5-2014

Numerical Examination of Flux Correction for Solving the Navier-Stokes Equations on Unstructured Meshes

Dalon G. Work
Utah State University

NUMERICAL EXAMINATION OF FLUX CORRECTION FOR SOLVING THE NAVIER-STOKES EQUATIONS ON UNSTRUCTURED MESHES

by

Dalon G. Work

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Dr. Aaron J. Katz Major Professor	Dr. Robert Spall Committee Member
Dr. Barton Smith Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2014

Abstract

Numerical Examination of Flux Correction for Solving the Navier-Stokes Equations on Unstructured Meshes

by

Dalon G. Work, Master of Science

Utah State University, 2014

Major Professor: Dr. Aaron J. Katz
Department: Mechanical and Aerospace Engineering

This work examines the feasibility of a novel high-order numerical method, which has been termed Flux Correction. It has been given this name because it "corrects" the flux terms of an established numerical method, cancelling various error terms in the fluxes and making the method higher-order. In this work, this change is made to a traditionally second-order finite volume Galerkin method. To accomplish this, higher-order gradients of solution variables, as well as gradients of the fluxes are introduced to the method. Gradients are computed using lagrange interpolations in a fashion reminiscent of Finite Element techniques. For the Euler Equations, Flux Correction is compared against Flux Reconstruction, a derivative of the high-order Discontinuous Galerkin and Spectral Difference methods, both of which are currently popular areas of research in high-order numerical methods. Flux Correction is found to compare favorably in terms of accuracy, and exceeds expectations for convergence rates. For the full Navier-Stokes Equations, the effect of curved elements on Flux Correction are examined. Flux Correction is found to react negatively to curved elements due to the gradient procedure's poor handling of high-aspect ratio elements.

(80 pages)

Public Abstract

Numerical Examination of Flux Correction for Solving the Navier-Stokes Equations on Unstructured Meshes

by

Dalon G. Work, Master of Science

Utah State University, 2014

Major Professor: Dr. Aaron J. Katz
Department: Mechanical and Aerospace Engineering

This work examines the feasibility of a novel high-order numerical method, which has been termed Flux Correction. This is accomplished by comparing it against another high-order method called Flux Reconstruction. These numerical methods are used to solve the Navier-Stokes equations, which govern the motion of fluid flow. High-order numerical methods, or those that demonstrate a third-order and higher solution error convergence rate, are rarely used on unstructured meshes when solving fluid problems. Flux Correction intends to make high-order accuracy available to the larger world of Computational Fluid Dynamics in a simple and effective manner. The advantages and disadvantages of the method can only be discovered when compared against other high-order numerical methods. This work accomplishes this by comparing Flux Correction and Flux Reconstruction in terms of accuracy, numerical dissipation, and solution times. Flux Correction is found to compare favorably in terms of accuracy, and exceed expectations for convergence rates. Flux Correction is also tested on high-order meshes, or meshes that use high-order polynomials in the construction of the unstructured triangle mesh. High-order meshes generate long, thin elements, which are found to negatively impact the convergence and accuracy of Flux Correction.

For my wife.
Here's to many more hectic and fun-filled years together.

For my God,
whose grace has given me a good life.

Acknowledgments

This work is supported by the Computational Research and Engineering for Acquisition Tools and Environments (CREATE) Program, which is sponsored by the U.S. Department of Defense HPC Modernization Program Office, and by the Army Research Office Fluid Dynamics Program.

I would like to thank my major professor, Dr. Aaron Katz, for taking a chance on me, and for patiently helping me through the learning process. I can only hope to be as smart as him one day.

A special thank you to my friend and mentor Kyle Horne. His enthusiasm for computational methods infected me during my undergraduate career, and placed me on the academic path I currently walk.

Also to my many friends and family who supported me, even if it was by giving me incredulous stares as I tried to explain what I was working on, then telling me that it sounds like great fun. They have spent many years teaching me how to laugh at myself, how to love, how to cry, and how to care. From such a large group of people I want to single out my parents, who have loved me, cared for me, prayed for me, taught me, and been there for me for over 25 years now. My life is what it is because of them.

Last of all I wish to thank my God and my wife. My Lord and Savior Jesus Christ has been my Exemplar, my Rock, and my Confidence since my teenage years. My wife has been the best as I spent many long hours on this project. I don't know why she would ever consent to marry and put up with a nerd like me. I will be eternally grateful for her faith in me, for her ability to remember the things that I forget, and for her amazing ability to pull my head out of the clouds and out of the ground. Here's to many more happy years together!

Dalon G. Work

Contents

List of Tables

List of Figures

Acronyms

BDF	Backward Difference Formula
BR	Bassi-Rebay
CDG	Central Discontinuous Galerkin
CFD	Computation Fluid Dynamics
CFL	Cauchy-Lewis Limit
CF	Central Flux
CG	Continuous Galerkin
DNS	Direct Numerical Simulation
DG	Discontinuous Galkerin
ENO	Essentially Non-Oscillatory
FC	Flux Correction
FD	Finite Difference
FE	Finite Elements
FR	Flux Reconstruction
FV	Finite Volume
IP	Internal Penalty
LDG	Local Discontinuous Galerkin
LES	Large Eddy Simulation
LHS	Left-Hand Side
MMS	Method of Manufactured Solutions
NDOF	Number of Degrees of Freedom
RHS	Right-Hand Side
RK	Runge-Kutta
RMS	Root Mean Square
SD	Spectral Difference

SUPG	Split-Upwind Pressure Galerkin
SV	Spectral Volume
VCJ	Vincent-Castonguay Jameson
WENO	Weighted Essentially Non-Oscillatory

Chapter 1

Introduction

Computational Fluid Dynamics (CFD) has become a mature, practical, and useful tool for design of fluid dynamic problems in industry. Many commercial and open source products are available to solve a variety of design issues. Indeed, many integrated product suites with a multitude of solver strategies, turbulence models, automated meshing of geometries, and even multiphysics capabilities are generally available. The general solutions available to the public today are generally second-order accurate in space and time, meaning these schemes usually fall into the broad categories of Finite Difference (FD), Finite Volume (FV), or Finite Element (FE) methods [1].

Second-order methods are generally adequate for many problems. However, there are many problems, rotorcraft design and flapping wings, for example, for which the established practices produce too much numerical dissipation and cannot accurately resolve vortex-dominated flows. Recent advances in turbulence modeling, including Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS), have been shown to require reduced numerical dissipation. In order to obtain realistic answers without excessive grid refinement, methods with a higher order of accuracy (third or greater) are required. While many higher-order methods have been and are being actively developed in academia, they have made very little progress into the professional engineering world. This disappointing result can be ascribed to a few reasons.

First, high-order methods, which generally are more unstable and mathematically stiff than their low-order counterparts, require more sophisticated solution techniques to keep them from diverging. Large amounts of work have been done on low-order schemes, rendering them very robust, in that they can be solved quickly for very complex problems, without risk of the solution being erroneous or diverging.

Second, high-order methods still do not have reliable techniques to handle strong discontinuities in the solution. It is well-known that high-order approximations suffer from Gibb's Phenomenon, or oscillations in the approximation that do not exist in the exact solution. This is especially true in high-gradient regions and discontinuities in the solution. Low-order methods provide many methods for capturing the interaction of shock waves without these oscillations.

Third, most high-order methods are different enough from their low-order cousins that implementing them would require a complete rewrite of a software package code base. Most software packages in use today have been around for a long time, using code that has been tweaked, optimized, debugged and polished for a particular methodology and work flow, encompassing many thousands of lines of code. Having to start from scratch to implement a more accurate method is a daunting proposition that requires enormous investments of time, energy, and capital.

While the previous hurdles are great, they are not insurmountable. The purpose of this thesis is to evaluate a new high-order method called Flux Correction (FC), which shows much promise in overcoming these hurdles. The Flux Correction Method uses error analysis to determine where the errors in a traditional, low-order method come from and then "upgrade" the method to a higher-order. For this work, this is done on a traditionally second-order node-centered Galerkin method and upgraded to a third-order accurate method.

The end result of FC is to add a correction to the numerical flux that defines the scheme. The correction, along with an additional gradient computation, is the only change to the traditional scheme. This allows a code base to be upgraded to third-order with a single subroutine call and placed in the appropriate location. This overcomes the third hurdle, as no major rewrites are necessary to a code base. As the traditional method remains essentially the same, the mature solution techniques and limiters can be applied to the method to enhance solution time and shock-capturing, thus we have overcome hurdles one and two.

This work will detail high-order methods on unstructured meshes instead of structured meshes. Unstructured meshes allow for the solution of differential equations on complex geometries by breaking the continuous domain Ω and its boundary Γ into a discrete representation of multiple irregularly sized elements or volumes Ω_k, each with its own boundary Γ_k. The differential equation is then solved on each element in some fashion. Often, numerical methods assume these elements are linear, consisting only of the necessary number of nodes to form the basic element shape. For example, a linear quadrilateral element consists of four nodes. It has been shown that high-order methods require high-order elements to produce accurate answers [2,3]. Curved elements allow for better approximation of the surface geometry, but can possibly lead to badly deformed elements or even invalid elements. To date, Flux Correction has not yet been tested with curved elements. The effect of curved elements on Flux Correction is tested in this work.

1.1 A Brief History of High-Order Methods

A few histories of high-order methods have been published in recent years. Two notable summaries are available by Wang [1] and Vincent and Jameson [4,5]. The summary by Wang is more of a history, with Jameson focusing on what work needs to be done in order to bring higher-order methods to a wider audience. Here, these works are summarized to provide background to the present thesis.

The oldest high-order schemes consist of Finite Difference (FD) methods, Finite Volume (FV) methods, and Finite Element (FE) methods. Finite Difference uses differential equations in a strong conservation law form, with the solution variables places along regular mesh lines. This form of numerical method is generally the first one taught in engineering courses and is highly intuitive. Designing and implementing high-order methods is relatively easy using FD. FD is not conducive to complex geometries, limiting it's usefulness to rather simple geometries. FD methods are not considered further because the topic of this work is evaluating high-order methods over complex geometries on unstructured meshes.

1.1.1 Finite Volume Methods

k-Exact

k-Exact (k-E) [6,7] Methods are a direct extension of Godunov-type FV methods. Each cell holds one solution average. A high-order form of the solution is then constructed within each cell based off the average values of a surrounding stencil. The method is discontinuous, and Riemann solvers are used at cell interfaces. Generally these are done at multiple points on each face, and high-order Gauss quadrature is used to numerically integrate the result. Integrated flux balances are then used to update the cell averaged solution. k-Exact methods are not spatially compact, and in 3D the memory requirements can become very large. They can be oscillatory around shocks, but limiters that lower the order of a cell may be applied.

ENO/WENO

Essentially Non-Oscillatory (ENO) [8, 9] and Weighted Essentially Non-Oscillatory Methods (WENO) [9–11] are very similar to k-Exact Methods. ENO methods form multiple solution polynomials and choose the "smoothest" one to represent the solution inside the cell. This is done to avoid discontinuities in the cell. WENO methods use multiple solution polynomials, but perform a weighted average of all the polynomials. Similarly, WENO methods are typically smoother and more accurate than ENO methods for a given mesh. They also exhibit better steady-state convergence. Limiters are naturally built-in to the methods as the multiple polynomials can be low-order ones.

1.1.2 Finite Element Methods

Continuous Galerkin

Continuous Galerkin Methods (CG) [12,13] are essentially high-order versions of traditional FE methods. They employ a higher order polynomial solution inside of the cells, with neighboring cells sharing the same solution value at the interface. This forms a globally coupled matrix that must be solved. These methods are compact in nature, and various

strategies have been introduced to decouple the solution and reduce the computational expense of a full matrix inversion. Like other high-order methods, capturing shocks can be difficult, and complicated weighting functions are often introduced to preserve upwind characteristics. Many methods have been formed, such as the Streamline Upwind Petrov-Galerkin (SUPG) [14], Galerkin Least-Squares, and Taylor Galerkin.

Discontinuous Galerkin

Discontinuous Galerkin Methods (DG) [15, 16] are similar to their continuous cousins, except that neighboring elements do not share solution values at the interface. By using Riemann solvers at the interface from FV methods, upwinding for hyperbolic systems can be easily implemented. High-orders are easily achievable by increasing the order of the interior polynomials. Their compact, local nature makes them easy to parallelize. These advantages have made DG very popular in the past ten years, and it is arguably the most well-known high-order method. The discontinuous nature of DG requires special treatment for the viscous terms, for which many solutions have been proposed.

1.1.3 Other Methods

This section describes other high-order methods that do not fall neatly into the traditional categories of high-order numerical methods.

Spectral Volume

Spectral Volume Methods (SV) [17, 18] borrows ideas from traditional Finite Volume methods and k-Exact schemes, and draws ideas from Discontinuous Galerkin methods as well. Each element is subdivided into sub-elements, with the finite volume equation being applied to each sub-element. The results are then used to construct a solution polynomial over the parent element. The method requires many integrations, making it very expensive. A quadrature-free version has been formulated though, mitigating this need. The methods are compact in nature, and solution limiting on individual sub-elements is possible to enhance shock resolution. It is also capable of leveraging many mature solution convergence

techniques due to its FV background.

Spectral Difference

Spectral Difference Methods (SD) [19,20] appear to be similar in spirit to FE methods, but they use the differential form instead of the integral weak form of the governing equations. In an element, solution points are defined where the solution is known. From these, an interpolating polynomial can be formed. "Flux" points are defined on the boundaries of the element. The solution is interpolated to the flux points, and approximate Riemann solvers are employed to evaluate the flux at element boundaries. The derivative of the flux is evaluated at the solution points from the interpolating polynomial. This method is a popular one, as it is easy to understand, compact, and has shown promise in capturing shocks and utilizing convergence acceleration techniques.

Flux Reconstruction

Flux Reconstruction (FR) [21, 22] is a modification of Spectral Difference methods. The fluxes are split into two components, a discretized flux and a correction flux. The discretized flux is completely local, and is defined only by the interpolating polynomial derived from the solution points. The correction flux is formed so that it will "lift" the fluxes defined on the boundary to the common interface flux computed by a Riemann Solver. The correction flux function will then propagate this movement inward into the element. The combination of the divergence of the discretized flux and the divergence of the correction flux is used to form a residual at the solution points of each element. Depending on the choice of correction function, FR has have been shown to reduce to a Discontinuous Galerkin Method or a Spectral Difference Method with an infinite number of combinations with good characteristics.

