on Figure 3.1. The scheme relates the solution change in the control volume with the fluxes

across the closed path integral on the volume's boundaries. Figure 3.1 provides an illustration of a control volume on space-time plot for a one dimensional problem. The changes of the solution over time depend on the numerical fluxes across the boundaries with the neighbouring cells. Mathematically, the time rate of change for $i-th$ cell's solution in one dimensional problem, can be represented as

$$\frac{du_i}{dt} = -\frac{1}{\Delta x}\left[\mathbf{F}_{i+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2}}\right], \tag{3.4}$$

where the variables $\mathbf{F}_{i+\frac{1}{2}}$ and $\mathbf{F}_{i-\frac{1}{2}}$ represent the right and left hyperbolic fluxes for the respective boundaries. These fluxes are solutions to the Riemann problems on the right and left control volume boundaries. This leads the solution to transport solution values in the direction of the flow. For hyperbolic system, the Godunov's method is a stable, upwind, first-order scheme for solving PDEs. For a higher order scheme and for PDEs containing not only hyperbolic fluxes, the method needs to be modified.

3.1.2 Semi-Discrete Form

The advection-diffusion equation of Equation (2.1) can be placed in a semi-discrete form through the application of the finite-volume method. For two dimensional quadrilateral grid, with the average solution $\bar{u}_{i,j}$ belonging to the cell (i,j), the semi-discrete form of Equation (2.1) takes the form

$$\frac{\mathrm{d}\bar{u}_{i,j}}{\mathrm{d}t} = -\frac{1}{A_{i,j}}\sum_{l=1}^{N_f}\sum_{m=1}^{N_G}\left(\omega\vec{\mathbf{F}}\cdot\vec{n}\Delta l\right)_{i,j,l,m} + \phi_{i,j} = R_{i,j}\,, \tag{3.5}$$

where

$$\bar{u}_{i,j} = \frac{1}{A_{i,j}}\int_A u_{i,j}dA\,. \tag{3.6}$$

Each control volume has its area $A_{i,j}$, and the time-rate of change is calculated using Gaussian numerical integration over N_G Gaussian points and summing the fluxes of all faces. For a quadrilateral cell, there are $N_F = 4$ faces that cover the entire closed-loop integral around the control volume. The solution average, $\bar{u}_{i,j}$, is calculated using reconstructed solution, $u_{i,j}$, in each control volume. The term $R_{i,j}$ represents a residual.

The residual represents the flux imbalance with the source term at each numerical cell at the current time step. For a fully converged, steady-state solution, the residual should be zero. The overall degree of accuracy of the numerical scheme depends on the degree of accuracy of the solution reconstruction in each cell, order of Gaussian integration and the order of time marching scheme that is used to advance the solution by a discrete time step.

3.1.3 Flux Evaluation for Advection-Diffusion Equation

The advection-diffusion equation requires a treatment of both hyperbolic and elliptical fluxes for each control volume in the domain. The hyperbolic, or advective, flux is defined as $\vec{F_a} \cdot \vec{n} = u\vec{V} \cdot \vec{n}$. A scalar advection problem involves a solution to the Riemann problem with one characteristic speed, $\vec{V}$. This means that the flux, for one dimensional problem, at $x/t = 0$ is based on an upwind value of $u(x/t) = 0$. The graphical representation of the solution propagation for such problems is shown on Figure 3.2. In case of two dimensional problems, the upwinding of the solution is guaranteed by taking a dot product of velocity vector, $\vec{V}$, and a normal vector of the computational cell's interface, $\vec{n}$. The possible values this gives for the hyperbolic flux are

$$\vec{F_a} \cdot \vec{n} = \begin{cases} u_l(\vec{V} \cdot \vec{n}) & \text{if} \quad \vec{V} \cdot \vec{n} \geq 0\,. \\ u_r(\vec{V} \cdot \vec{n}) & \text{if} \quad \vec{V} \cdot \vec{n} < 0\,. \end{cases} \tag{3.7}$$

The flux at cell's interface a, $\vec{F_a}$, is either the average value of the cell (i, j), or the average value of the neighbouring cell. When looking at left cell interface, the left solution, u_l, refers to the value stored at the neighbouring bell to the left of the cell (i, j). The right solution, u_r, is the value of cell (i, j). For the right cell interface, the left value refers to the solution value in cell (i, j), while the right value refers to the value stored in the right neighbour of the cell (i, j). The Riemann problems need to be solved at each interface of the cell (i, j) for all Gauss quadrature points on the interface.

The elliptic, or diffusive, flux for advection-diffusion equation is defined as $\vec{F_d} \cdot \vec{n} = -\kappa\vec{\nabla}u \cdot \vec{n}$. As with the hyperbolic flux, elliptic flux needs to be integrated over the interface of each computational cell. Numerically, the first-order forward-difference in

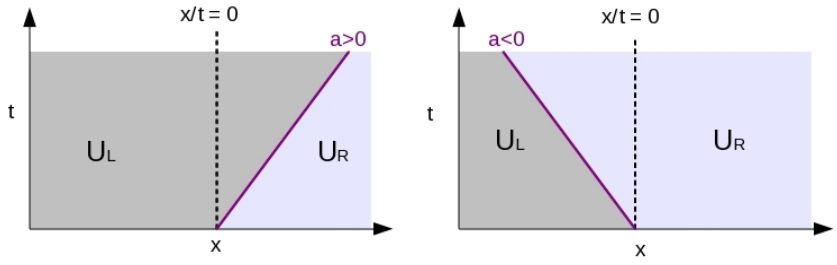


Figure 3.2: Riemann problem solution to a one dimensional scalar advection problem.

one dimension for the left and right interface fluxes of cell i, assuming constant diffusion coefficient, κ, is

$$F_{d,left} = \kappa \frac{u_i - u_{i-1}}{\Delta x} \quad \text{and} \quad F_{d,right} = -\kappa \frac{u_{i+1} - ui}{\Delta x} \,. \tag{3.8}$$

The solution variables, u, for each case represent the reconstructed interface solution for each side of the cell's interface. Once the fluxes are summed, the elliptic flux becomes a second-order central-difference approximation to term $\kappa(\partial^2 u/\partial x^2)$. For a two dimensional case, the average gradient for each interface is estimated using diamond-path reconstruction [39]. The flux evaluation again reduces to a second-order, central-difference approximation of the term $\kappa(\partial^2 u/\partial x^2)$ in case where the diffusion coefficient, κ, is constant.

3.1.4 Second-Order Scheme and Slope Limiting

The accuracy of the solution representation in each control volume depends on the order of the solution reconstruction. The higher the degree of solution reconstruction, the better the solution can represent solutions to complex problems. However, as mentioned in Section 3.1.1, Godunov showed that only first-order solution scheme is monotone. The first step to produce higher order finite-volume scheme is to create a second-order scheme that preserves monotonicity of the first-order scheme. Increasing the order of the numerical scheme has many advantages. Firstly, it can produce more accurate solutions since more of the solution features can be represented by higher-order polynomials as compared to piecewise constant representation of the solution. Secondly, the computational

resources needed to achieve comparative solution accuracy can be lower for a higher-order solution scheme as lower mesh densities can produce better solution accuracy when compared to a first-order scheme. Lastly, the higher-order schemes are better suited to the computational hardware that is currently being used. This means that the solution can be calculated more quickly with a higher-order scheme even if the calculation of the solution requires a similar number of operations. The advantage of the higher-order schemes comes from the ability to use more information from stored data than first-order scheme. The final result is that the computer's processor(s) can be used more effectively and the number of communications between different components in a computer is reduced.

The extension to second-order accuracy is easiest to explain by considering a one dimensional problem. The idea is to use a linear solution reconstruction with a slope limiter to preserve monotonicity. This means that the numerical dissipation is reduced while the monotonicity is preserved by making the problem first-order near discontinuities such as shock waves. Solution reconstruction for cell i in this case is given by

$$u_i(x) = \bar{u}_i + \psi_i \left[\frac{\partial u}{\partial x}\Big|_i (x - x_i) \right] \tag{3.9}$$

Where the reconstructed solution, $u_i(x)$, is a piecewise linear representation of the solution. The average solution, $\bar{u}_i$, stored at the center of the cell is a pivot point through which passes the linear line representing the reconstructed solution. The slope of the unlimited solution reconstruction is given by the first-derivative of the solution with respect to the spatial axis along which the solution is being reconstructed, $\partial u/\partial x$. This derivative is approximated using central stencil and least-squares reconstruction. This produces a second-order central difference for the first derivative using the values stored at the neighbouring cells

$$\frac{\partial u}{\partial x}\Big|_i = \frac{1}{2\Delta x}(\bar{u}_{i+1} - \bar{u}_{i-1})\,. \tag{3.10}$$

The term ψ_i in Equation (3.9) is the slope limiting term. It can have values between zero and one. The unlimited solution which is valid for smooth regions of the solution between local extrema has a slope limiter with the value of one. If the solution reconstruction would produce new extrema, the reconstruction would be limited to prevent this situation. For sharp discontinuities that would produce new maxima, the slope limiter has value

of zero. In this way, the monotonicity of Godunov's first-order scheme is preserved and the smooth parts of the solution are better represented. The slope limiters used in this work were described by Barth and Jespersen [40] and Venkatakrishnan [41]. These slope limiters have been implemented and successfully tested in the computational framework and proved to work well for problems solved using highly parallel computing.