Flux Correction

Flux Correction (FC) [23,24] is wholly dissimilar to all methods previously encountered. The main premise of FC is that the global solution error is driven by the truncation error

of the method, and that by identifying the sources of the truncation error, modifications can be made to an existing method to "upgrade" it to be higher-order. So far, this idea has been used with a common second-order linear node-centered Galerkin method. By changing the interface flux definition for the Riemann Solver and using higher-order approximations for the solution gradients, the second-order method is upgraded to third-order for inviscid terms and fourth-order for viscous terms. The changes can be formulated as a "correction" to the interface flux definition, allowing easy insertion into an existing code. Because the fundamental nature of the method has not changed, mature limiters and convergence acceleration techniques are already available. Unlike many of the previous methods, the order of accuracy of FC is fixed by the accuracy of the gradients and the relationship between the truncation error and global error.

1.2 Purpose of Thesis

The purpose of this thesis is to evaluate the merits of the Flux Correction method by comparing with second-order methods and other high-order methods. There are three comparisons that can be done. The first is a mathematical validation using the Method of Manufactured Solutions [25]. This assures us of the validity of the method and implementation. The second is experimental verification, where we compare numerically derived results with well-known and established experimental data. This tells us that the method is capable of reproducing real-world data and gives us more confidence in the methods ability to predict the physical world. The third is a comparison against similar numerical methods in areas of convergence rates, time elapsed, stability, accuracy, and complexity. This last comparison highlights strengths and weaknesses in the compared numerical methods, offering possibilities where one might trump the other.

Flux Correction is not derived from any high-order parents, and thus has no obvious method to compare against. The Flux Reconstruction method was chosen as a comparison, as it recovers the both the DG and SD methods [26], which are the most actively researched high-order methods right now. Figure 1.1 shows the relationships between the various methods. FR is intuitive and lends itself to parallelization due to it's discontinuous nature

[27]. It has well-established stability properties and is actively being researched. Because of this, it was chosen as a basis for comparision with Flux Correction.

The objective of this thesis is twofold: First, to establish the viability of Flux Correction as a high-order, unstructured numerical method for CFD. This is accomplished by a comparision with Flux Reconstruction, an actively researched method at the time of writing. Viability is determined through metrics such as accuracy, convergence rates, and computation time. The second objective is to discover the effect of curved meshes on Flux Correction, whether they are detrimental or helpful to the method.

1.3 Thesis Outline

The rest of this thesis is outlined as follows. Chapter 2 covers necessary background topics, including the governing equations to be solved in this work, generation of curved meshes, and the Method of Manufactured Solutions. Chapter 3 derives the Flux Correction scheme from a second-order linear Finite Volume Galerkin Method, and explains the methods used in it's solution. Chapter 4 details the formulation of the Flux Reconstruction method, and briefly reviews it's history. Chapter 5 describes the test cases used to evaluate Flux Correction against Flux Reconstruction, as well as test the effects of mesh curvature. Chapter 6 then concludes this work with thoughts on future work for Flux Correction.

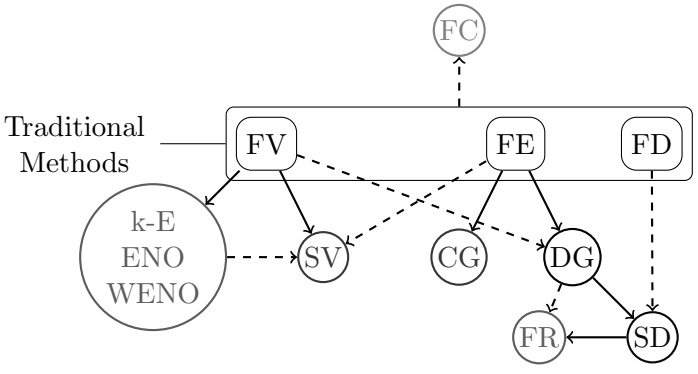


Fig. 1.1: Relationship graph for higher-order methods. The solid lines indicate a direct descent, while dashed lines indicate an indirect descent. The boldness of a method indicates it's relative popularity.

Chapter 2

Background

This chapter covers miscellaneous topics necessary for the rest of this work. These include a description of the governing equations to be solved, a description of the high-order meshing methods used, and a description of the Method of Manufactured Solutions.

2.1 Governing Equations

The equations that govern fluid motion follow from considering Conservation of Mass, Conservation of Momentum, and Conservation of Energy. This well-known result, known as the Navier-Stokes equations, is shown here in tensor notation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0, \tag{2.1a}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i + P\delta_{ij}}{\partial x_j} + \frac{\partial \sigma_{ij}}{\partial x_j} + \rho g_i = 0, \tag{2.1b}$$

$$\frac{\partial \rho e}{\partial t} + \frac{\partial \rho h u_j}{\partial x_j} - \frac{\partial \sigma_{ij}}{\partial x_j} u_i - \frac{\partial q_j}{\partial x_j} - \rho g_j u_j = 0. \tag{2.1c}$$

We define P as the thermodynamic pressure, g_i as the ith component of the body force, e as the total energy (internal plus kinetic) per unit mass and $h = e + \frac{P}{\rho}$ as the enthalpy. q_j is the jth component of the heat flux vector. This can be related to temperature through Fourier's Law of Heat Conduction:

$$q_j = -\kappa T_{x_j}, \tag{2.2}$$

where κ is the thermal conductivity, which in general is dependent on temperature.

The only assumption that has been made so far is that the fluid is a continuum, thus neglecting the molecular nature of the fluid. For this work, we will also neglect gravity. This is a good approximation for fluids with low densities and small vertical scales. Since this work focuses on aerodynamic applications, our working fluid will be air, with little vertical

change. Next, we will assume the fluid is Newtonian. This assumes the stress in the fluid is linearly proportional to the strain in the fluid, and determines the form of σ_{ij}. Third, for this thesis these equations will only be considered in two dimensions.

The equations as they have been given are not very convienent for numerical computation, and can be written as an advection-diffusion equation:

$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{F_i} - \nabla \cdot \mathbf{F_v} = \mathbf{S}. \tag{2.3}$$

The conserved variable vector q is given as:

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{pmatrix}. \tag{2.4}$$

The inviscid flux vector is $\mathbf{F_i} = \langle f_i, g_i \rangle$, where

$$f_i = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uh \end{pmatrix}, \qquad g_i = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vh \end{pmatrix}, \tag{2.5}$$

The viscous flux vector is $\mathbf{F_v} = \langle f_v, g_v \rangle$, with

$$f_v = \begin{pmatrix} 0 \\ \sigma_{xx} \\ \sigma_{xy} \\ u\sigma_{xx} + v\sigma_{xy} - q_x \end{pmatrix}, \qquad g_v = \begin{pmatrix} 0 \\ \sigma_{yx} \\ \sigma_{yy} \\ u\sigma_{yx} + v\sigma_{yy} - q_y \end{pmatrix}, \tag{2.6}$$

where the Newtonian stress tensor is written as:

$$\sigma_{xx} = 2\mu u_x - \frac{2}{3}\mu\left(u_x + v_y\right), \tag{2.7a}$$

$$\sigma_{yy} = 2\mu v_y - \frac{2}{3}\mu\left(u_x + v_y\right), \tag{2.7b}$$

$$\sigma_{xy} = \sigma_{yx} = \mu\left(u_y + v_x\right), \tag{2.7c}$$

where u_x indicates the partial derivative of u with respect to x, or $\frac{\partial u}{\partial x}$.

2.2 High-Order Meshing

Meshing a 2D continuous domain has generally consisted of using linear or quadratic quadrilaterals and triangles. The enhanced accuracy of higher-order methods means that coarser meshes may be used to achieve the same accuracy. The coarser meshes could no longer approximate the surface in a reasonable fashion, leading to a reduction in accuracy. The fact that higher-order meshes are necessary for higher-order schemes has been shown by various authors [2, 3].

In this work, we test Flux Correction using linear and cubic triangular unstructured meshes. This section describes how the meshes used were generated.

Dey [28] suggests two routes that can be taken when generating high-order meshes. The first approach involves generating a high-order surface tessellation from the exact surface definition and then forming the volume mesh around the high-order surface tesselation. This is referred to as the "direct approach."

The second approach is to start with a linear volume mesh, with nodes located on the exact surface definition. Surface faces are then curved in some fashion to approximate the given surface. Curving the faces can be done using the original surface definition, or using interpolating splines to approximate the surface. Dey calls this the "*a posteriori* approach." The advantage of this method is that generating the initial mesh can be done using available and robust software. The process of curving the surface can lead to invalid elements, where the mesh crosses over itself. Steps must then be taken to either split the

invalid element into valid ones, or to curve the invalid element so that it is no longer invalid. This is the approach taken for this work. While many approaches to remove invalid elements have been proposed, we follow the approach presented by Allen [29]. This approach was originally formulated for moving meshes, but can be applied to grid generation.

First, the nodes on the moving surfaces must be identified. A moving surface is the actual surface geometry of interest. For instance, a 3-element airfoil consists of 3 moving surfaces. Several geometric and weighting factors are then defined for each node p not on a moving surface:

$$0 \leq \alpha_{nc}^{p,ns} \leq 1 \tag{2.8a}$$

$$S_{nc}^{p,ns} = \left| \mathbf{r}^P - \mathbf{r}_{connect(nc)}^{p,ns} \right| \tag{2.8b}$$

$$S_F^p = \left| \mathbf{r}^p - \mathbf{r}_{farfield}^p \right|. \tag{2.8c}$$

These are subject to the condition

$$\sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} = 1 \quad ns = 1...nsurfaces, \quad nc = 1...nconnect. \tag{2.9}$$

In equations (2.8c) and (2.9), $\mathbf{r}^p$ is the global position vector of the current node, $\mathbf{r}_{connect(nc)}^{p,ns}$ is the position vector of the nc-th connection point on moving surface ns for node p. In the following, $\mathbf{r}^p$ refers to the initial position of point p, while $\mathbf{r}^p(t)$ refers to the adjusted position vector.

A distance function for each moving surface ns is defined for each point p as:

$$\psi^{p,ns} = \frac{\sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} S_{nc}^{p,ns}}{S_F^p + \sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} S_{nc}^{p,ns}} \quad ns = 1...nsurfaces. \tag{2.10}$$

Allen suggests using $nconnect = 2$ for 2D, where the two connections on a moving surface ns are the two points on the surface that are closest to the node p in question. Each moving surface should affect the position of node p. This necessitates the use of a weighting function

to combine the effects of all the moving surfaces on point p. These are described as:

$$S^{p,ns} = \sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} S_{nc}^{p,ns} \quad ns = 1...nsurfaces, \tag{2.11}$$

$$S_{min}^{p} = \min\left(S^{p,1}, S^{p,2}, ..., S^{p,nsurfaces}\right), \tag{2.12}$$

$$S_{surface}^{p,ns} = \frac{S_{min}^{p}}{S^{p,ns}} \quad ns = 1...nsurfaces, \tag{2.13}$$

$$S_{Total}^{p} = \sum_{ns=1}^{nsurfaces} \left(S_{surface}^{p,ns}\right)^{ssc}, \tag{2.14}$$

$$\phi^{p,ns} = \frac{S_{surface}^{p,ns}}{S_{Total}^{P}} \quad ns = 1...nsurface, \tag{2.15}$$

where ssc is a scaling component and typically takes a value of 2. The previous equations describe the *translation* component of the mesh movement scheme. This does not preserve orthogonality in the moving mesh, and the procedure must also take into account the rotation of the surface movement in order to preserve the orthogonality of the mesh. This is possible as long as $nconnect \geq 2$.

A translation vector is defined for each point p based off the translation vectors of its connection points as follows:

$$\Delta\mathbf{r}_{T}^{p,ns} = \sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} \Delta\mathbf{r}_{connect(nc)}^{p,ns}, \tag{2.16}$$

where $\Delta\mathbf{r}_{connect(nc)}^{p,ns}$ is the displacement vector of connection point nc on surface ns, or $\mathbf{r}(t) - \mathbf{r}(0)$. A rotation vector, referenced from an arbitrary origin, can be defined for a node

p as:

$$\Delta \mathbf{r}_R^{p,ns} = (R^{p,ns} - I)\left(\mathbf{r}^p - \sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} \mathbf{r}_{connect(nc)}^{p,ns}\right), \tag{2.17}$$

$$R^{p,ns} = \begin{pmatrix} \cos\theta^{p,ns} & \sin\theta^{p,ns} \\ -\sin\theta^{p,ns} & \cos\theta^{p,ns} \end{pmatrix}, \tag{2.18}$$

$$\theta^{p,ns} = \sum_{nc=1}^{nconnect} \alpha_{nc}^{p,ns} \theta_{connect(nc)}^{p,ns}, \tag{2.19}$$

$$\theta_{connect(nc)}^{p,ns} = \arccos\left(\frac{\mathbf{r}_{connect(nc)}^{p,ns} \cdot \mathbf{r}_{connect(nc)}^{p,ns}(t)}{\|\mathbf{r}_{connect(nc)}^{p,ns}\| \|\mathbf{r}_{connect(nc)}^{p,ns}\|}\right), \tag{2.20}$$

where I is the identity matrix. Once the rotation and displacement vectors have been defined for a point, the total movement of a node p can be found by

$$\mathbf{r}^p(t) = \mathbf{r}^p + \sum_{ns=1}^{nsurfaces} \phi^{p,ns}\left(\Delta \mathbf{r}_T^{p,ns}\left(1 - \psi^{p,ns}\right)^{st} + \Delta \mathbf{r}_R^{p,ns}\left(1 - \psi^{p,ns}\right)^{sr}\right), \tag{2.21}$$

with st and sr being scaling components for the translation and rotation portions, respectively. These control how far into the mesh the displacements are propogated.

While the scheme outlined above was originally developed for moving surface meshes, it can be applied to mesh generation to prevent invalid elements in the *a posterioir* approach. First, a linear mesh is formed using available software. Linear edges along the surface geometry in question are made higher order by adding interior nodes at regular intervals along the length of the edge. This defines the original position of the connection points, $\mathbf{r}_{connect(nc)}^{p,ns}$. The interior nodes are then "snapped" to the exact surface definition, which becomes the new timestep $\mathbf{r}_{connect(nc)}^{p,ns}(t)$. For consistency, interior nodes are added to all edges in the mesh. The algorithm described above is then applied to all nodes (excepting the surface nodes), linear and interior.

For cubic and higher meshes, nodes interior to the cells are also necessary. These nodes are *not* moved according to the above scheme as this will lead to non-linearity inside elements [30]. In order to place these correctly, we utilize the methodology illustrated by

Solin [31].

For triangles, the nodes interior to the element are first located *as if the element was a linear element* and then nudged into the necessary position by the deformed edges. Mathematically, this looks like:

$$\mathbf{x}^p = \mathbf{x}_L^p + \mathbf{x}_e^p, \quad \mathbf{x}^p = \langle x^p, y^p \rangle \,. \tag{2.22}$$

The initial placing is done by mapping the element to a reference element, placing the interior nodes, and then mapping them back to the physical element as if it were a linear element. This comprises the first term on the RHS, $\mathbf{x}_L^p$. The second term adds in the non-linearity from the edges as:

$$\mathbf{x}_e^p = \sum_{j=1}^{3} \mathbf{x}_{e_j}^{int}(\zeta)\, \lambda_A(\mathbf{r}) \lambda_B(\mathbf{r}), \tag{2.23a}$$

$$\mathbf{x}_{e_j}^{int}(\zeta) = \frac{\Delta \mathbf{x}_{e_j}(\zeta)}{\frac{1}{4}\,(1-\zeta)\,(1+\zeta)}, \quad \zeta \neq \pm 1, \tag{2.23b}$$

$$\Delta \mathbf{x}_{e_j}(\zeta) = \mathbf{x}_{e_j} - \frac{1}{2}\,[1-\zeta]\,\mathbf{x}_{e_j}(\zeta = -1) - \frac{1}{2}\,[1+\zeta]\,\mathbf{x}_{e_j}(\zeta = 1), \tag{2.23c}$$

$$\zeta = \lambda_B(\mathbf{r}) - \lambda_A(\mathbf{r}). \tag{2.23d}$$

$\mathbf{x}_{e_j}^{int}$ is defined to be zero at $\zeta = \pm 1$. The λ functions are the linear mapping functions associated with the nodes of edge j. For example, the linear mapping to an equilateral triangle is:

$$\mathbf{x}(\mathbf{r}) = \frac{1}{6}\left(-3r + 2 - \sqrt{3}s\right)\mathbf{x}_1 + \frac{1}{6}\left(3r + 2 - \sqrt{3}s\right)\mathbf{x}_2 + \frac{1}{6}\left(2 + 2\sqrt{3}s\right)\mathbf{x}_3. \tag{2.24}$$

For edge 1 (comprised of nodes 1 and 2) of this particular mapping,
$\lambda_A = \frac{1}{6}\left(-3r + 2 - \sqrt{3}s\right)$ and $\lambda_B = \frac{1}{6}\left(3r + 2 - \sqrt{3}s\right)$. Subtracting λ_B and λ_A maps the $\mathbf{r}$ location of the interior point to the parameter ζ of the edge.

The function $\mathbf{x}_{e_j}$ is the parametric equation describing edge j. In lieu of an exact equation, a polynomial fit through the high-order edge of order h can be formed via a

Vandermonde matrix:

$$\begin{pmatrix} 1 & \zeta_0 & \zeta_0^2 & \cdots & \zeta_0^h \\ 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^h \\ \vdots & & \ddots & \ddots & \vdots \\ 1 & \zeta_h & \zeta_h^2 & \cdots & \zeta_h^h \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_h \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_h \end{pmatrix}, \tag{2.25}$$

where $-1 \leq \zeta \leq 1$. The polynomial coefficients c can be found by solving the system given in equation (2.25) and then used in $\mathbf{x}_{e_j}$.

2.3 Verification with the Method of Manufactured Solutions

The Method of Manufactured Solutions (MMS), is a method of verifying that the implementation of a given algorithm is free of coding errors and gives the expected order of accuracy. It was originally detailed by Roache [25]. The method consists of choosing an exact solution and then substituting it into the differential equation to be solved. This determines a source term that will force the solution to the chosen solution.

The algorithm will solve toward the exact solution, with some error. A grid refinement study can then be used to determine the order of accuracy of the method.

In the case of a complex set of equations, such as the Navier-Stokes equations, the exact solution must be carefully chosen so as to exercise all terms in the equation. The solution must also be "difficult" enough that the algorithm cannot solve it exactly, while not becoming unsolvable.

The chosen MMS solution for density in this work is shown here:

$$\rho = \rho_0 + \rho_x \sin\left(\frac{\alpha_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{\alpha_{\rho y}\pi y}{L}\right) + \rho_{xy} \cos\left(\frac{\alpha_{\rho xy}\pi xy}{L^2}\right). \tag{2.26}$$

Here ρ_0 is the freestream value, and ρ_* and α_* are arbitrarily chosen constants. Similar solutions can be found for u, v, and P, and used as source terms in the conserved variables for equation (2.3).

Chapter 3

Flux Correction

The goal of Flux Correction is to cancel the leading order error terms of an already existing numerical method, thus "upgrading" it to a higher order of accuracy. In order to accomplish this, where the truncation error terms arise in a method must be understood.

In this work, we examine a common linear Galerkin node-centered method which is traditionally second-order and "upgrade" it to be third-order accurate. The discretization of the Galerkin method is well-known [32]. Since the Galerkin method itself is not the focus of this work, it will not be detailed here. For the sake of completeness, it is given in the Appendix.

Before describing FC, a brief discussion of error terms is necessary to understand the justification used in the development of the method. Following that, the method will be demonstrated in 1D for understanding, leading then to the 2D formulation. This chapter follows the methodology outlined by Katz, Sankaran, and Pincock [23, 24].

3.1 The Effect of Truncation Error on Solution Error

Truncation error arises from the discretization of a continuous differential equation. Importantly, truncation error is distinct from solution error, which is defined as the difference between the true solution and the converged discretized solution. In the end, it is the solution error that we are concerned about. As the mesh is refined, the solution error drops. The rate at which the solution error drops leads to the notion of order of accuracy. The order of accuracy of the truncation and solution error are not necessarily the same, but are related.

To understand the effect truncation error has on the solution error, consider a general conservation law:

$$q_t + \nabla \cdot \mathbf{F} = 0. \tag{3.1}$$

q represents conserved variables, $\mathbf{F}$ is the flux. For linear $\mathbf{F}$, the discretization of equation (3.1) at steady-state becomes:

$$\mathcal{D}\left\{q^h\right\} = Bq_b, \tag{3.2}$$

where $\mathcal{D}$ is the discretization operator, and B incorporates the boundary conditions q_b. The discrete solution exactly satisfies this algebraic system of equations. If the exact solution q is substituted into the left-hand side (LHS) of equation (3.2), an error term must be added to the right-hand side (RHS):

$$\mathcal{D}\{q\} = Bq_b + E_t, \tag{3.3}$$

where E_t is the truncation error.

The solution error E_s is, by definition, the exact solution minus the discrete solution, $E_s = q - q^h$. Rearranging equations (3.2) and (3.3) and substituting into the solution error definition yields:

$$\mathcal{D}\{E_s\} = E_t. \tag{3.4}$$

From this we find that the truncation error and the solution error are related through the discretization operator. This can be viewed as the truncation error driving the solution error. Unfortunately, there does not appear to be anyway to prove the order of the solution error from the truncation error. Numerical observations have shown that the order of the solution error is never lower than the order of the truncation error.

3.2 1D Flux-Correction

We wish to understand the origins of the errors in the Galerkin version of the hyperbolic equation

$$q_t + f_x = S(x). \tag{3.5}$$

First, using the 1D discretization in figure 3.1 we can define the following:

$$\Delta x_i = \frac{1}{2}\left(\Delta x_{i-1/2} + \Delta x_{i+1/2}\right), \quad \Delta x_{i+1/2} = x_{i+1} - x_i. \tag{3.6}$$

The discretization of this equation in a Galerkin fashion leads to a discretized flux derivative at node i (shown in figure 3.1), shown here:

$$f^h_{x,i} = \frac{1}{\Delta x_i}\left(F^h_{i+1/2} - F^h_{i-1/2}\right). \tag{3.7}$$

The superscript h represents a discretized value, and not an exact value. The source term discretization is given by:

$$S^h_i = \frac{2}{3}S_i + \frac{1}{6\Delta x_i}\left(S_{i-1}\Delta x_{i-1/2} + S_{i+1}\Delta x_{i+1/2}\right). \tag{3.8}$$

This is the point where the traditional Galerkin and the Flux Correction methods diverge. We will cover the traditional first to understand where the truncation error comes from, followed by the Flux Correction method.

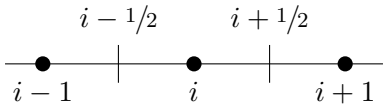


Fig. 3.1: Discretization used in the derivation of the Linear Galerkin Method

3.2.1 Traditional Galerkin

In order to proceed, we need to find a useful way to compute the fluxes at the halfway points, $F_{i-1/2}$ and $F_{i+1/2}$. In the traditional method, this is done using

$$F^h_{i+1/2} = \frac{1}{2}\left(f^h_i + f^h_{i+1}\right) - D^h_{i+1/2}, \tag{3.9a}$$

$$D^h_{i+1/2} = \frac{1}{2}\left|A^h\left(q^R_{i+1/2}, q^L_{i+1/2}\right)\right|\left[q^R_{i+1/2} - q^L_{i+1/2}\right], \tag{3.9b}$$

where $A^h = \frac{\partial f}{\partial q}$ and is a function of the reconstructed state of q^h at the midpoint. The dissipation term, shown in equation (3.9b), is added to enforce upwinding, making the method numerically stable. Substituting equation (3.9) into equation (3.7), the flux derivative at node i becomes:

$$f^h_{x,i} = \frac{1}{2\Delta x_i}\left(f^h_{i+1} - f^h_{i-1}\right) - \frac{1}{2\Delta x_i}\left(D^h_{i+1/2} - D^h_{i-1/2}\right). \tag{3.10}$$

The method is now in a form that is convenient to determining the truncation error. The two terms of the RHS of equation (3.10) will be addressed separately, starting with the flux term. The source terms will follow.

Substituting a Taylor series, centered on node i for the exact flux f:

$$\begin{aligned}\frac{1}{2\Delta x_i}\left(f^h_{i+1} - f^h_{i-1}\right) = f_{x,i} &+ \frac{1}{2}\left(\Delta x_{i+1/2} - \Delta x_{i-1/2}\right) f_{2x,i} \\ &+ \frac{1}{12\Delta x_i}\left(\Delta x^3_{i+1/2} + \Delta x^3_{i-1/2}\right) f_{3x,i} \\ &+ O\left(h^3\right).\end{aligned} \tag{3.11}$$

Equation (3.11) is first-order accurate for irregular grids and second-order for regular grids (which cancels the second term on the RHS). We now treat the numerical dissipation term in equation (3.10). q^L and q^R are the estimated values of q at $x_{i+1/2}$ approaching from the left and right sides, respectively. These can be estimated in a linear fashion by extrapolating them using a truncated Taylor series expansion, shown here:

$$q^L_{i+1/2} = q^h_i + \frac{1}{2}\Delta x_{i+1/2} q^h_{x,i}, \qquad q^R_{i+1/2} = q^h_{i+1} - \frac{1}{2}\Delta x_{i+1/2} q^h_{x,i+1}. \tag{3.12}$$

The gradient can be computed any convenient manner. For now, let us say that the gradient $q^h_{x,i}$ approximates the true gradient with an error of order p.

$$q^h_{x,i} = q_{x,i} + O(h^p). \tag{3.13}$$

Substituting in the definition of the dissipation from equation (3.9b), the exact solution of q, and inserting the arbitrarily accurate derivative from equation (3.13), we find the truncation error for the dissipation terms to be:

$$\begin{aligned}\frac{1}{2\Delta x_i}\left(D_{i+1/2} - D_{i-1/2}\right) = -\frac{1}{48\Delta x_i}\left(\Delta x^3_{i+1/2}\left|A_{i+1/2}\right| - \Delta x^3_{i-1/2}\left|A_{i-1/2}\right|\right) q_{3x,i} \\ +O\left(h^3\right) + O\left(h^p\right).\end{aligned} \tag{3.14}$$

The terms in the RHS are, in order, second-order, third-order, and p-order. If p is equal to 1, then the dissipation terms are first order. If $p \geq 2$, their limiting influence is removed. Expanding the source term in equation (3.8) through a Taylor series expansion, we find that

$$\begin{aligned}S^h_i = S_i + \frac{1}{3}\left(\Delta x_{i+1/2} - \Delta x_{i-1/2}\right) S_{x,i} \\ +\frac{1}{12\Delta x_i}\left(\Delta x^3_{i+1/2} + \Delta x^3_{i-1/2}\right) S_{2x,i} \\ +O\left(h^3\right),\end{aligned} \tag{3.15}$$

which appear to be first, second, and third-order terms, respectively. Substituting our expanded derivatives (equations (3.11), (3.14), and (3.15)) into the standard hyperbolic equation of (3.5), we can obtain the total truncation error as:

$$\begin{aligned}\varepsilon = \left(\Delta x_{i+1/2} - \Delta x_{i-1/2}\right)\left(\frac{1}{2}F_{2x,i} - \frac{1}{3}S_{x,i}\right) \\ -\frac{1}{48\Delta x_i}\left(\Delta x^3_{i+1/2}\left|A_{i+1/2}\right| - \Delta x^3_{i-1/2}\left|A_{i-1/2}\right|\right) q_{3x,i} \\ +O\left(h^3\right) + O\left(h^p\right).\end{aligned} \tag{3.16}$$

Note that in order to reach this result we used the fact that $F_{3x,i} = S_{2x,i}$ exactly at steady-state. This causes the $O\left(h^2\right)$ term to cancel.

From equation (3.16) we can see that the truncation error comes from two sources. The first, contained in the flux-source term on the RHS, is $O\left(h^1\right)$, and comes from the averaging approximation of the midpoint flux of equation (3.9a). The second source comes from the gradient approximation, which is of order p. For the traditional Galerkin method, this is $O\left(h^1\right)$.

3.2.2 Flux-Corrected Galerkin

In the previous section it was discovered that the source of the truncation error in the traditional Galerkin method originates from two main sources: The reconstructed states q^R, q^L, and the central difference approximation of the flux derivative.