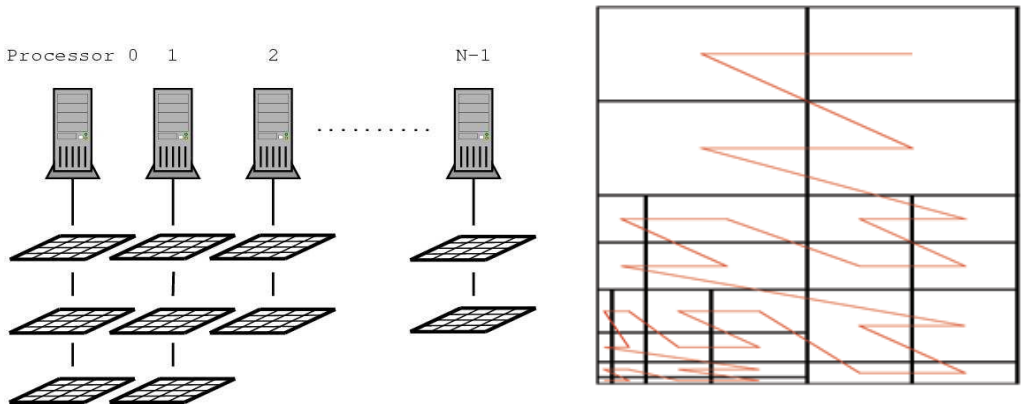
3.1.5 Time Marching Integration Method

The semi-discrete advection-diffusion equation shown by Equation (3.5) must be integrated in time using a time marching method. The time marching algorithms used in this work include the explicit Euler method and a second-order predictor-corrector method for global solution accuracy of first and second order, respectively [42]. The predictor corrector method is of the Runge-Kutta type. The numerical results were obtained using a second order Runge-Kutta method. For the problems considered, the solutions were advanced in time until a steady state solution was achieved.

3.2 Parallel Implementation

The proposed finite-volume scheme was implemented within a computational framework that makes use of multi-block quadrilateral meshes stored in a quad-tree data structure. This facilitates computational domain decomposition, distribution and inter-domain communication easy. The implementation is efficient and scalable for both explicit and implicit solution methods on distributed-memory and multi-processor architecture [3, 24]. The *a posteriori* refinement flagging and all of its supporting parts are designed to be compatible with the original code. This means that the code can be run with all solution blocks using a single processor on one extreme, or n-processors can be called to solve a single solution block. The entire range of processor loading and parallel solution domain decomposition was tested and performed as expected.

The solution algorithm has been implemented using the C++ programming language with a message passing interface (MPI) library handling the inter-processor communication [43]. The solution blocks are distributed across all processors using Morton ordering [2]. The goal is to distribute the solution blocks of equal computational cost evenly



(a) Parallel Domain Decomposition **(b)** Morton Ordering on 2D Grid

Figure 3.3: Domain Decomposition and Block Ordering

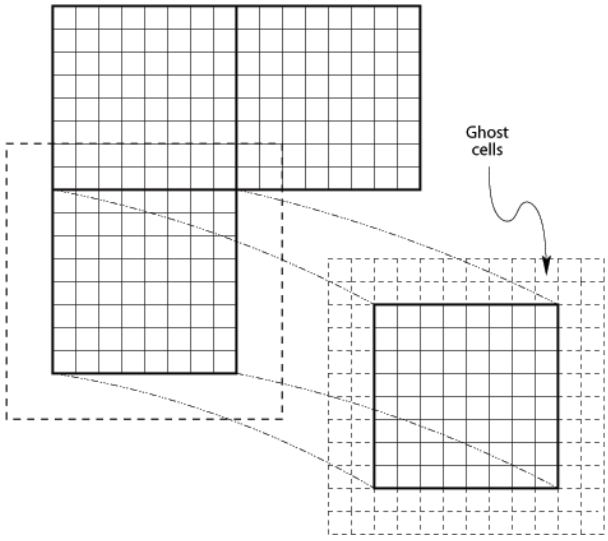


Figure 3.4: Two layers of overlapping "ghost" cells contain solution information from neighbouring blocks.

across the available computational resources. Figure 3.3a shows the solution blocks distribution over N processors and Figure 3.3b illustrates the Morton ordering of blocks. The two processes work hand-in-hand to distribute the domain evenly and keep blocks that are close together on a single processor as much as possible. This approach improves the performance and scalability of the problem over a wide range of computational platforms raging from a cheap school laptop to a cluster of supercomputers.

The communication between blocks is accomplished by the use of ghost cells. Fig-

ure 3.4 shows how the ghost cells fit into the computational domain. The ghost cells contain data from neighbouring blocks. This means that even if the blocks are not being solved on the same computer, the solution blocks are still connected and the problem is being solved as a global problem.

Solution Procedure for the Adjoint Problem

The error estimation technique considered here requires the solution of the adjoint problem and the evaluation of the residual sensitivities. The residual sensitivities $\partial\mathbf{R}/\partial\mathbf{U}$ were calculated using a finite difference method

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} = \frac{\mathbf{R}(\mathbf{U}+\epsilon) - \mathbf{R}(\mathbf{U})}{\epsilon}\,, \tag{3.11}$$

where the terms $\mathbf{R}(\mathbf{U})$ represent the solution residual using solution $\mathbf{U}$ and ϵ is a small perturbation on the order of 1×10^{-8}.

For the adjoint calculation, new solution methods and data structures were required in addition to the already implemented linear algebra tools within the CFFC framework. The adjoint problem involves the solutions of large linear systems of equations. The Jacobian for global system of size m by n has a size of $(m \times n)$ by $(m \times n)$. Since the large matrices for the linear systems are sparse, a new data structure was created. It is compatible with the data structures already used in the framework and its behaviour is similar to an existing dense matrix data structure within the code. The evaluation of the $Ax = b$ linear systems was found to be inefficient with previously implemented linear solvers, so a new library was added to the computational framework. Trilinos library by Sandia National Laboratories [44] was used to solve the linear systems. The generalized minimal residual method (GMRES) in the Trilinos package was found to be an efficient solver. All of the new additions to the code maintain the scalability and ability to solve the problem in parallel fashion.

Chapter 4

AMR on Body-Fitted Multi-Block Meshes

4.1 Adaptive Mesh Refinement

The aim of an adaptive mesh refinement is to produce a mesh that is dense in regions where the solution algorithm requires the solution domain to be well resolved, while keeping the less-critical parts of the solution domain on a coarse mesh. The parts of the solution domain that contribute the most to the error of the solution are prime candidates for mesh refinement. Section 4.2 elaborates on what it means for a solution to be accurate and how the AMR process can be guided.

4.1.1 Overview of Adaptive Mesh Refinement Scheme

The AMR technique originally developed by Groth *et al.* is used in this work [24–27]. A body-fitted, block-based, automatic, adaptive mesh refinement is used with finite-volume scheme to solve fluid problems. The computational domain is discretized into mesh containing discrete cells that form the smallest individual control volumes of the problem. The cells are grouped into blocks that each have $(m \times n)$ cells for a two dimensional case. This makes the computational cost of solving each block similar and thus balancing the computational resources on multi-processor platforms becomes easier compared to other AMR strategies. The blocks are stored in a quad-tree data structure

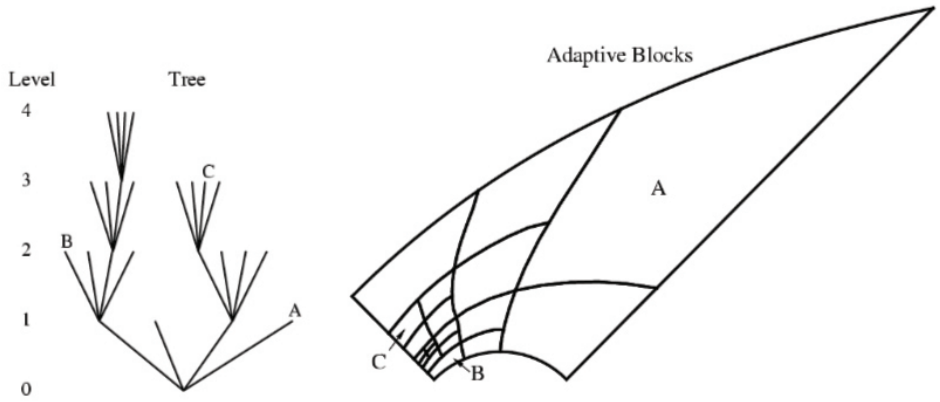


Figure 4.1: Solution blocks of a computational mesh after four levels of AMR on an initial block, shown with the corresponding quad-tree data structure [3].

as shown on Figure 4.1. Each block that is refined, becomes a parent block to four new solution blocks. This makes the mesh of the new solution blocks twice as dense as was the mesh density of the parent block. The quad-tree arrangement allows a quick and effective storage of the neighbouring blocks, level of refinement and level of refinement of neighbouring blocks. This information is necessary to ensure that the largest difference of refinement level between blocks is one.

4.1.2 Refinement Procedure of a Solution Block

Once the solution algorithm reached the adaptive mesh refinement stage and the solution block has been selected to be legally refined, many steps need to be performed to double the current mesh density. Firstly, the original solution block data is copied and the current solution block becomes a parent to four new quad-tree branches. Each new quad-tree branch contains a new solution block that inherits data from the parent blocks via direct injections. The new blocks are assigned their neighbours and communication is established between the neighbouring blocks. Ghost cells are assigned values according to the type of boundary that is found at each edge. Now, the mesh density in the place of the original mesh is doubled and contains four new solution blocks that fit into the domain as illustrated by Figure 4.2 and the solution blocks are ready to be solved.