Starting with equation (3.7), we again tackle the problem of determining the flux at the halfway locations between nodes. Instead of using the traditional definition given in equation (3.9a), we use the following definition:

$$F^h_{i+1/2} = \frac{1}{2}\left(F^L_{i+1/2} + F^R_{i+1/2}\right) - D^h_{i+1/2}, \tag{3.17}$$

where $D^h_{i+1/2}$ is the same as equation (3.9b). Notice here that we are now reconstructing the *flux* to the midway point, and is done in a similar fashion as the solution variable q:

$$F^L_{i+1/2} = f^h_i + \frac{1}{2}\Delta x_{i+1/2} f^h_{x,i}, \qquad F^R_{i+1/2} = f^h_{i+1} - \frac{1}{2}\Delta x_{i+1/2} f^h_{x,i+1}. \tag{3.18}$$

The truncation error of equation (3.18) is dependent on the order of the gradient approximation, and not on the actual approximation.

Since the fluxes are now the same form as the conserved variables, the only step is to determine an appropriate method of estimating the gradient at the $x_{i+1/2}$ and $x_{i-1/2}$. This needs to be done for the solution values and for the fluxes, separately. In 1D a simple and compact second-order method can be derived from Taylor series as:

$$q^h_{x,i} = \frac{\Delta x^2_{i-1/2} q^h_{i+1} - \Delta x^2_{i+1/2} q^h_{i-1} + \left(\Delta x^2_{i+1/2} - \Delta x^2_{i-1/2}\right) q^h_i}{\Delta x_{i+1/2}\Delta x_{i-1/2}\left(\Delta x_{i+1/2} + \Delta x_{i-1/2}\right)}. \tag{3.19}$$

The major advantage to the choice of flux definition in equation (3.17) is that it can be rewritten in terms of the traditional flux definition from equation (3.9a), and defined as a correction to the traditional flux.

$$F^h_{i+1/2} = \frac{1}{2}\left(f^h_i + f^h_{i+1}\right) - D^h_{i+1/2} - \frac{1}{4}\Delta x_{i+1/2}\left(f^h_{x,i+1} - f^h_{x,i}\right), \tag{3.20a}$$

$$F^h_{i+1/2} = F^h_{i+1/2,linear} - C^h_{i+1/2}. \tag{3.20b}$$

This "corrected" flux can be used in equation (3.7). The improved gradient $q^h_{x,i}$ must be used in $F^h_{i+1/2,linear}$.

A source term can also be derived following this same methodology. With the previous changes, the truncation error of the flux terms becomes:

$$\begin{aligned}&\frac{1}{\Delta x_i}\left[\frac{1}{2}\left(F^L_{i+1/2} + F^R_{i+1/2}\right) - \frac{1}{2}\left(F^L_{i-1/2} + F^R_{i-1/2}\right)\right] \\ &= F_{x,i} - \frac{1}{24\Delta x_i}\left(\Delta x^3_{i+1/2} + \Delta x^3_{i-1/2}\right) F_{3x,i} + O\left(h^3\right) + O\left(h^p\right).\end{aligned} \tag{3.21}$$

The order of this approximation is (starting with the second term on the RHS) 2,3,p. Using the second order gradient of the flux and solution variables, the flux terms are brought up to an order of 2.

3.3 2D Flux Correction

The two dimensional formulation can be determined in a similar fashion as the 1D case. The hyperbolic equation in 2D is given as:

$$q_t + \nabla \cdot \mathbf{F} = 0. \tag{3.22}$$

A triangulation around node 0 is shown in figure 3.2. We now define the divergence of a vector-valued function θ at node 0 as:

$$\nabla^h \cdot \theta^h = \frac{1}{V_0}\sum_i \frac{1}{2}\left(\theta^h_i + \theta^h_0\right) \cdot \mathbf{n}_{0i} S_{0i} = \frac{1}{V_0}\sum_i \frac{1}{2}\left(\theta^h_i - \theta^h_0\right), \tag{3.23}$$

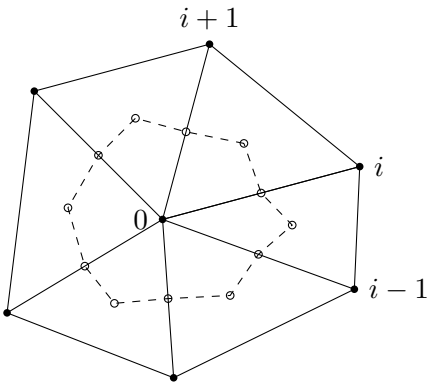


Fig. 3.2: Node-centered stencil on a two-dimensional unstructured triangular grid

where $\mathbf{n}_{0i}$ is the area-weighted normal of the median-dual face located between nodes 0 and i. These faces are shown as the dashed lines. They connect the centroids of the triangular elements with the midpoint of the lines bounding the triangle. These can be approximated by connecting the centroids of the triangular elements instead.

Applying the vector divergence from equation (3.23) to (3.22) results in a flux term similar in form to the 1D case:

$$\nabla^h \cdot \mathbf{F}^h = \frac{1}{V_0} \sum_i F^h_{0i} S_{0i}, \tag{3.24}$$

where F^h_{0i} is a numerical flux between nodes 0 and i. This is our starting point for both the traditional and flux correction methods.

The traditional method approximates F^h_{0i} as:

$$F^h_{0i} = \frac{1}{2}\left(f^h_0 + f^h_i\right) - D^h_{0i} \tag{3.25a}$$

$$D^h_{0i} = \frac{1}{2}\left|A^h_{0i}\right|\left(q^R_{0i} - q^L_{0i}\right), \tag{3.25b}$$

where $A = \frac{\partial f}{\partial q}$ is the directed flux Jacobian, q^R_{0i} and q^L_{0i} are the solution variables reconstructed to the midpoint of the edges connecting nodes 0 and i as follows:

$$q^L_{0i} = q^h_0 + \frac{1}{2}\Delta\mathbf{r}^T_{0i}\nabla^h q^h_0, \quad q^R_{0i} = q^h_i - \frac{1}{2}\Delta\mathbf{r}^T_{0i}\nabla^h q^h_i. \tag{3.26}$$

There are many methods to approximate the gradient of q, and it can be shown that the truncation error of the complete method depends on the accuracy of the gradient approximation. The truncation error also depends on the form of the approximation for F^h_{0i}. Flux Correction uses a higher order gradient approximation and changes the definition of the flux to

$$F^h_{0i} = \frac{1}{2}\left(F^L_{0i} + F^R_{0i}\right) - D^h_{0i}, \tag{3.27}$$

which uses a reconstructed flux instead of an average flux:

$$F^L_{0i} = f^h_0 + \frac{1}{2}\Delta\mathbf{r}^T_{0i}\nabla^h f^h_0, \quad F^R_{0i} = f^h_i - \frac{1}{2}\Delta\mathbf{r}^T_{0i}\nabla^h f^h_i. \tag{3.28}$$

This method can be cast into a "correction" of the linear flux, similiar to the 1D case, and used in equation (3.24). The correction is given as:

$$\begin{aligned} F^h_{i+1/2} = \frac{1}{2}\left(f^h_i + f^h_{i+1}\right) - D^h_{i+1/2} - \frac{1}{4}\Delta x_{i+1/2}\left(f^h_{x,i+1} - f^h_{x,i}\right) \\ F^h_{i+1/2} = F_{i+1/2,linear} - C^h_{i+1/2}. \end{aligned} \tag{3.29}$$

3.4 2D Gradient Approximation

The changes that FC makes to the linear Galerkin method now involves a high-order computation of the gradient of the solution variable q and a high-order computation of the gradient of the *flux* f. Katz and Sankaran employed a quadratic least-squares methodology in their original paper [23]. However, least-squares mathods have been shown to be sensitive to high aspect ratios and curvature of a mesh. This indicates that they will give erroneous gradients in a practical viscous mesh needed to resolve boundary layers.

Katz and Pincock then developed a new gradient method using element mapping methods used in Finite Element and Spectral Volume methods [24]. By estimating gradients in this manner, gradient stencils can be kept compact, promoting stability and solution speed. Unfortunately, it brings with it the need for high-order elements on the boundary. Despite this, this is the gradient methodology that will be used in the work. It should be noted that Flux Correction only specifies the need for high-order gradients, not how they are to

be found.

The two previous works by Katz have only dealt with straight boundaries and elements. This work will investigate the effects of curved elements on the Flux Correction method.

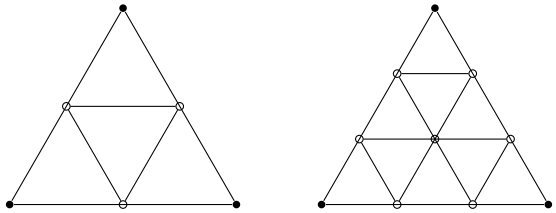
Element mappings for the gradient reconstruction are formed by subdividing each triangle in the mesh into "sub-triangles." These sub-triangles are formed by placing nodes at equally spaced intervals inside the parent triangle, and are illustrated in figure 3.3. From here, a lagrange polynomial is fitted over the parent element, using the extra nodes in the parent element as the interpolation points. The refined mesh with the extra nodes is used in the solution, while the coarse parent mesh is used to provide gradients at the nodes through the lagrange polynomials, as follows:

$$\left.\frac{\partial q}{\partial x}^h\right|_i = \sum_j q_j^h \left.\frac{\partial l_j}{\partial x}\right|_i, \qquad \left.\frac{\partial q}{\partial y}^h\right|_i = \sum_j q_j^h \left.\frac{\partial l_j}{\partial y}\right|_i, \tag{3.30}$$

where l_j is the lagrange polynomial associated with node j in the parent element, and $\left.\frac{\partial q}{\partial x}^h\right|_i$ is the gradients computed at node i in the element. If the lagrange polynomials are given in a standard reference element, then mappings can be used to find the x and y derivatives as:

$$\left.\frac{\partial l_j}{\partial x}\right|_i = \frac{1}{J_i}\left(\frac{\partial l_j}{\partial r}\frac{\partial y}{\partial s} - \frac{\partial l_j}{\partial s}\frac{\partial y}{\partial r}\right)_i, \quad \left.\frac{\partial l_j}{\partial y}\right|_i = \frac{1}{J_i}\left(-\frac{\partial l_j}{\partial r}\frac{\partial x}{\partial s} + \frac{\partial l_j}{\partial s}\frac{\partial x}{\partial r}\right). \tag{3.31}$$

Since the Galerkin method is a continuous method, neighboring triangles will have multiple estimates for the gradients at edge and corner nodes. To make the method consistent, the



(a) Quadratic element (b) Cubic element

Fig. 3.3: Parent triangles ($\cdot$) and subtriangles ($\circ$)

multiple values at nodes are "Jacobian-Averaged":

$$\left.\frac{\partial q^h}{\partial x}\right|_i = \frac{\sum_{k\in i} J_k q^h_{x,k}}{\sum_{k\in i} J_k}, \quad \left.\frac{\partial q^h}{\partial y}\right|_i = \frac{\sum_{k\in i} J_k q^h_{y,k}}{\sum_{k\in i} J_k}, \tag{3.32}$$

where k is the various approximations to the gradient at node i. For linear elements, this reduces to the Green-Gauss procedure.

3.5 Viscous Terms Consideration

Quadratic gradients lead the inviscid terms to be globally third-order, but viscous terms stay second-order. Pincock [24] discovered if cubic gradients are used, then the viscous terms jump to fourth-order. The cubic gradients can cause the inviscid terms to become unstable on the boundary, however. This was resolved by using quadratic gradients on the boundary nodes for the inviscid terms, while using cubic gradients for the inviscid terms on the interior nodes and for the viscous terms throughout the domain. To form a quadratic gradient on a cubic triangle, overlapping quadratic triangles are extracted from the cubic triangle. These are shown in figure 3.4.

Viscous terms require special treatment for them to be stable and accurate. Positivity and stencil compactness have been shown to be necessary in any viscous discretization [33]. Pincock investigated the stability of the viscous terms in the method and found that stability could be achieved in using the same methodology as the inviscid terms, *without any Jacobian averaging.*

With no Jacobian-averaging, stencil compactness is preserved. Similar to the inviscid flux, the viscous flux is:

$$F^{v,h}_{0i} = \frac{1}{2}\left(F^{v,L}_{0i} + F^{v,R}_{0i}\right), \tag{3.33}$$

with left and right corrected fluxes,

$$F^{v,L}_{0i} = f^{v,h}_0 + \frac{1}{2}\Delta\mathbf{r}^T_{0i}\nabla^h f^{v,h}_0, F^{v,R}_{0i} = f^{v,h}_0 - \frac{1}{2}\Delta\mathbf{r}^T_{0i}\nabla^h f^{v,h}_i. \tag{3.34}$$

By using the corrected fluxes across median-dual interfaces, the viscous terms are treated

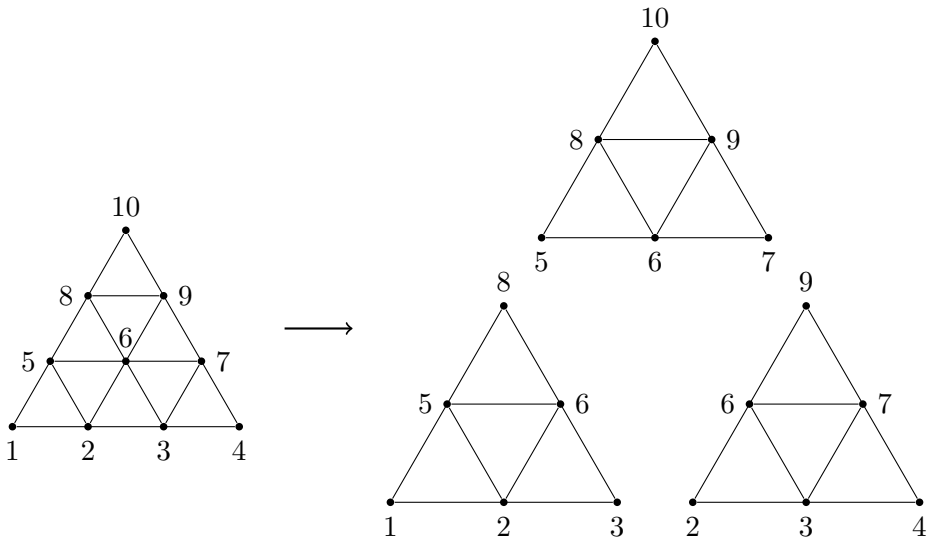


Fig. 3.4: Decomposition of a cubic element into three quadratic elements

in a similiar manner to the inviscid terms, which is necessary to retain the accuracy of the method.

3.6 Source Terms

When present, source terms must also be discretized in a correct manner. This can include MMS terms, unsteady time terms, or turbulent production/destruction terms. The linear Galerkin discretization for source terms in given as:

$$S_0^h = \sum_i \frac{1}{2} \left(S_0 + S_i \right) V_{0i} \tag{3.35a}$$

$$V_{0i} = \frac{1}{4} \Delta \mathbf{r}_{0i} \cdot \mathbf{A}_{0i}, \tag{3.35b}$$

where $\Delta \mathbf{r}_{0i}$ is the position vector from node 0 to node i, and $\mathbf{A}_{0i}$ is the median dual face area vector associated with the edge connecting node 0 and node i. This discretization can be shown to be second-order for irregular grids and third-order for regular grids. In order to be consistent with the FC method, equation (3.35) is replaced with a FC approximation,

$$S_0^h = \sum_i \frac{1}{2}\left(S^L + S^R\right)_{0i} V_{0i}, \tag{3.36a}$$

$$S_{0i}^L = S_0 - \frac{1}{2}\Delta\mathbf{r}_{0i}^T \nabla^h S_0 - \frac{1}{8}\Delta\mathbf{r}_{0i}^T \Delta^{h2} S_0 \Delta\mathbf{r}_{0i}, \tag{3.36b}$$

$$S_{0i}^R = S_i - \frac{1}{2}\Delta\mathbf{r}_{0i}^T \nabla^h S_i - \frac{1}{8}\Delta\mathbf{r}_{0i}^T \Delta^{h2} S_i \Delta\mathbf{r}_{0i}. \tag{3.36c}$$

The discrete source term gradients must be computed as:

$$\nabla^h S = \nabla S + O\left(h^q\right) \quad \nabla^{h2} S = \nabla^2 S + O\left(h^{q-1}\right). \tag{3.37}$$

This discretization leads to solution error of $O\left(h^3\right)$ on regular and irregular grids. For the second derivative, the derivative is taken of the derivative local in in each element. These are the Jacobian-averaged to reconcile the multiple approximations using equation (3.32).

3.7 Unsteady Time Terms

Unsteady time terms are treated using a k-step backward difference formula (BDF), which in general assumes the following form:

$$\frac{\partial q^h}{\partial t} = \frac{1}{\Delta t}\left(\gamma_1 q^{n+1} + \sum_{i=0}^{1-k} \gamma_i q^{n+i}\right), \tag{3.38}$$

where γ_i depend on the order of the time derivative, and Δt is the time step. The iteration in physical time is defined as n.

Pincock studied two BDF formulations, BDF2 and BDF3. BDF2 is a second-order method and BDF3 is third-order. While third-order temporal accuracy would be more desirable, it quickly became unstable, and thus will not be used for this work. The coefficients for BDF2 are: $\gamma_1 = 1/2, \gamma_0 = -2, \gamma_{-1} = -1/2$.

3.8 Solution Method

Because FC is based on finite volume methodology, a plethora of mature solution

techniques already exist. For this work, we will use an explicit Runge-Kutta psuedo-time approach with implicit residual smoothing, adaptive pseudo-time stepping, and multigridding. These methods work well for low-Reynolds number cases and in the absence of extreme mesh anistrophy. To actually solve these equations, A mass-lumped pseudo-time derivative may be added to the discrete equations:

$$V\frac{\partial q}{\partial \tau} + R(q) = 0, \tag{3.39}$$

where τ is the pseudo-time variable. $R\left(q^h\right)$ is the unsteady discrete residual

$$R\left(q\right) = \sum_i F_{0i}^h - \sum_i F_{0i}^{v,h} - S^h(q)\,. \tag{3.40}$$

This pseudo-time equation is treated with an explicit n_s-stage Runge-Kutta scheme of Jameson [34], which will hereafter be referred to as J-RK. This splits the pseudo-time residual into stage updates.

$$\begin{gathered} q^{k+1,0} = q^k, \\ q^{k+1,m} = q^k + \Delta q^{k+1,m}, m = 1, \ldots, n_s, \\ q^{k+1} = q^{k+1,n_s}, \end{gathered} \tag{3.41}$$

where k is the pseudo-time counter, m is the stage counter, and $\Delta q^{k+1,m}$ is the mth stage update. By treating the fluxes explicitly and the physical time source terms implicitly in pseudo-time, the update equation becomes:

$$\begin{gathered} \left(a_\tau + a_t\right)\Delta q^{k+1,m} - a_t \Delta q^{k+1,m-1} + R\Big(q^{k+1,m-1}\Big) = 0, \\ a_\tau = \frac{V}{\alpha_m \Delta\tau}, \quad a_t = \frac{V_{\gamma_1}}{\Delta t}, \end{gathered} \tag{3.42}$$

where α_m is the RK coefficient for stage m. Here, the LHS has been mass-lumped for convenience in computing the update in pseudo-time. The RHS retains the consistent source discretization which is satisfied at steady-state. Equation (3.42) is used to determine

the stage update for equation (3.41). Before updates are applied, they are smoothed with an implicit residual smoothing operation [35].

3.9 Multigrid

The mesh refining procedure shown in figures 3.3 and 3.4 gives a convenient agglomeration to be used in a multigrid solver. The multigrid used in this work is the standard Full Approximation Storage (FAS) algorithm of Brandt [36]. Starting from cubic elements, the mesh is coarsened to quadratic elements, then from there to linear elements. Restriction and prolongation operations are performed by interpolating solutions, residuals, and corrections using the available lagrange interpolating polynomials over each element. Using these available interpolations allows for more accurate transfers then conventional averaging or injection procedures. Multigrid forcing terms are added on coarse levels in the standard fashion. This methodology was observed to provide good convergence acceleration for all test cases.

3.10 Boundary Conditions

Boundary conditions for this method are implemented using a "selection matrix," which operates on the discretized equations of motion to specify the discrete residual for boundary nodes. This approach was originally proposed by Allmaras [37], and the method shown here is essentially a generalization of the one proposed by Allmaras and used by Folkner [38].

To incorporate the boundary conditions, we multiply equation (3.42) by a selection matrix L, which selects combinations of the interior equations. This is then augmented with the additional conditions, denoted here as R_b.

$$L\left[\left(a_\tau + a_t\right)\Delta q^{k+1,m} - a_t \Delta q^{k+1,m-1} + R\Big(q^{k+1,m-1}\Big)\right] + R_b\Big(q_b^{k+1,m}, q^{k+1,m-1}\Big) = 0. \quad (3.43)$$

Note that the boundary node is evaluated implicitly at stage m. The additional R_b conditions can be linearized by:

$$R_b\Big(q_b^{k+1,m}, q^{k+1,m-1}\Big) \approx R_b\Big(q_b^{k+1,m-1}, q^{k+1,m-1}\Big) + \frac{\partial R_b}{\partial q_b}\Delta q^{k+1,m} - \frac{\partial R_b}{\partial q_b}\Delta q^{k+1,m-1}. \quad (3.44)$$

Substituting this into (3.43), an expression is obtained which may be solved for the mth stage update at the boundaries:

$$\begin{aligned}\left[L\left(a_\tau + a_t\right) + \frac{\partial R_b}{\partial q_b}\right]\Delta q^{k+1,m} - \left(a_t L + \frac{\partial R_b}{\partial q_b}\right)\Delta q^{k+1,m-1} \\ + LR\Big(q^{k+1,m-1}\Big) + R_b\Big(q_b^{k+1,m-1}, q^{k+1,m-1}\Big) = 0.\end{aligned} \quad (3.45)$$

Note that, to solve this for $\Delta q^{k+1,m}$, the expression in the brackets must be inverted. In two dimensions, this is a 4x4 matrix, and is only required at boundary nodes. Once the updates are computed, these are included in the implicit residual smoothing operation described earlier. In this work, we use inviscid and viscous walls, inflow and outflow conditions, and dirichlet conditions. The selection matrices and R_b for these conditions can be found in Folkner's thesis [38].

Chapter 4

Flux Reconstruction

Flux Reconstruction starts by combining the inviscid and viscous term of equation (2.3) into a single flux term, as:

$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{F}\left(q, \nabla q\right) = 0. \tag{4.1}$$

The inviscid fluxes are a function of q and the viscous fluxes are a function of q and ∇q.

Equation (4.1) can be rewritten as a system of equations:

$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{F}\left(q, \mathbf{b}\right) = 0, \tag{4.2a}$$

$$\mathbf{b} - \nabla q = \mathbf{0}. \tag{4.2b}$$

The solution to this system of equations is contained inside domain Ω which is bounded by boundary Γ. The solution domain Ω is discretized into triangular elements, where the region of a single element k is denoted by Ω_k^D, with the element boundary denoted by Γ_k^D. The solution q inside an element is also discretized into solution q^D and in general is not continuous across element boundaries.

In order to facilitate implementation, the solution is mapped to a reference equilateral triangle in (r, s) with vertices $\left(-1, -\frac{1}{\sqrt{3}}\right)$, $\left(1, -\frac{1}{\sqrt{3}}\right)$, $\left(0, \frac{2}{\sqrt{3}}\right)$. This triangle is shown in figure 4.1.

With this mapping, the solution variables and the fluxes can be transformed to the reference space with the following equations:

$$\widehat{q} = Jq, \quad J = x_r y_s - x_s y_r, \tag{4.3a}$$

$$\widehat{\mathbf{F}} = \left\langle \widehat{f}, \widehat{g} \right\rangle = \left\langle y_s f - x_s g \, , \, -y_r f + x_r g \right\rangle. \tag{4.3b}$$

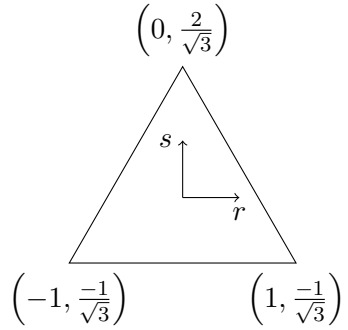


Fig. 4.1: Reference triangle used in this work

Utilizing the discretization and the reference space transformation, equation (4.2) becomes:

$$\widehat{q}_t^D + \widehat{\nabla} \cdot \widehat{\mathbf{F}}^D = \mathbf{0}, \tag{4.4a}$$

$$\widehat{\mathbf{b}}^D - \widehat{\nabla}\widehat{q}^D = \mathbf{0}, \tag{4.4b}$$

where $\widehat{\nabla}$ is the gradient operator in the reference space.

4.1 Procedure

A general overview of Flux Reconstruction (FR) in two dimensions is given here. We follow the basic procedure given in the works of Hyunh, Jameson, and Williams [21, 22, 39, 40].

The approximate solution in reference space is defined with a two-dimensional polynomial of degree p. The polynomial is formed from $N_{sp} = \frac{1}{2}(p+1)(p+2)$ solution points, which are placed within the triangle. This solution polynomial is given as:

$$\widehat{q}^D = \sum_{i=1}^{N_{sp}} \widehat{q}_i^D l_i(\mathbf{r}). \tag{4.5}$$

Each polynomial l_i is defined to be 1 at node i and 0 at all other nodes, in the lagrange fashion. The interpolation field described in equation (4.5) is also used to interpolate derivatives and fluxes.

Equation (4.4b) is solved first. This gives the derivatives of q to be used in the viscous fluxes. To accomplish this, common "flux points" are defined along edges separating two adjacent cells. The number of required flux points along an edge is $N_{fp} = p + 1$. This number is chosen so that the order of the interpolating polynomial along the edge will match the order of the interior 2D polynomial. Adjacent cells are required to have flux points at the same physical location on the edge. In the following description, quantities on flux points will be denoted with the subscript f, j. f represents the face, and j the flux point on that face. For triangles, f ranges from 1 to 3, and j from 1 to N_{fp}.

Using the interior polynomial, the solution is interpolated to the flux points along the edges and then transformed to physical space using equation (4.3a). Each flux point now has two solutions (q^L, q^R) at the flux points. An "interface" value (q^I) is then computed using the left and right states at that flux point. This interface value is used to make the solution continuous in a weak manner across element boundaries. Actually computing the interface value can be done in many fashions, including Central Flux (CF) [41], Local Discontinuous Galerkin (LDG) [15], Compact Discontinuous Galerkin (CDG) [42], Internal Penalty (IP) [43], Bassi Rebay 1 (BR1) [44], or Bassi Rebay 2 (BR2) [45]. The interface value is then transformed back to the reference space for each cell.

The derivatives of $\widehat{q}$ are then split into two parts: a discontinuous derivative that is local to the cell, and a correction that involves the interface values.

$$\widehat{\mathbf{b}}^D = \widehat{\nabla}\widehat{q}^D + \widehat{\nabla}\widehat{q}^C. \tag{4.6}$$

The correction value $\widehat{q}^C$ at the flux points is required to be

$$\widehat{q}^C_{f,j} = \widehat{q}^I_{f,j} - \widehat{q}^D_{f,j}. \tag{4.7}$$

The discontinuous derivative is easily computed from (4.5) by taking the derivative of the lagrange polynomial. The correction derivative is not as straightforward, and the handling

of it is crucial to the method. It is defined as follows:

$$\widehat{\nabla}\widehat{q}^{C}\left(\mathbf{r}\right)=\sum_{f=1}^{3}\sum_{j=1}^{N_{fp}}\widehat{q}_{f,j}^{C}\,\psi_{f,j}(\mathbf{r})\;\widehat{\mathbf{n}}_{f,j}. \tag{4.8}$$

The function $\psi_{f,j}\left(\mathbf{r}\right)$ is a "lifting operator," which moves the discontinuous solution at flux point f,j to the interface value, and propogates that movement into the interior of the element. Putting it all together, the equation to solve for $\mathbf{b}$ at a solution point i becomes:

$$\widehat{\mathbf{b}}_{i}^{D}=\sum_{m=1}^{N_{sp}}\widehat{q}_{m}^{D}\,\widehat{\nabla}l_{m}\left(\mathbf{r}_{i}\right)+\sum_{f=1}^{3}\sum_{j=1}^{N_{fp}}\widehat{q}_{f,j}^{C}\,\psi_{f,j}(\mathbf{r}_{i})\;\widehat{\mathbf{n}}_{f,j}. \tag{4.9}$$

In the next stage, the fluxes are computed in a similiar manner. The interpolation from (4.5) is used to define the discontinuous flux for each component of the flux in the reference space. The solution at the flux points is used to compute the fluxes. This gives two values at each flux point. These two flux values are utilized to form a common interface flux.

The flux, as currently defined, includes both advective and diffusive components. For fluids problems, these are generally known as the inviscid and viscous terms, respectively. Each requires a different method to determine the common interface flux. The inviscid terms can be found using a Riemann Solver, for example, in a manner following Roe [46] or Rusanov [47]. The viscous terms can be computed using any of CF, LDG, CDG, IP, RB1, or RB2 mentioned previously.