The procedure for coarsening a group of four solution blocks is similar. The four solution blocks revert to the resolution of their parent block and all steps described for

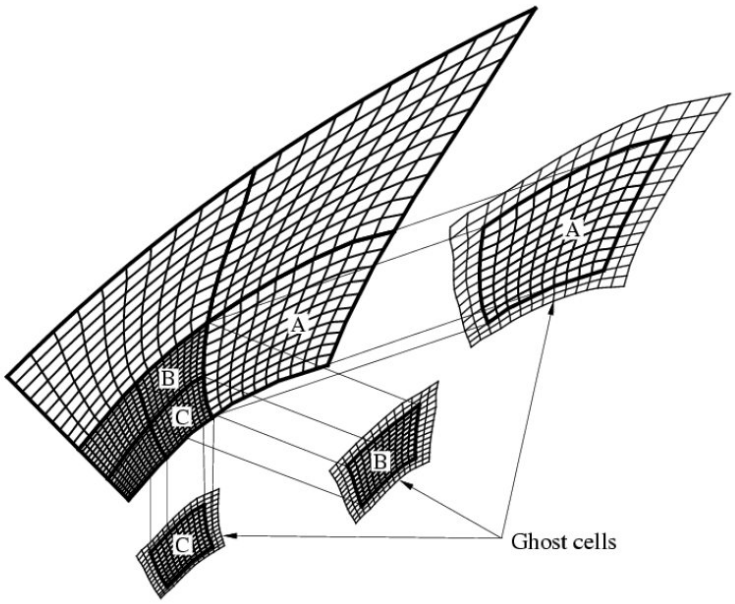


Figure 4.2: Multi-block quadrilateral mesh in block-based AMR with layers of overlapping ghost cells to facilitate inter-block communication.

refinement are carried in the same manner, only the solution is reduced to half the mesh density.

4.1.3 Solution Prolongation of a Solution Block during *a Posteriori* Adjoint-Based Grid Refinement

For *a posteriori* adjoint based error estimation and mesh refinement, the solution must be solved to obtain a steady state result and the errors in solution functional need to be estimated based on the theory described in Section 2.2. The fine grid terms are estimated using a modified AMR refinement procedure. The coarse grid solution is copied into new data structures and then the whole grid is uniformly refined in manner described in Section 4.1.2. The problem of the original refinement procedure is that it uses a direct injection of the solution values from coarse to fine grid. This creates a set of piecewise-constant tiles on a two dimensional mesh. Hence, the refinement produces unacceptably poor fine grid solution estimation with large residuals. Therefore, the procedure was modified and direct injection of solution values was replaced by a linear-piecewise solution reconstruction that uses monotonicity preserving slope limiter. This produces a much

more realistic fine gird solution estimation.

4.2 Adaptive Mesh Refinement Criteria

The required accuracy of a numerical solution greatly depends on how the solution is to be used. If one is interested in the flow features, such as shock waves, vortices or mixing layers, gradient based adaptive mesh refinement as described below is a reasonable choice for finding the important parts of the solution domain for mesh refinement. For many engineering purposes; however, the actual fluid flow features may not be the most important parts of the solution when it comes to estimating lift, drag on a wing, or heat energy being transmitted to a wall inside an engine. For such problems, *a posteriori*, adjoint-based adaptive criteria provides a better refinement criteria as it focuses on the errors in the functional of interest, rather than on physical flow features.

The advantage of the non-uniform mesh density can be measured using refinement efficiency parameter, η, defined as:

$$\eta = 1 - \frac{N_{used\ cells}}{N_{cells\ on\ uniformly\ refined\ mesh}}\,. \tag{4.1}$$

As the number of used cells approach the number of cells used by uniform mesh refinement, the η parameter approaches zero. Hence, the greater non-uniformity of the mesh refinement, the greater the adaptive mesh refinement efficiency and the greater the computational cost savings.

4.2.1 Gradient-Based Refinement Criteria

For the purpose of studying many fluid flows, physics based, gradient based refinement criteria can be quite effective. The magnitude of density gradient, compressibility and vorticity can be used quite strength forwardly to indicate the presence of shock waves, contact surfaces and turbulence. For reactive flows, temperature gradients and gradients in species mass fractions indicate flame fronts [2].

The adaptive mesh refinement guided by a physics-based refinement strategy compares a gradient of some solution quantity X to a refinement threshold value. If the

gradient measure, ϵ, given by $\epsilon \propto |\nabla X|$, is larger than a set refinement limit, the solution block is flagged for refinement and similarly, if the gradient measure is small, the blocks are flagged for refinement. As described before, the actual refinement process must also satisfy other criteria before the blocks actually undergoes mesh resolution change.

4.2.2 Smoothness Indicator-Based Refinement Criteria

For a finite-volume scheme of higher order than two, as described by Ivan [7], a measure of solution reconstruction smoothness of the proposed higher order central essentially non-oscillatory (CENO) reconstruction scheme was used to direct the mesh refinement. This criterion looks at local solution errors and guides the AMR to make the solution smoother by lowering the solution discontinuities across cell boundaries.

The smoothness indicator of Ivan [7] is calculated using the following expression:

$$S = \frac{\alpha(SOS - DOF)}{\max((1-\alpha), \varepsilon)(DOF - 1)}, \tag{4.2}$$

where SOS refers to the size of the reconstruction stencil, DOF means the degrees of freedom, or number of unknowns, and ε is a small constant to prevent division by zero.

$$\alpha = 1 - \frac{\sum_{\gamma}\sum_{\delta}\left(u^k_{\gamma,\delta}(\vec{\bar{r}}_{\gamma,\delta} - u^k_{i,j}(\vec{\bar{r}}_{\gamma,\delta})\right)^2}{\sum_{\gamma}\sum_{\delta}\left(u^k_{\gamma,\delta}(\vec{\bar{r}}_{\gamma,\delta} - \bar{u}^k_{i,j})\right)^2} \tag{4.3}$$

The γ and δ indices include all control volumes in the reconstruction stencil for cell (i, j). The term $u^k_{\gamma,\delta}$ is a k-exact polynomial solution reconstruction. The $\vec{r}_{\gamma,\delta}$ refers to the centroid of the (γ, δ) cell. The refining criteria is specified as:

$$R_c = e^{-\frac{max(0,S)}{U_S S_c}}, \tag{4.4}$$

where the U_S is a scaling coefficient for AMR adjustment and S_c is a cut-off value. The values R_c can take lie in the range (0,1]). The maximum value of R_c in each block is then used to determine whether refinement, coarsening, or no mesh density change should be applied.

4.2.3 Functional Error-Based Refinement Criteria

Adaptive mesh refinement that aims to minimise a specific functional error can use the error estimates defined in Section 2.2 to guide the AMR. The refinement criteria considered here provide the user freedom to chose the best criteria from particular error solution, or let the code automatically asses if each solution block should be refined based on all available criteria.

Refinement Criteria Based on Computable Correction for Advection-Diffusion Equation

The computable correction described by Equation (2.16) can be used as an indicator of the accuracy of the functional calculation on next level of mesh refinement. It estimates the error of functional on prolonged fine grid solution using adjoint weighted residuals as

$$\left|(\psi_h^H)^T R_h(u_h^H)\right| . \tag{4.5}$$

By minimising this error, the functional value will approach the true numerical value. This is done by refining the solution in areas where the computable correction is the largest. By making the computable correction smaller, the estimated fine grid functional is more accurate and is closer to the true functional value.

The solution block is selected for refinement if the computable correction value in any of the cells in the solution block has a higher value than specified refinement threshold. The alternative approach is to use the computable correction value for the entire block. The advantage of using values of individual cells is that there cannot be a block in the solution that has a poorly resolved solution, with relatively high errors in a small part of the solution and well resolved rest of the block. By looking at individual cell values, the refinement criteria is more aggressive to select block for flagging.

Refinement Criteria Based on Primal and/or Dual Error in Computable Correction

The use of error in computable correction, as defined by the primal error formulation

$$\left|(\psi_h|_{u_h^H} - \psi_h^H)^T R_h(u_h^H)\right| , \tag{4.6}$$

and dual error formulation

$$\left|\{R_h^\psi(\psi_h^H)\}^T(u_h - u_h^H)\right| , \tag{4.7}$$

can also be sued to minimise the errors in primal and dual adjoint solutions. By minimising the computable correction errors, both the functional and residual errors are minimised.

The three options for using these error estimates are to use either of the two estimates, or combine their value together. As with the computable correction criteria, the refinement criteria looks at the primal and/or dual computable, and combined values of error in correction to decide if any of them are above refinement threshold.

Refinement Criteria Based on Total Error Estimate for s Studied Functional

The total error of the estimated fine grid functional is defined as

$$\left|(\psi_h^H)^T R_h(u_h^H) + (\psi_h|_{u_h^H} - \psi_h^H)^T R_h(u_h^H)\right| , \tag{4.8}$$

This error estimate provides the best possible error estimate using the error formulations described in Chapter 3. The error values at each cell are compared to a set refinement threshold value. If the value is greater than the refinement threshold, the block that contains the inspected cell is selected for refinement.

Automatic Refinement Criteria Based on a Posteriori Adjoint Solution

The adaptive mesh refinement using *a posteriori* adjoint-based error estimates can be guided using all of the previously described refinement criteria. In this way, the refine-

ment process evaluates the particular functional error estimates for each cell against an appropriate refinement threshold value. If any of the calculated functional errors exceed the allowed level, the solution block containing the cell will then be refined. This procedure guarantees that none of the functional errors can be exceeded in any part of the solution. Note that the result sections of this thesis do not contain solutions obtained using this more comprehensive error flagging procedure. Instead, the individual refinement criteria have been evaluated separately.

Chapter 5

Numerical Results

5.1 Results Using a Posteriori Error Analysis

The *a posteriori* adjoint-based error estimation and block-based AMR strategy was tested by examining the application of the scheme to the solution of several problems for the advection diffusion equation on two dimensional, quadrilateral grids. The problems were selected to highlight the capability and behaviour of the refinement criteria.

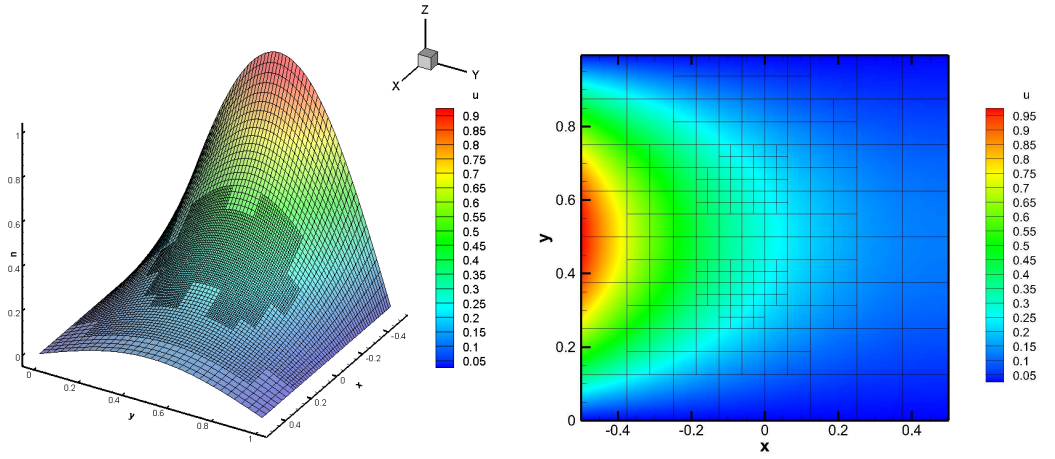
For the cases considered, the functional of interest was taken to be integral of the solution over a finite sub-region of the solution domain and given by

$$f = \iint_A u(x, y)\, dxdy\,, \tag{5.1}$$

where the area over which the integral is being evaluated A is the computational domain. For the cases considered, A was taken to be a circular region centred in the middle of the domain defined by a radius of $r \leq 0.201$.

5.1.1 Test Problem with Péclet Number of 1.09

The first test case for refinement criteria based on computable correction and errors in computable correction was set up on a rectangular grid with inflow on west boundary, outflow on east boundary and Dirichlet boundaries on north and south boundaries set to zero. The inflow has profile of sine wave on the interval $[0, \pi]$. The advection term for



(a) 3D view of $u(x, y)$. **(b)** Distribution of solution $u(x, y)$.

Figure 5.1

this problem was set to be $|\vec{V}| = 3.5$ in the positive x-direction and the diffusion constant was set to a value of $\kappa = 3.2$. Therefore, the Péclet number defined by Equation (2.4) has value of 1.09. Figure 5.1 shows the solution of the described problem in two and three dimensional views. The smooth profile and changing gradients of the solution provide a good opportunity to test the code as the residuals and solution reconstructions need to be performed on solution cells with various complexity of solution shape.

The converged solution is analysed and the errors are calculated based on the adjoint solution. According to the selection of a particular flagging method(s), the solution blocks are analysed and if selected for refinement, the AMR routine refines the grid and the solution is once more solved to a converged state.

Computable Correction Refinement Criteria

The computable correction is calculated as described in Chapter 4. Since the solution has a smooth, curved profile and both, the solution reconstruction and the prolongation operator in the adjoint calculation use linear-piecewise solution reconstruction, the solution residuals of the prolonged solution, computable corrections and errors in computable corrections are non-zero. Even for a fully converged solution, the residuals on the prolonged solution will not be at machine-zero level due to the second-order solution

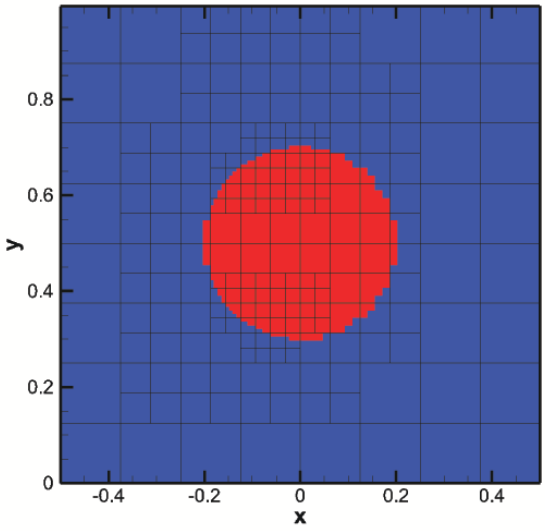


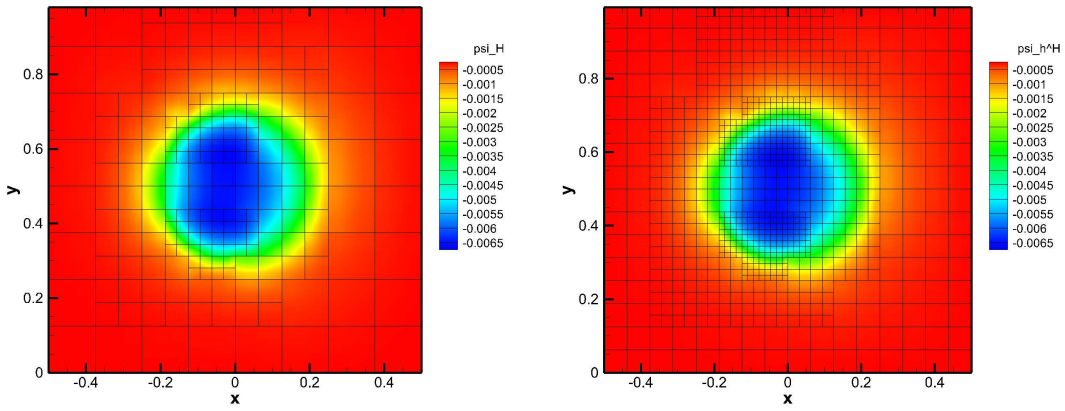
Figure 5.2: The functional of interest for the test cases (shown in red).

reconstruction scheme used.

The functional of interest for this problem is the value of the integral of u on a circular section at the center of the solution domain. The values of the term $\partial f_H/\partial \mathbf{U}_H$ in Equation (2.14) is shown on Figure 5.2. The high value depicted in red corresponds to the circular section of interest in which the integral is evaluated.

The adjoint problem solutions, for the described problem, on fine grid are shown on Figure 5.3. The prolonged solution of the functional is shown on Figure 5.6a. The estimated fine grid solution is not as accurate as a real fine mesh solution, but it is much cheaper to compute.

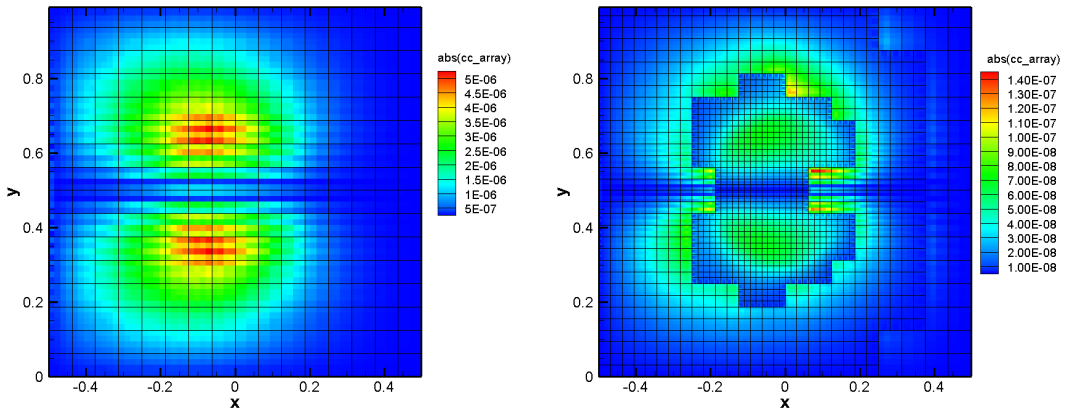
After obtaining the prolonged, fine grid solution, solving the adjoint problem and calculating the computable corrections for the numerical domain, it is possible to analyse the domain for the purposed of AMR guidance. The computable correction, as given by Equation (2.16), highlights regions of the computational domain, where the cells have relatively high residuals and the adjoint solution has significant values. In this case, it is expected to find higher residuals on the curved sides of the solution to the north and south of line $y = 0.5$, with diminishing values towards the east side of the domain. The adjoint solution is concentrated in the center of the domain, as expected and shown on Figure 5.3, so the expected regions of high computable correction values are located to the



(a) Adjoint solution on coarse mesh. **(b)** Prolonged adjoint solution.

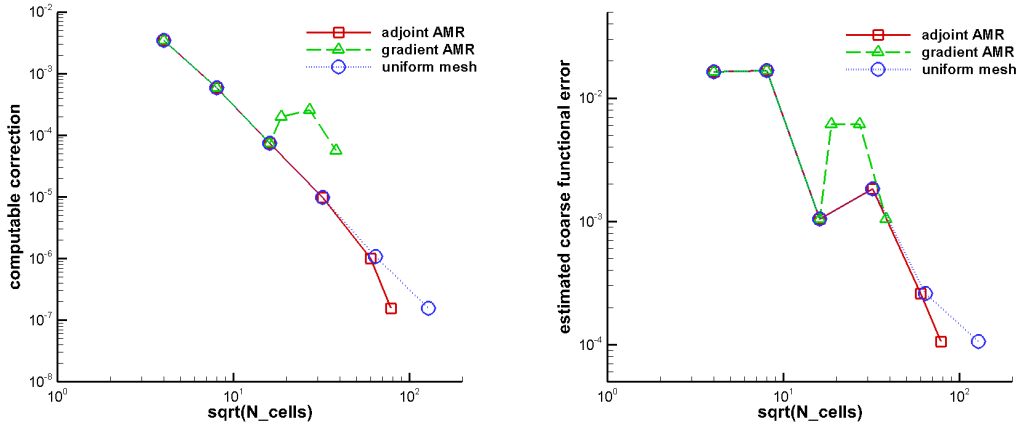
Figure 5.3: Adjoint solutions on coarse and fine grids.

north-west and south-west of the domain's center. The actual solution of the computable correction values confirm this analysis as shown on Figure 5.4. The regions that should be selected for refinement have the highest values of error. Therefore, the refinement criteria looks at the computable correction for each cell in the solution domain and if it is higher than initially set refinement threshold, the solution block that contains the cell is selected for refinement; if it can be legally refined by the AMR scheme.