Once the interface fluxes for the inviscid and viscous terms have been found, they are added together and then transformed back to the reference space for each neighboring element. At this point, the flux is separated into two separate components, a discontinous and a correction component. The divergence of the flux can then be found as:

$$\widehat{\nabla}\cdot\widehat{\mathbf{F}}^{D}=\widehat{\nabla}\cdot\widehat{\mathbf{F}}^{D}+\widehat{\nabla}\cdot\widehat{\mathbf{F}}^{C}, \tag{4.10}$$

where

$$\widehat{\mathbf{F}}_{f,j}^{C}=\widehat{\mathbf{F}}_{f,j}^{I}-\widehat{\mathbf{F}}_{f,j}^{D}. \tag{4.11}$$

The correction flux is constructed using correction vector functions $\mathbf{h}_{f,j}(\mathbf{r})$. These are two dimensional polynomials of order p. The correction functions are required to satisfy the following condition:

$$\mathbf{h}_{f,j}(\mathbf{r}_{m,n}) \cdot \widehat{\mathbf{n}}_{m,n} = \begin{cases} 1 & \text{if } f = m \text{ and } j = n \\ 0 & \text{if } f \neq m \text{ and } j \neq n \end{cases}. \tag{4.12}$$

The correction flux at solution point i is then determined by:

$$\widehat{\mathbf{F}}^{C}(\mathbf{r}_i) = \sum_{f=1}^{3} \sum_{j=1}^{N_{fp}} \left[\left(\widehat{\mathbf{F}}^{I}_{f,j} - \widehat{\mathbf{F}}^{D}_{f,j} \right) \cdot \widehat{\mathbf{n}}_{f,j} \right] \mathbf{h}_{f,j}(\mathbf{r}_i) = \sum_{f=1}^{3} \sum_{j=1}^{N_{fp}} \Delta_{f,j} \mathbf{h}_{f.j}(\mathbf{r}_i). \tag{4.13}$$

Once this is done, the flux derivatives can be computed at each solution point. The discontinuous term is computed by taking the divergence of the lagrange interpolating polynomial. The correction term can be found by finding the divergence of the vector correction function. To simplify notation, this will be notated by:

$$\phi_{f,j}(\mathbf{r}) = \widehat{\nabla} \cdot \mathbf{h}_{f,j}(\mathbf{r}). \tag{4.14}$$

Thus the completed equation at solution point i becomes:

$$\frac{\partial \widehat{q}_i}{\partial t} = -\sum_{k=1}^{N_{sp}} \widehat{f}^{D}_{k} \frac{\partial l_k}{\partial r}(\mathbf{r}_i) - \sum_{k=1}^{N_{sp}} \widehat{g}^{D}_{k} \frac{\partial l_k}{\partial s}(\mathbf{r}_i) - \sum_{f=1}^{3} \sum_{j=1}^{N_{fp}} \Delta_{f,j} \phi_{f,j}(\widehat{\mathbf{r}}_i) \tag{4.15}$$

This residual equation can then be integrated in time using any number of integration schemes. This aspect will be considered later on. In summary, the nature of a FR scheme depends on:

1. The location of the solution points. (Section 4.2)

2. The location of the flux points. (Section 4.2)

3. The methodology for calculating the interface values $q^{I}_{f,j}$. (Section 4.3)

4. The methodology for calculating the interface flux $\mathbf{F}^I_{f,j}$. (Section 4.3)

5. The form of the solution correction field $\psi_{f,j}$. (Section 4.4)

6. The form of the divergence $\phi_{f,j}$ of the correction functions $\mathbf{h}_{f,j}$. (Section 4.4)

Each of these points will be addressed in the following sections.

4.2 Solution and Flux Point Locations

The location of the solution points is critical in minimizing aliasing errors. Previous analysis of FR in one dimension has shown that the locating the solution and flux points at good integration points is necessary to minimize aliasing errors [48]. Castonguay et al. have shown that a stable choice for flux point locations is the 1D Gauss integration points, and a stable choice for the solution points locations is the numerical quadrature points given by Taylor [49].

4.3 Interface Values

As mentioned previously, there are three interface values that need to be formed. The first is a solution q^I interface value. This is used in the determination of $\mathbf{b}$ or ∇q. Since our implementation only handles inviscid terms, the derivative of the solution values is not necessary, and is not used. The second is the inviscid flux interface value $\mathbf{F}^I_{inv}$. As mentioned previously, this can be computed using any Riemann Solver. In this work, the Approximate Riemann Solver of Roe is used. The third is the viscous flux interface value $\mathbf{F}^I_{vis}$. These can be computed using any of the methods previously mentioned. Again, in this work, the viscous terms are not included.

4.4 Correction Functions

Vincent, Castonguay, and Jameson have developed a form of the correction field titled as Vincent-Castonguay-Jameson (VCJ) Schemes [40]. These have been proven to be stable for linear problems, and have been shown to be stable for non-linear problems as well. For

simplicity, both the solution correction field $\psi_{f,j}$ and the flux correction field $\phi_{f,j}$ are taken as the same, although there is nothing enforcing this.

The correction fields $\phi_{f,j}$ are assumed to take the form

$$\phi_{f,j} = \sum_{k=1}^{N_{sp}} \sigma_k \psi_k(\mathbf{r}), \tag{4.16}$$

where ψ_k is the 2D orthonormal Dubiner basis given by

$$\psi_k(\mathbf{r}) = \frac{2}{3^{0.25}} \mathcal{P}_v^{(0,0)}(a) \mathcal{P}_w^{(2v+1,0)}(b)(1-b)^v, \tag{4.17}$$

where v and w are given implicitly by the following:

$$k = w + v(p+1) - \frac{v}{2}(v-1) + 1, \quad (v,w) \geq 0, \quad v + w \leq p. \tag{4.18}$$

$\mathcal{P}_n^{(\alpha,\beta)}$ is the normalized n-th order Jacobi polynomial, and a and b are given by

$$a = \frac{3r}{2 - \sqrt{3}s}, \quad b = \frac{1}{3}\left(2\sqrt{3}s - 1\right). \tag{4.19}$$

Finally, correction field coefficients σ_k can be found by solving

$$c \sum_{k=1}^{N_{sp}} \sigma_k \sum_{m=1}^{p+1} \binom{p}{m-1} \left(\mathcal{D}^{(m,p)}\psi_i\right)\left(\mathcal{D}^{(m,p)}\psi_k\right) = -\sigma_i + \int_\Gamma \left(\mathbf{h}_{f,j} \cdot \widehat{\mathbf{n}}\right) \psi_i d\Gamma, \text{ for } 1 \leq i \leq N_{sp}. \tag{4.20}$$

This methodology is characterized by the parameter c. Schemes of varying types and properties can be found solely by changing the value of c. For instance, when $c = 0$, the collocation-based DG Method is recovered in 1D [26]. Through several theoretical studies and numerical experiments, values of c that provide better accuracy or timestepping or stability have been found. The value of c is dependent on the time integration method and the order p [21, 39, 40, 48].

4.5 Time Integration and values of c

The time integration of the residual equation in equation (4.15) can be done in any number of ways. Castonguay, Vincent, Jameson, and Williams tested several methods of time integration, while varying the value of c, searching for maximum explicit time-step limits. They discovered that the 4th order, 5-stage, 2N-storage RK scheme of Carpenter [50] yielded the highest maximum time-step limit. This will hereafter be referred to as C-RK. For this integration method, four special values of c have been discovered. These are listed in table 4.1.

c_{dg} is the value of c that has been shown to recover a DG scheme in 1D. This value of c has also been shown to be the most accurate of all values of c. c_{sd} has been shown to recover a SD method in 1D. c_{hu} recovers a scheme shown by Huynh [21] to be particularly stable. c_+ is the value that yields the maximum possible explicit time-step. For this work, we will focus mainly on c_{dg} and c_+ for comparisions to FC.

For steady-state problems, we can also integrate in time using the RK method introduced by Jameson [34]. This method splits the diffusive and convective terms of the equations and treats them separately. This is results in a larger stability envelope, but ruins the temporal accuracy. Thus, this method is only suitable for steady-state problems, where time accuracy is not an issue. This method will be known as J-RK, and is explained in more detail in section 3.8.

A p-multigrid was implemented, but was not found to help very much. This mirrors the results found by Nastase [51], in which p-multigrid was found to be not very effective without h-multigridding.

Table 4.1: The four special values of c for C-RK with $p = 2$

c_{dg}	0
c_{sd}	$6.18e-3$
c_{hu}	$1.39e-2$
c_+	$4.30e-2$

Chapter 5

Numerical Results

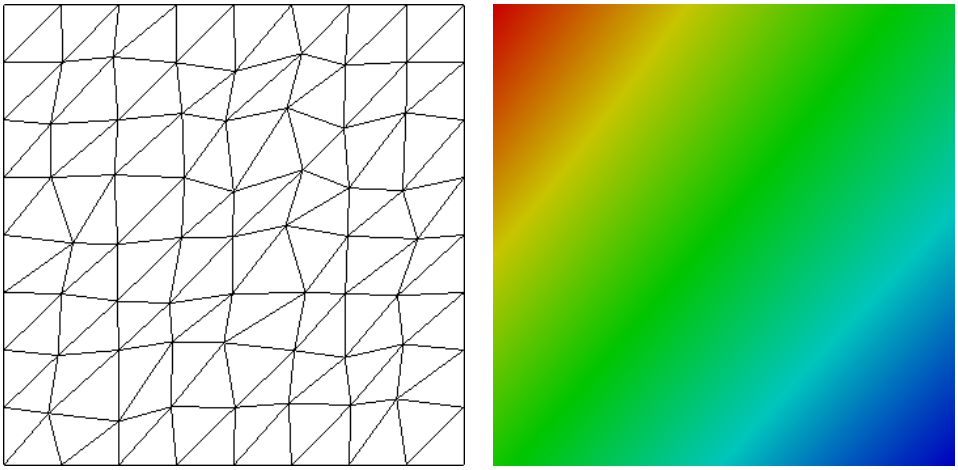
In order to compare Flux Correction with Flux Reconstruction, a code was written for each method. The FR implementation does not include viscous terms and will not attempt to include these into the comparison. Here we present results for two comparision tests: The first is a simple accuracy study using MMS, the second is a traveling isentropic vortex study to evaluate the numerical dissipation of FC and FR. Cost results for the MMS test are also presented. The effect of curved elements on Flux Correction is then studied, using an unsteady, vicous, circular cylinder case.

5.1 Method of Manufactured Solutions

The Method of Manufactured Solutions is used to ensure that the implementations are free from coding mistakes and that the expected order of accuracy is recovered. This is covered in more detail in section 2.3.

This solution is used with grid refinement studies to determine the order of accuracy of the methods. The governing equations were solved over a unit square domain. The meshes used are characterized by the number of boundary edges along one side of the square. Meshes of 4, 8, 16, 32, and 64 triangles on a side were used. The nodes of the mesh were then perturbed randomly to introduce non-uniformity into the mesh and prevent possible supraconvergence of the solution error. Figure 5.1(a) demonstrates a mesh with $N = 8$ triangles on a side.

This case tests only the inviscid terms, but the implementation of FC allowed for using either quadratic or cubic approximations for the gradients. Intuitively, the the cubic version is expected to be more accurate than the quadratic, and this is clearly seen in figure 5.2(a). There is no difference between quadratic or cubic gradient accuracy for first or second order



(a) Sample Grid (b) MMS Density Solution

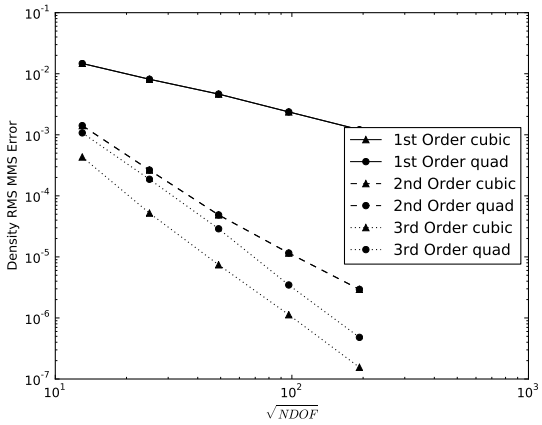
Fig. 5.1: Setup for the Method of Manufactured Solutions

because the implementation reverted to the standard first-order Green-Gauss derivative procedure in these cases. Flux Reconstruction has an infinite number of possible solution options, characterized by the parameter c. Jameson [48] showed that the properties of a particular method vary with c, and that the method is most accurate when $c = 0$. For this particular study, we use $c = 0$. We integrate in time with both the J-RK and C-RK schemes. Figure 5.2(b) shows the MMS results for both of these. As expected, the choice of time-integration scheme does not affect the spatial accuracy.

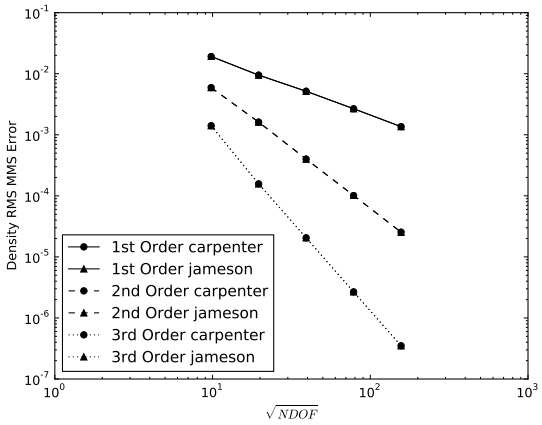
Figure 5.2(c) shows the comparision between FR and FC. For the first-order method, both methods are comparable in accuracy. For the second and third-order methods, FC has an obvious advantage over FR. Comparing figure 5.2(c) with 5.2(a), we see that in the third-order case, FC loses this advantage should quadratic gradients be used. In a practical application though, viscous terms would be included, requiring cubic gradients to maintain the third-order accuracy.