(a) 4th level of refinement. **(b)** 6th level of refinement.

Figure 5.4: Computable correction values on different mesh densities.



(a) Computable correction at different refinement levels.

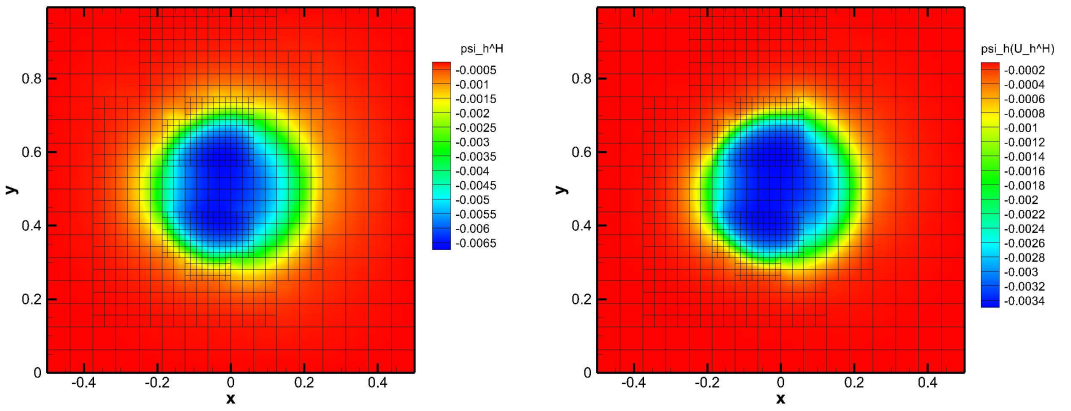
(b) Functional errors at different refnement levels.

Figure 5.5: Effects of AMR guided by computable corrections on functional.

The computable correction values on uniformly refined grid are shown on Figure 5.4a. As the solution grid is refined, the computable corrections become gradually smaller as shown on Figure 5.5b. Moreover, the functional value also converges closer to true value as the highest refinement level is increased. The estimated fine grid functional and the coarse mesh functional become more similar as the computable correction becomes smaller.

The block-based AMR strategy was applied to the first test problem from refinement level one to refinement level six using the computable correction refinement criteria with the threshold level set to 2×10^{-7}. The final adapted mesh had a refinement efficiency of 62.7% efficient and the relative error between the estimated fine mesh functional and the coarse mesh functional was 0.31%.

When compared to other more standard methods, the adjoint-based AMR method produced more accurate functional value than the gradient-based AMR. More importantly, the adjoint-based AMR produced results with accuracy of uniform mesh that has more than twice the computational cells. Moreover, the gradient-based AMR does not guarantee that the functional value and its errors will improve with each successive refinement.



(a) Prolonged adjoint solution.

(b) Adjoint solution on fine mesh using prolonged solution.

Figure 5.6: Estimated adjoint solutions on fine grid.

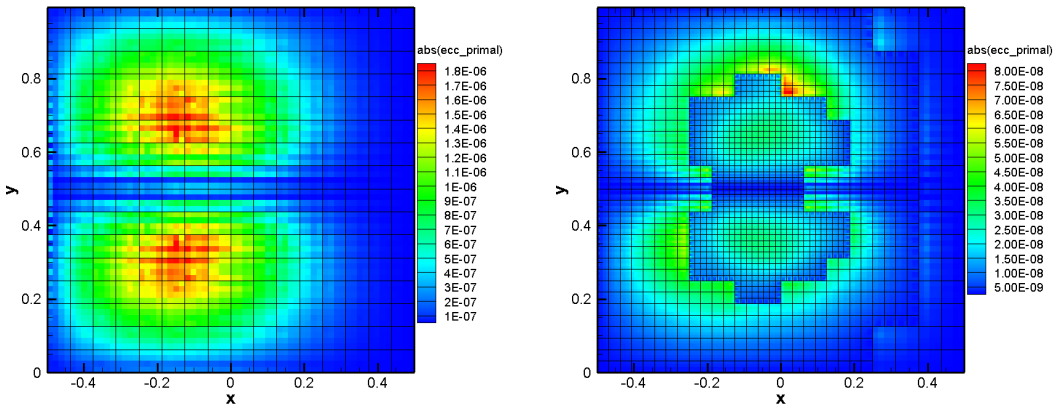
Error in Computable Correction Refinement Criteria

Just like the computable correction values can be used to guide adaptive mesh refinement, the errors in the computable corrections can be successfully used for this purpose. The primal errors are computed using Equation (2.16), while the errors using the dual formulation is described by Equation (2.19).

It can be seen that the solutions on Figure 5.6 are similar, but not exactly the same. This is important for the calculation of the error in computable correction as shown by Equation (2.16). Following the same reasoning as in the previous section, the critical regions of the solution to be candidates for mesh refinement are expected to be, just like for the computable corrections, in the north-west and south-west regions from the center of the solution domain.

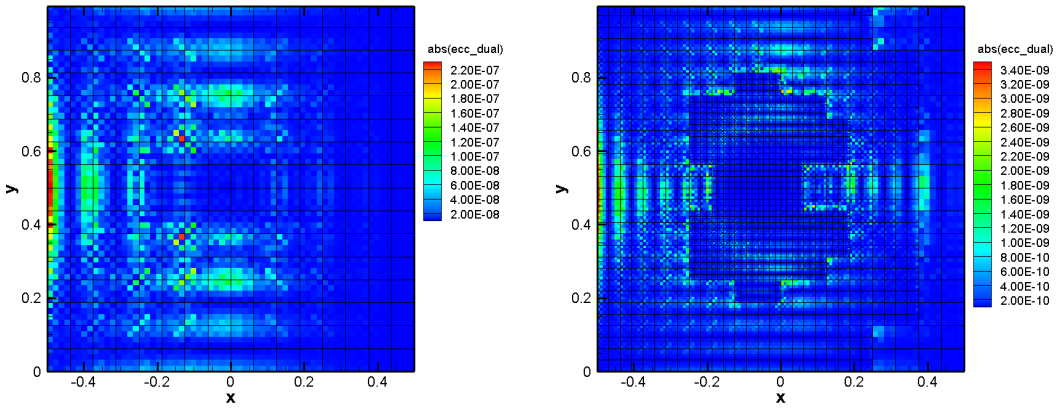
The regions of the solution domain that have the highest errors in the computable corrections correspond to the regions where the computable correction values have the highest magnitude. Therefore, both criteria select the same regions for adaptive mesh refinement.

The Figure 5.9 demonstrates that the error in computable correction is successfully reduced by the guided adjoint AMR and denser uniform mesh. Again, the adjoint-based



(a) 4th leel of refinement. **(b)** 6th level of refinement.

Figure 5.7: Error in computable correction values on different mesh densities.



(a) 4th leel of refinement. **(b)** 6th level of refinement.

Figure 5.8: Dual error in computable correction values on different mesh densities.

AMR shows better efficiency than the uniformly refined mesh. The gradient-based AMR is performing poorly and the reduction of the errors in computable correction is not guaranteed by successive mesh refinements.

The error in computable correction refinement criteria can be guided by either the errors from primal problem definition, or the dual problem definition, or both. The primal and dual errors have been observed to be similar in magnitude and location. The

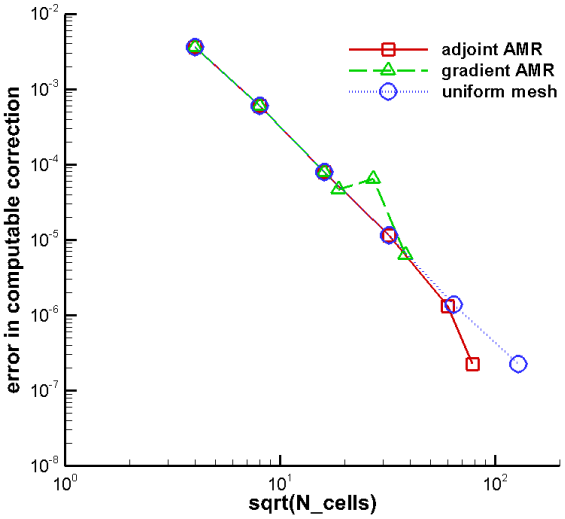


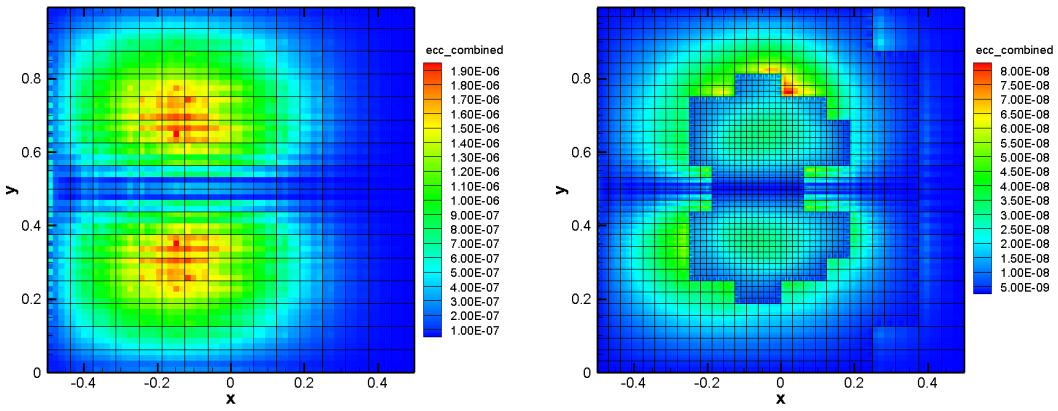
Figure 5.9: Effects of AMR on errors in computable corrections.

dual errors have not been as reliable as the primal errors and this issue is discussed in Chapter 6. Also, the dual errors are much more expensive to calculate than the computable correction of the primal variant of errors in computable corrections.

The Figure 5.8 illustrates the dual errors. It can be seen that they are not matching the primal errors perfectly, but the location of the largest errors is similar to those in primal errors of computable correction. The dual formulation of the errors is more sensitive to larger solution gradients that result in larger residuals.

The method proposed by Venditti and Darmofal [29] uses the combined errors in computable correction. This way, the errors caused by both residuals and adjoint errors are guaranteed to be minimised. The combined error is shown on Figure 5.10. The adaptive mesh refinement performed on the domain has successfully reduced the errors in both primal and dual errors in computable correction.

The numerical results of using errors in computable correction are very similar to the test case that used the computable correction refinement criteria. The criteria behave virtually identically for this test case if the refinement threshold values set so that they target the same numerical regions.



(a) 4th leel of refinement. **(b)** 6th level of refinement.

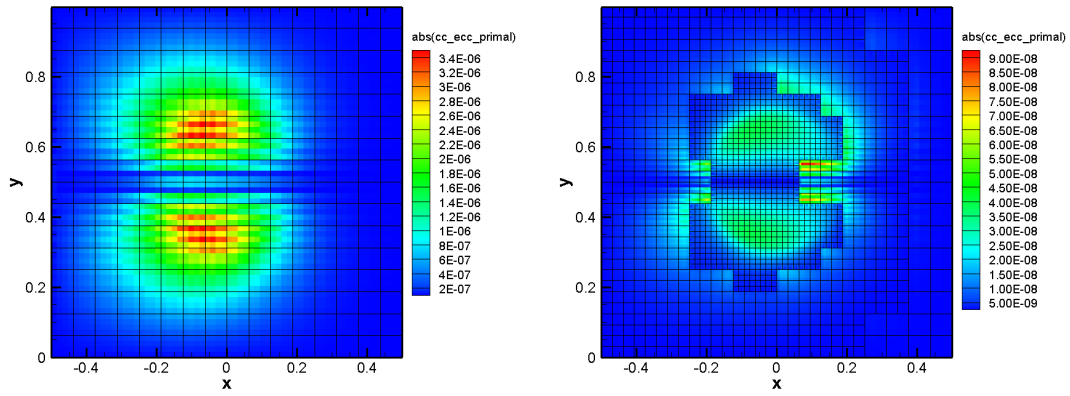
Figure 5.10: Combined, primal and dual, errors in computable correction on different mesh densities.

Refinement Criteria Based on Both Computable Correction and Error in Computable Correction

The error of the estimated fine grid functional value to the true fine grid functional is estimated by Equation (2.16). The criterion described in this section uses both calculated corrections to the estimated fine grid functional. The use of both computable correction and the error in computable correction aims to reduce the best estimate of the total calculated error for the estimated fine grid functional.

The combined computable correction and the error in computable correction shown on Figure 5.11 demonstrate the improvement of the error magnitude graphically on the solution domain. The Figure 5.12 shows the reduction of the functional estimated error in comparison with other refinement strategies.

It can be seen that the gradient-based AMR is not performing well and the adjoint-based AMR is showing greater efficiency than the uniform mesh refinement. The Figure 5.12 shows 63% less computational cells for the adjoint-based AMR as compared to uniformly refined mesh, yet the functional accuracy is of the same magnitude. This saving will be greatly improved on a test case that is not used primarily for code development with a large domain and a functional that focuses on a small portion of the



(a) 4th leel of refinement. **(b)** 6th level of refinement.

Figure 5.11: Computable errors on different mesh densities.

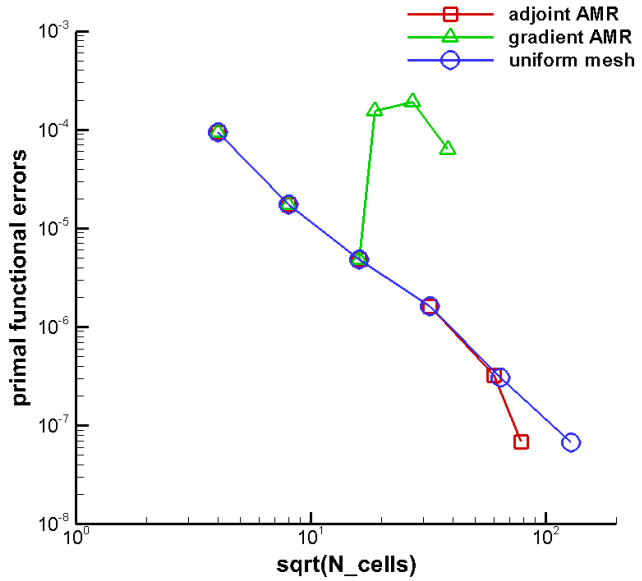
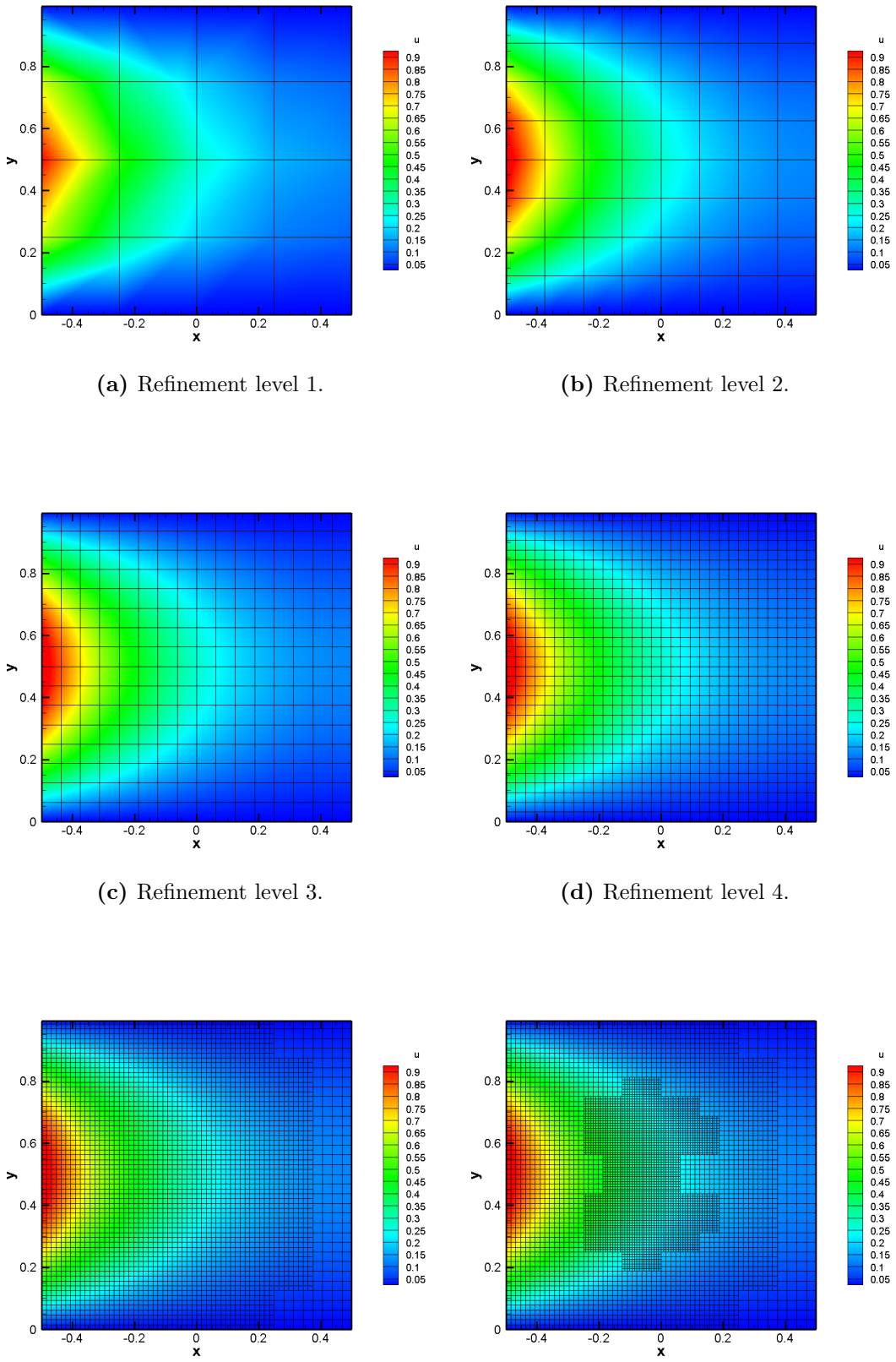


Figure 5.12: Effects of AMR on errors of fine grid functional estimate.

domain.

The solutions with mesh at various levels of refinement are shown on Figure 5.13.



(a) Refinement level 1.

(b) Refinement level 2.