5.2 Cost Studies

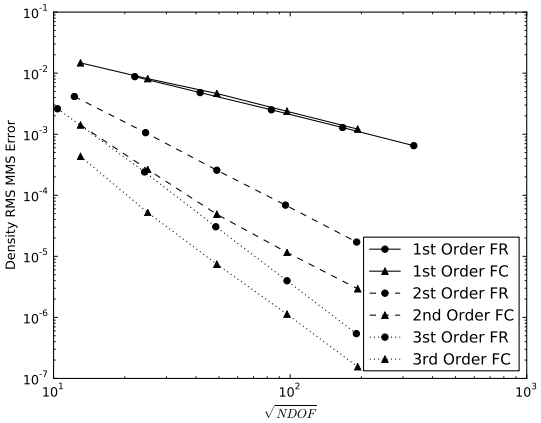
The practicing engineer has several decisions and compromises to make concerning cost, available time and resources, and the required level of accuracy. This section provides a comparision of FR and FC on the time required to obtain a given level of accuracy.



(a) Density errors for various FC orders



(b) Density errors for various FR time integration schemes



(c) Density errors comparing FC and FR

Fig. 5.2: MMS results

Benchmarking a numerical scheme depends on several factors, including, but not limited to: the skill of the programmer, method of implementing the scheme, choice of compiler and various optimizations, and hardware used. Finding the limiting factor among so many variables is an incredibly difficult, making benchmarks questionable at best. In general, it can be agreed that a scheme that delivers the same accuracy in a shorter time will be better than another that takes longer.

The FR and FC implementations were written by the same team of programmers in a manner similar to each other. Basic optimization rules were adhered to, (memory layout,etc.) but no special effort was made to discover optimal implementations. All tests were run in serial.

Here we will present detailed cost results for the Order of Accuracy study from section 5.1. Due to the conclusions drawn from this study, such detail will not be presented for the other conducted experiments.

The results presented here used the same setup from section 5.1. In order to uncover a hardware preference of the implementations, the tests were run on two different setups. The first setup was an AMD Phenom II X4 955 3.2 Ghz CPU, with 1333Mhz DDR3 memory. Compilation was done using version 4.8.1 of the GNU compiler suite. The second architecture was an Intel Core i3-2330M 2.2Ghz CPU, with 1333Mhz DDR3 memory. Compilation was done using version 4.7.3 of the GNU compiler suite. For both setups, the O3 optimization level was used with full use of available SSE registers.

Four categories of plots are shown in figures 5.3, 5.4, and 5.5. Three of the categories are plotted against the true solution error in density instead of NDOF or a characteristic length scale. This can be interpreted as the amount of effort required to achieve a given level of solution error. These numbers should not be interpreted as hard values, but indicate trends of the methods as more accuracy is required.

The first plot category is a density RMS convergence plot. This shows the RMS level of the density variable as a function of iteration, which should go to zero for a steady-state case. Iteration convergence is not dependant on the hardware, but should only be a function of

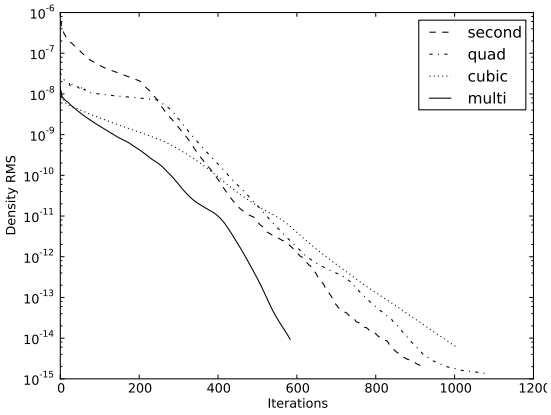
the numerical scheme and the particular problem to be solved. The second category shows the number of iterations versus level of accuracy. The third category shows the elapsed walltime versus level of accuracy. The results for both the AMD and Intel setups are shown for the second and third categories. The fourth category shows the time per iteration of the method vs. level of accuracy. These plots also show the standard deviation to give a measure of the effect of "Operating System Jitter" on the walltime.

Figure 5.3 compares various implementations of Flux Correction. These show the difference in the traditional second-order linear Galerkin method (second), Flux Correction with quadratic gradients (quad), Flux Correction with cubic gradients (cubic), and Flux Correction with full p multi-grid, implicit residual smoothing, and relaxation. For all cases, a maximal CFL was determined and used.

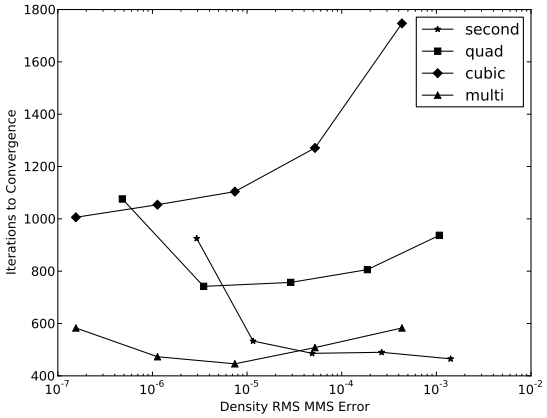
Figure 5.4 compares the J-RK (jameson) and C-RK (carpenter) time-integration methods for the third-order Flux Reconstruction. For both cases, $c = 0.043$, and a maximal CFL was determined and used.

Figure 5.5 is a restatement of the J-RK FR results, the cubic FC and the multi-grid cubic FC results. This allows for a closer consideration of the two methods.

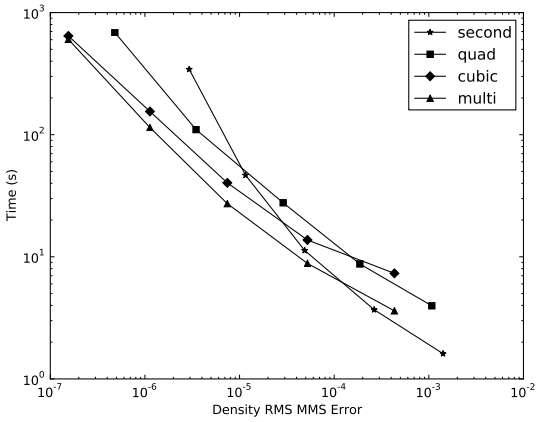
The various FC tests easily highlight the superiority of FC over the traditional Galerkin method. From the walltime plots in figures 5.3(c) and 5.3(d), it is obvious that as the error goes down, the second-order method takes longer than the cubic methods to achieve the same level of accuracy. This exemplifies the fact that at some point, high-order methods become more cost effective than low-order methods. Another most interesting aspect of Flux Correction is shown in the number of iterations to convergence (figure 5.3(b)). The second-order and quadratic cases exhibit the expected increase in iterations on the finest mesh, while the cubic case shows a *decrease* in needed iterations with decreasing error. It is hypothesized that this is because the finer meshes result in smoother cubic polynomials. Coarser meshes might be exhibiting nonexistent solution oscillations, thus taking longer to converge to the correct solution. The multigrid case keeps the number of iterations fairly level, as is expected. While multi-grid did not significantly decrease the time required, it



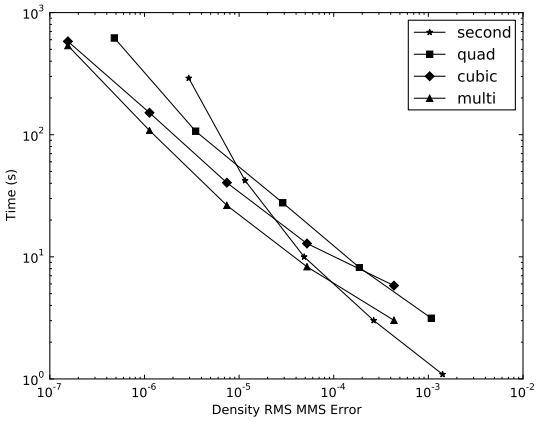
(a) FC convergence history on fine grid



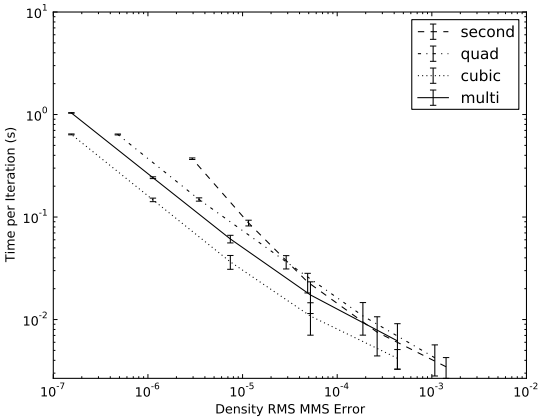
(b) FC iterations to RMS convergence for given true accuracy



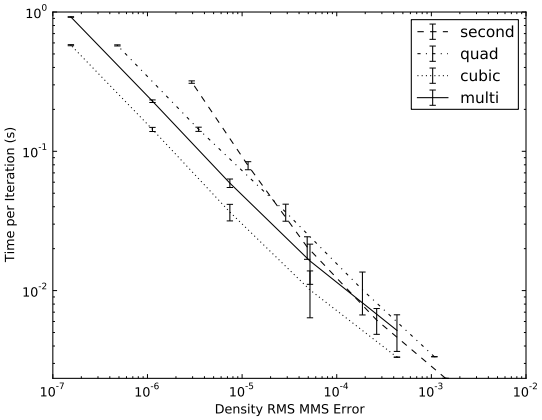
(c) FC walltime to given accuracy on AMD Phenom II



(d) FC time for given accuracy on Intel i3

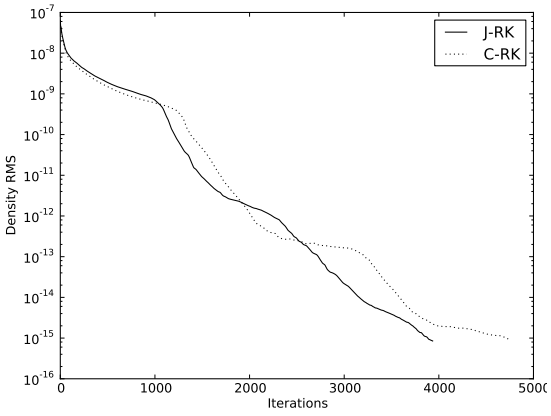


(e) FC time per iteration for given accuracy on AMD Phemon II

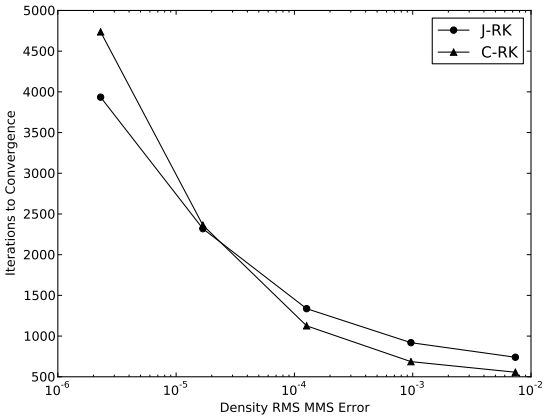


(f) FC time per iteration for given accuracy on Intel i3

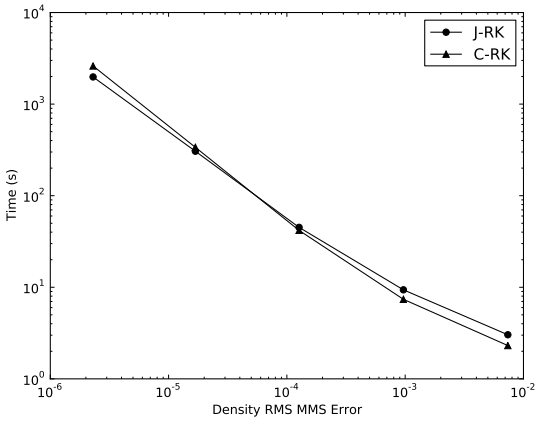
Fig. 5.3: Results from Flux Correction Cost Studies.



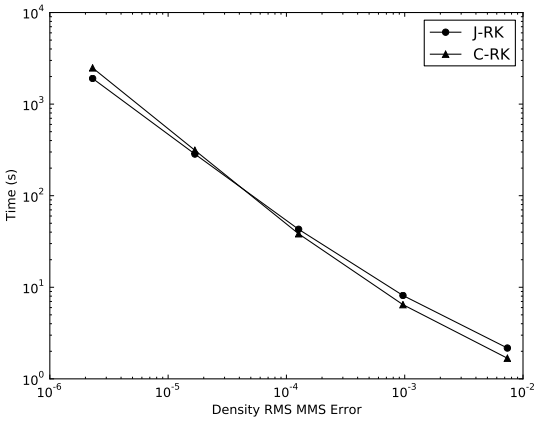
(a) FR convergence history on fine grid



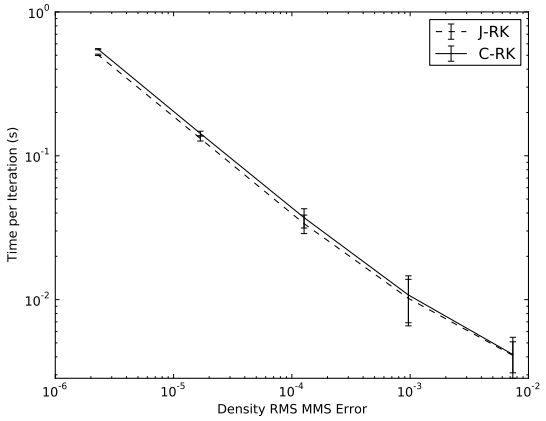
(b) FR iterations to RMS convergence for given true accuracy



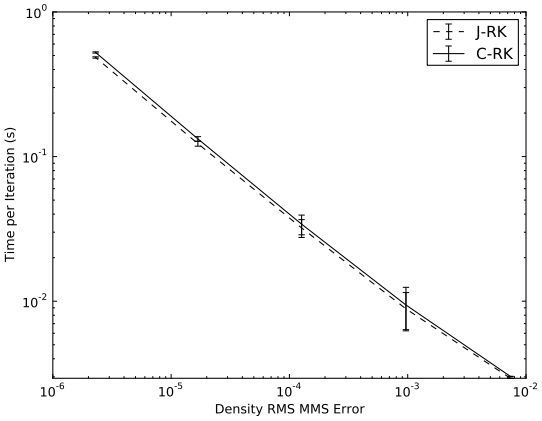
(c) FR walltime to given accuracy on AMD Phenom II