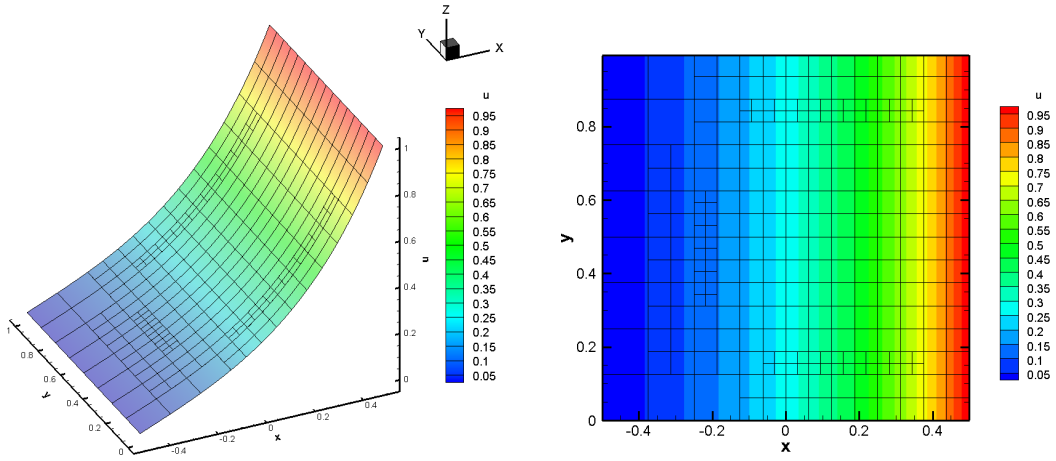
(c) Refinement level 3.

(d) Refinement level 4.

(e) Refinement level 5.

(f) Refinement level 6.

Figure 5.13: Comparison between the solutions and meshes during the AMR procedure.



(a) 3D view of $u(x, y)$. **(b)** Distribution of solution $u(x, y)$.

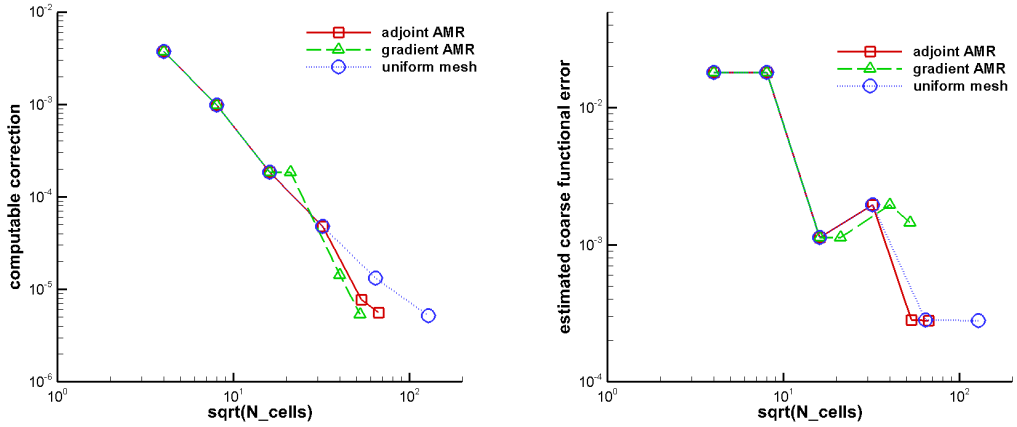
Figure 5.14

5.1.2 Test Case for a Diffusion Problem with a Linear Source

This test case involves a pure diffusion with a source. The problem is defined to have no gradient on solution at the north and south boundaries of the rectangular domain and Dirichlet boundary conditions on west and east sides of the domain. The west boundary is set to zero and east boundary to a value of 1. The source term is defined as $\phi(u) = \bar{u}/\tau$. The term $\bar{u}$ is the average cell solution value and the term τ is a constant. For this problem, τ was set to a value 1/6. The solution to this problem can be seen on Figure 5.14.

Just as for the advection-diffusion test case, the adaptive mesh refinement based on the a posteriori error estimates was compared to the gradient-based AMR and an uniform mesh refinement over six levels of refinement. The procedure was the same and the computable correction, errors in computable correction and the combined computable correction with its error was examined for AMR performance.

The shape of the solution is much simpler than the one in the advection-diffusion test case, but the surface has continually changing curvature. This means that the second order reconstruction method cannot fully describe any part of the solution along the x-axis. This is in contrast of the previous test case, where some parts of the solution can be better estimated using second order solution reconstruction. The expected higher



(a) Computable correction at different refinement levels.

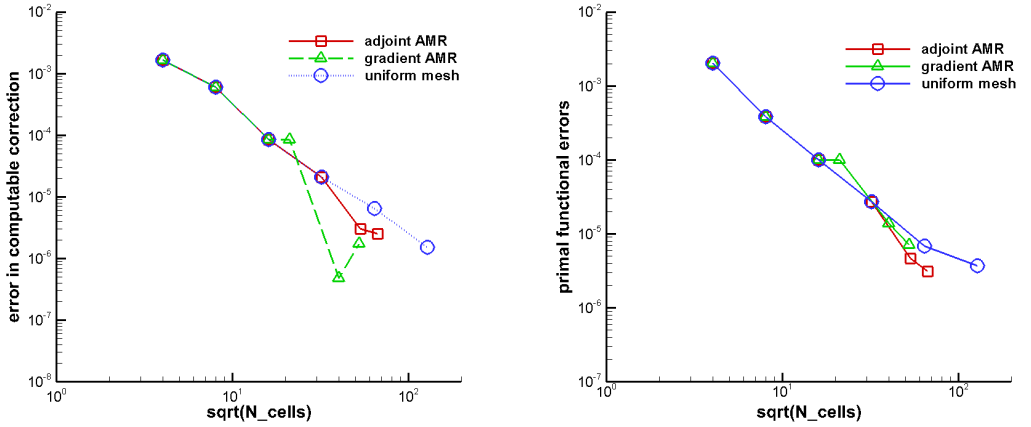
(b) Functional errors at different refnement levels.

Figure 5.15: Effects of AMR guided by computable corrections on functional.

residuals manifested themselves in higher minimal errors for this case when compared to the previous test case.

The overall performance of all studied mesh refinements can be seen on Figure 5.15. The gradient-based AMR was able to perform well in reducing the residuals in the domain and the Figure 5.15a clearly shows that the gradient-based AMR has the greatest efficiency for the studied problem. The gradient-based AMR is more efficient by 38% compared to adjoint-based AMR. The issue is that only the adjoint-based AMR and uniform mesh refinement were able to improve the fine functional estimate, as shown on Figure 5.15b. The gradient-based AMR was never able to bring the functional value to a converging value over the six levels of refinement. The difference between the coarse grid functional and the estimated fine grid functional was too great through out the refinements. The functional estimate of the adjoint-based AMR was able to converge quicker than the other methods and used less resources. The sixth lever of refinement for the gradient-based AMR uses about the same number of cells as the fifth level of adjoint-based refinement yet, the adjoint-based AMR has functional error better by an order of magnitude.

The Figure 5.16a shows the study of the error in computable correction over the studied six levels of refinement for all three refinement methods. The best performance



(a) Error in computable correction. **(b)** Total estimated fine functional error

Figure 5.16: Effects of AMR on errors of fine grid functional estimate.

may be attributed to the gradient-based AMR. However, the other methods have a clear and stable error development. The uniform mesh refinement and adjoint-based AMR positively reduce error in the computable correction with each level of refinement. The gradient-based method is quick to lower the error, but the next level of refinement may not necessarily improve the errors. At the highest level of refinement, the refinement methods were at similar level of error in the computable correction. The adjoint-based AMR was 73% more efficient than the uniform mesh refinement.

The error estimate, shown on Figure 5.16b, that uses both the computable correction and the error in computable correction seem to be the indicator of mesh refinement accuracy for this problem. The gradient-based AMR falls short of the other two methods and the adjoint-based AMR is more efficient than the uniform mesh refinement while providing comparable accuracy.

Chapter 6

Conclusions, Contributions and Future Research

The goal of this these research was to add *a posteriori* adjoint-based error estimation criteria to the adaptive mesh refinement capability of the CFFC numerical framework. As shown in Chapter 5, the overall goal was accomplished. The test cases demonstrate the new capability of the code to focus on mesh areas that are the most critical to accurate evaluation of observed functional. The existing, physics-based, refinement criteria would not refine the mesh in the test cases at all and no improvement in the functional accuracy would be possible. The new adaptive mesh refinement criteria include the choice of using the computable correction values, primal errors in computable corrections, the combined error estimate using the computable correction and error in computable correction, dual errors in computable corrections and the combined values of the errors in computable correction. Each criterion has its own threshold value that affects the overall AMR behaviour. Since the computable corrections are the cheapest to compute and seem to deliver the same quality of the refinement guidance as the criteria based on errors in computable correction, the best choice for simple advection-diffusion problems is to use the computable correction criterion without the calculation of the dual, or even primal error in computable correction. For estimating the accuracy of different refinement methods, the computable error with the primal error estimate seem to work reliably. The calculation of dual errors is much more expensive than the calculation of the primal errors.

The added benefit of having both error estimates may be realized for a more complex class of problems than studied here. Since, the observed differences between the primal and dual errors were minimal, the use of the dual formulation of the error in computable corrections may not be the most practical option for AMR guidance used with simple advection-diffusion problems.

The adjoint-based AMR was shown to perform reliably on all test cases and delivered functional estimate with accuracy comparable to uniform mesh refinement at a lower computational cost. The gradient-based AMR may improve the functional value in some cases, but this is not guaranteed. Therefore, adjoint-based AMR is a better alternative, as it guarantees that the mesh refinement will continue to improve the functional estimate to some converged value.

The test cases have small domain and the solution varies largely within the domain. Moreover, the functional focuses on a large part of the domain. This means that the adjoint-based AMR does not dramatically outperform the uniform mesh refinement. However, the test cases demonstrate that the *a posteriori* error estimates can be used to successfully guide AMR and the developed scheme performs as expected. A full potential of adjoint-based AMR would be seen, for example, on a pressure integral over a wing surface. Such problem involves a small, specialised area of interest in a large numerical domain. The practical problems however cannot be used for code development due to their high computational cost.

6.1 Contributions

The addition of the *a posteriori* adjoint-based error estimation capability to the CFFC code extended the capability of the numerical framework to better evaluate problems involving functionals. It is possible to estimate the functional errors and let the code to perform automatic adaptive mesh refinement to improve the functional accuracy. The added refinement criteria include all of the computed errors with individual refinement thresholds to give the code user the ability use a specific functional error as a refinement criteria, or let the code use all of the described refinement criteria to guide the AMR process. The adjoint-based error estimates have been shown to improve the functional

accuracy with each mesh refinement. Even on a small domain with a large area of interest, the AMR method using *a posteriori* adjoint-based error estimation showed accuracy level of a uniformly refined mesh at a lower computational cost.

6.2 Recommendations for Future Work

The work described in this work can be used as a basis for future work on *a posteriori* error estimation for Euler and Navier-Stokes equations in two and three dimensions. The implementation of adjoint based AMR can be beneficial for reactive flow simulations. It is also possible to extend the isotropic *a posteriori* error estimation AMR guidance to an anisotropic AMR guidance.

The current scheme has spatial accuracy of second order; however, the high-order finite-volume schemes are possible. Therefore, higher than second order *a posteriori* error estimation would be beneficial for the overall capability of the computational framework. The addition of high-order error estimation would permit the use of both p-refinement as well as h-refinement.

Bibliography

[1] Ramanathan, R. M., "Intel Multi-Core Processors, Making the Move to Quad-Core and Beyond," Tech. rep., Intel Corporation, 2006.

[2] Northrup, S. A., *A Parallel Implicit Adaptive Mesh Refinement Algorithm for Predicting Unsteady Fully-Compressible Reactive Flows*, Ph.D. thesis, University of Toronto, 2014.

[3] Groth, C. P. T. and Northrup, S. A., "Parallel Implicit Adaptive Mesh Refinement Scheme for Body-Fitted Multi-Block Mesh," Paper 2005-5333, AIAA, June 2005.

[4] Keyes, D. E., "A Science-Based Case for Large-Scale Simulation Volume 1," Report prepared for the Office of Science, U. S. Department of Energy, July 2003.

[5] Keyes, D. E., "A Science-Based Case for Large-Scale Simulation Volume 2," Report prepared for the Office of Science, U. S. Department of Energy, September 2004.

[6] Hartmann, R. and Houston, P., "Error estimation and adaptive mesh refinement for aerodynamic flows," .

[7] Ivan, L., *Development of High-Order CENO Finite-Volume Schemes with Block-Based Adaptive Mesh Refinement*, Ph.D. thesis, University of Toronto, October 2010.

[8] Versteeg, H. K. and Malalasekera, W., *An Introduction to Computational Fluid Dynamics. The Finite Volume Method.*, Pearson Education Limited, 2007.

[9] Berger, M. J., "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *Journal of Computational Physics*, Vol. 53, 1984, pp. 484–512.

[10] Berger, M. J., "Data Structures for Adaptive Grid Generation," Vol. 7, No. 3, 1986, pp. 904–916.

[11] Berger, M. J. and Colella, P., "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *Journal of Computational Physics*, Vol. 82, 1989, pp. 67–84.

[12] Powell, K. G., Roe, P. L., and Quirk, J., "Adaptive-Mesh Algorithms for Computational Fluid Dynamics," *Algorithmic Trends in Computational Fluid Dynmaics*, edited by M. Y. Hussaini, A. Kumar, and M. D. Salas, Springer-Verlag, New York, 1993, pp. 303–337.

[13] De Zeeuw, D. and Powell, K. G., "An Adaptively Refined Cartesian Mesh Solver for the Euler Equations," *Journal of Computational Physics*, Vol. 104, 1993, pp. 56–68.

[14] De Zeeuw, D. L., *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm for Solution of the Euler Equations*, Ph.D. thesis, University of Michigan, September 1993.

[15] Berger, M. J. and LeVeque, R. J., "An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries," Paper 89-1930, AIAA, June 1989.

[16] Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Base Geometry," Vol. 36, No. 6, 1998, pp. 952–960.

[17] Aftosmis, M. J., Berger, M. J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," Paper 2000-0808, AIAA, January 2000.

[18] Aftosmis, M. J., Berger, M. J., and Murman, S. M., "Applications of Space-Filling Curves to Cartesian Methods for CFD," Paper 2004-1232, AIAA, January 2004.

[19] Quirk, J. J. and Hanebutte, U. R., "A Parallel Adaptive Mesh Refinement Algorithm," Report 93-63, ICASE, August 1993.

[20] Berger, M. J. and Saltzman, J. S., "AMR on the CM-2," Vol. 14, 1994, pp. 239–253.

[21] Groth, C. P. T., Zeeuw, D. L. D., Powell, K. G., Gombosi, T. I., and Stout, Q. F., "A Parallel Solution-Adaptive Scheme for Ideal Magnetohydrodynamics," Paper 99-3273, AIAA, June 1999.

[22] Groth, C. P. T., De Zeeuw, D. L., Gombosi, T. I., and Powell, K. G., "Three-Dimensional MHD Simulation of Coronal Mass Ejections," Vol. 26, No. 5, 2000, pp. 793–800.

[23] Groth, C. P. T., De Zeeuw, D. L., Gombosi, T. I., and Powell, K. G., "Global Three-Dimensional MHD Simulation of a Space Weather Event: CME Formation, Interplanetary Propagation, and and Interaction with the Magnetosphere," Vol. 105, No. A11, 2000, pp. 25,053–25,078.

[24] Sachdev, J. S., Groth, C. P. T., and Gottlieb, J. J., "A Parallel Solution-Adaptive Scheme for Predicting Multi-Phase Core Flows in Solid Propellant Rocket Motors," *International Journal of Computational Fluid Dynamics*, Vol. 19, No. 2, 2005, pp. 157–175.

[25] Gao, X. and Groth, C. P. T., "A Parallel Adaptive Mesh Refinement Algorithm for Predicting Turbulent Non-Premixed Combusting Flows," *International Journal of Computational Fluid Dynamics*, Vol. 20, No. 5, 2006, pp. 349–357.

[26] Gao, X. and Groth, C. P. T., "A Parallel Solution-Adaptive Method for Three-Dimensional Turbulent Non-Premixed Combusting Flows," *Journal of Computational Physics*, Vol. 229, No. 5, 2010, pp. 3250–3275.

[27] Gao, X., Northrup, S. A., and Groth, C. P. T., "Parallel Solution-Adaptive Method for Two-Dimensional Non-Premixed Combusting Flows," Vol. 11, No. 2, 2011, pp. 76–95.

[28] Paillère, H., Powell, K. G., and De Zeeuw, D. L., "A Wave-Model-Based Refinement Criterion for Adaptive-Grid Computation of Compressible Flows," Paper 92-0322, AIAA, January 1992.

[29] Venditti, D. A. and Darmofal, D. L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," *Journal of Computational Physics*, Vol. 164, 2000, pp. 204– 227.

[30] Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, 2002, pp. 40– 69.

[31] Venditti, D. A. and Darmofal, D. L., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-Dimensional Viscous Flows," *Journal of Computational Physics*, Vol. 187, 2003, pp. 22– 46.

[32] Heuveline, V. and Rannacher, R., "Duality-Based Adaptivity in the *hp*-Finite Element Method," Vol. 11, 2003, pp. 1–18.

[33] Rannacher, R. and Vexler, B., "A Priori Error Estimates for the Finite Element Discretization of Elliptic Parameter Identification Problems with Pointwise Measurements," Vol. 44, 2005, pp. 1844–1863.

[34] Nemec, M. and Aftosmis, M. J., "Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," Paper 2007-4187, AIAA, June 2007.

[35] Nemec, M. and Aftosmis, M. J., "Adjoint sensitivity computations for an embedded-boundary Cartesian mesh method," *Journal of Computational Physics*, Vol. 227, No. 4, 2008, pp. 2724 – 2742.

[36] Nemec, M., Aftosmis, M. J., and Wintzer, M., "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries," Paper 2008-0725, AIAA, June 2008.

[37] Venditti, D. A. and Darmofal, D. L., "Anisotropic Adaptation for Functional Outputs of Viscous Flow Simulations," Paper 2003-3845, AIAA, June 2003.

[38] Godunov, S. K., "Finite-Difference Method for Numerical Computations of Discontinuous Solutions of the Equations of Fluid Dynamics," Vol. 47, 1959, pp. 271–306.

[39] Coirier, W. J., *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Michigan, 1994.

[40] Barth, T. J. and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes," Paper 89-0366, AIAA, January 1989.

[41] Venkatakrishnan, V., "On the Accuracy of Limiters and Convergence to Steady State Solutions," Paper 93-0880, AIAA, January 1993.

[42] Lomax, H., Pulliam, T. H., and Zingg, D. W., *Fundamentals of Computational Fluid Dynamics*, Scientific Computation, Springer, Berlin, 2001.

[43] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI*, MIT Press, Cambridge, Massachussets, 1999.

[44] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., Salinger, A. G., Thornquist, H. K., Tuminaro, R. S., Willenbring, J. M., Williams, A., and Stanley, K. S., "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, Vol. 31, No. 3, 2005, pp. 397–423.