(d) FR time for given accuracy on Intel i3

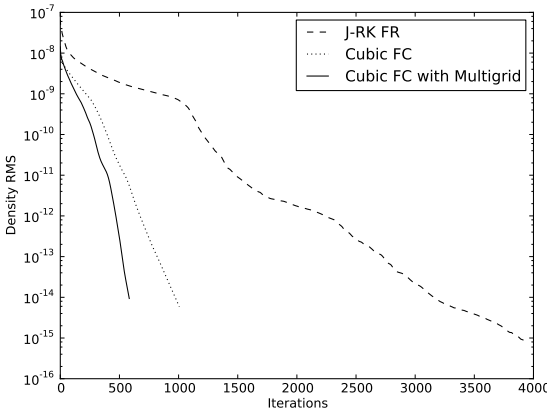


(e) FR time per iteration for given accuracy on AMD Phemon II

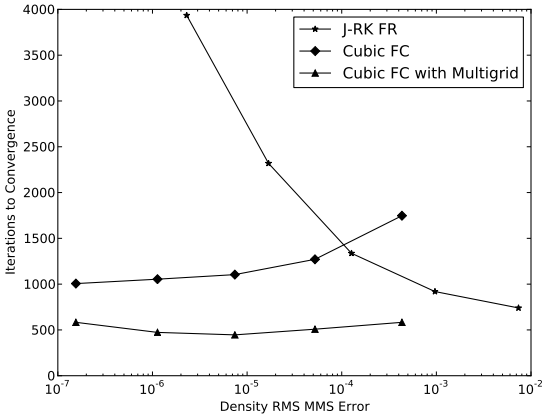


(f) FR time per iteration for given accuracy on Intel i3

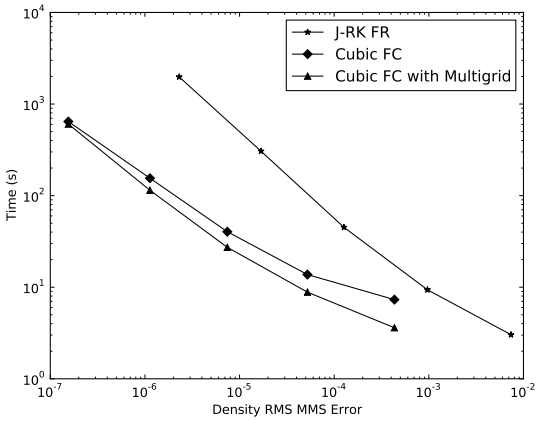
Fig. 5.4: Results from Flux Reconstruction Cost Studies. These plots show two possible cases. The first is integrating in time using J-RK, the second using C-RK.



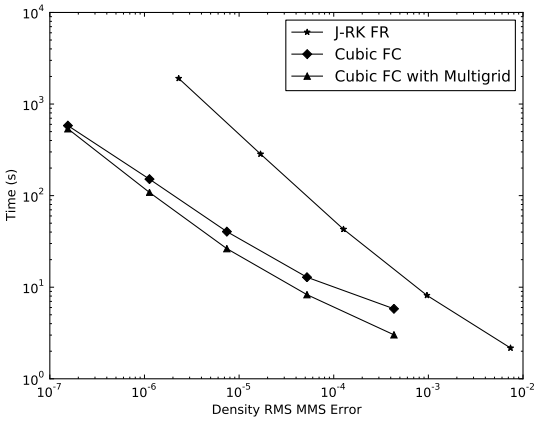
(a) FR vs FC convergence history on fine grid



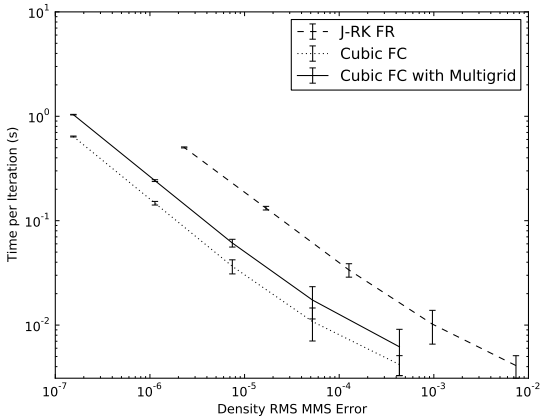
(b) FR vs FC iterations to RMS convergence for given true accuracy



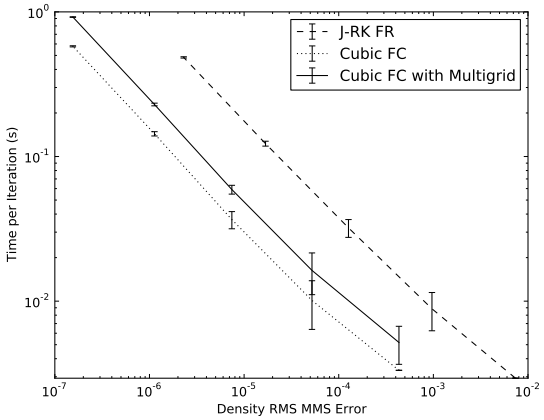
(c) FR vs FC walltime to given accuracy on AMD Phenom II



(d) FR vs FC time for given accuracy on Intel i3



(e) FR vs FC time per iteration for given accuracy on AMD Phemon II



(f) FR vs FC time per iteration for given accuracy on Intel i3

Fig. 5.5: Cost Comparisons. These plots show the comparision of the J-RK FR timings versus the third-order cubic gradient FC timings and the third-order cubic gradient full multi-grid with implicit residual smoothing and relaxation factors FC.

did significantly decrease the required iterations. This is expected to have a larger impact in more difficult, boundary-driven flows.

The FR tests show a rather surprising indifference to the time-integration method used. The main difference between the J-RK and C-RK shows in the iterations necessary for a certain level of error. The allowable CFL was also significantly, with J-RK having a maximum CFL of 9, and C-RK having a maximum CFL of 23. The time required, however, was almost the same. The difference could become more pronounced in a more difficult steady-state case, for which J-RK was designed.

The FR and FC codes both showed a preference to the Intel Core i3 setup. This could be due to any number of reasons, including more available registers, better compiler optimizations or faster IO speeds. The difference in timing however, ended up being slight. The important aspect of this comparision is that the trends of the results did not change across hardware setups. The error bars on the time iteration plots show that OS Jitter was significant on the less refined meshes. But as the NDOFs increased, it had a smaller impact on the time until it became almost negligible. Thus the timing with the smallest error is also the most reliable time on the diagram.

The superiority in the aspect of time for a steady-state solution in a serial computation becomes apparent when comparing FR and FC directly. The sharp increase in iterations of FR versus the flat trends of FC in figure 5.5(b) are especially telling. Figure 5.5(d) shows an almost order of magnitude difference in the time required. It must be remembered, however, that both implementations are similar, and the implementation might lean more towards a Continuous Finite Volume Method pattern. This would leave the FR implementation in a less-than-optimal situation. Moreover, due to the discontinuous nature of FR, it is relatively easy to parallelize. Williams [27] presented a 3D fully-compressible viscous arbitrarily high-order FR solver parallelized over multiple CPUs and GPUs with a weak scalability of 90%. The ease of this parallelization should not be overlooked. The major point that can be found from the comparision is the number of iterations required to converge to a steady-state. The flat profile of the FC is very attractive.

5.3 Isentropic Vortex

For inviscid flow, an isentropic vortex should mathematically last forever, never dissipating into the surrounding medium. By simulating such a vortex, a measure of the numerical dissipation inherent in a method can be obtained. The isentropic vortex case considered is the one described by Shu [52]. The case consists of a uniform flow, onto which an isentropic vortex is added.

$$\rho = u = P = 1 \tag{5.1a}$$

$$v = 0 \tag{5.1b}$$

$$\Delta u = -y\frac{\epsilon}{2\pi}\exp\left(\frac{1}{2}\left(1-R^2\right)\right) \tag{5.1c}$$

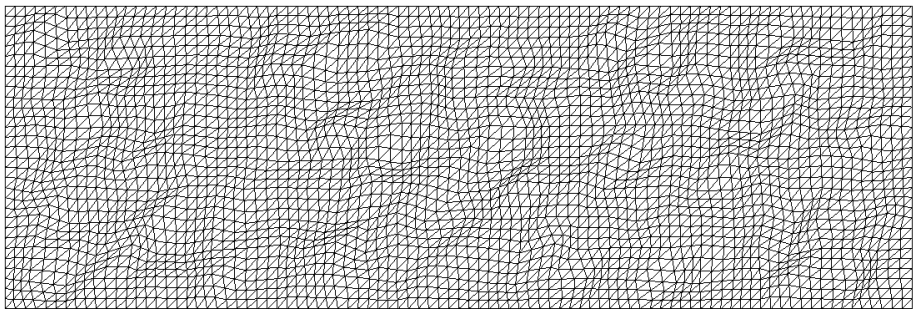
$$\Delta v = x\frac{\epsilon}{2\pi}\exp\left(\frac{1}{2}\left(1-R^2\right)\right) \tag{5.1d}$$

$$\Delta T = \frac{\left(\gamma-1\right)\epsilon^2}{8\gamma\pi^2}\exp\left(1-R^2\right) \tag{5.1e}$$

where $R = \sqrt{x^2+y^2}$ and $\epsilon = 5$ is the vortex strength. The exact solution is just a transposition of the vortex in the flow direction.

For this study, cases were run in a physical domain extending from $x : [-10, 50]$ and $y[-10, 10]$, with the vortex starting at $(0, 0)$ and ending at $(40, 0)$, taking a total of 40 seconds in physical travel time. A time step of 0.01 seconds was used. The domain is shown in the density plot of figure 5.6(b). It should be noted that the time step was imposed by the explicit third-order FR. FC does not suffer from such restrictive physical time steps, due to the implicit handling of the physical time terms. The boundary was updated with the exact solution at each physical time step. A structured triangle mesh was formed over the domain using a constant characteristic length, with the interior nodes perturbed randomly to introduce some non-uniformity. Meshes with varying characteristic lengths were used.

Figures 5.7 shows density in x at the horizontal centerline of the domain at $t = 40$. An absolute error metric is not used here, as separating out temporal and spatial errors is a difficult task. The discontinuities in the FR plots are due to it's discontinuous nature. By not forming a continuous domain from the discontinuous one, we more accurately show



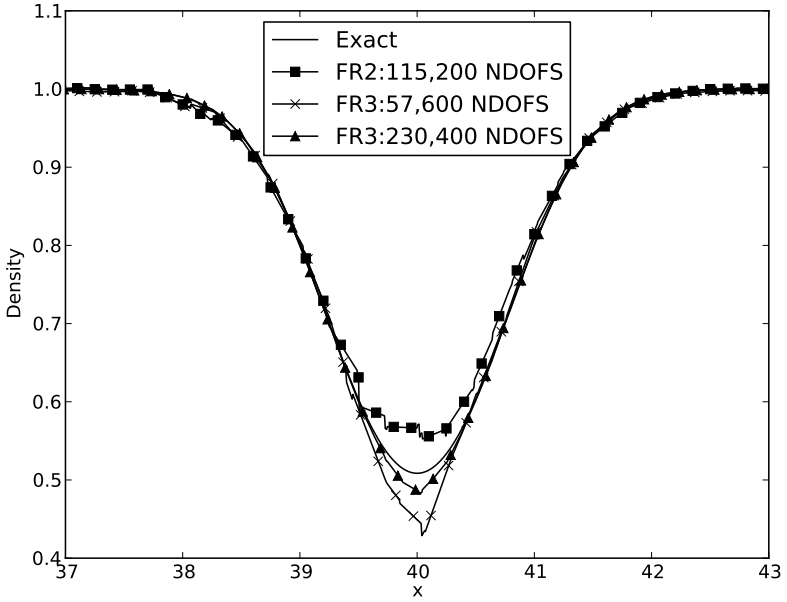
(a) Subdivided Mesh with 10×30 parent elements.

(b) Density plot at $t = 40s$ on finest FC mesh

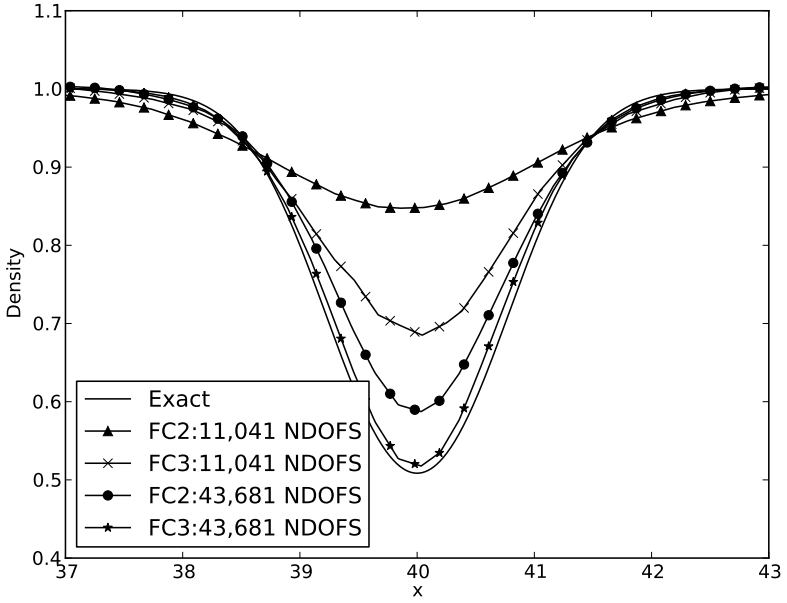
Fig. 5.6: Isentropic Vortex

the solution as it is implemented. Figure 5.7(a) shows the significant results from Flux Reconstruction. The third-order FR more closely matched the exact solution than the second-order FR with only half the NDOFS. The over-shoots at the center of the vortex are most significant, as they might represent an inability of FR to handle high gradients with a coarse mesh. This could translate into an inability of the method to correctly handle flows with shocks in them. As the mesh is refined, the center of the solution becomes much closer to the exact solution, but still is not perfect.

Figure 5.7(b) shows the significant results from Flux Correction. A huge increase in accuracy is seen between the traditional Galerkin Method and FC, indicating a substantial decrease in artificial dissipation with FC. The overshoot manifested in FR is not present with FC, suggesting a higher tolerance to high-gradient solutions than FR. The FC and FR plots should not be directly compared due to the disparaging NDOFS and completely different time-advancing methods. It should also be noted that while FR had more degrees of freedom, FC had more cells. It is believed that if the size of the cells were smaller in FR,



(a) FR Results



(b) FC Results

Fig. 5.7: Vortex results

then the overshoots would not be so prevalent.

In the matter of solution times, FR is a clear winner. The explicit and direct Runge-Kutta time integration used for FR completed the problem in a time of roughly one to two hours, while the dual-time approach used for FC took roughly 24 hours to solve. From this we conclude that if a small time step is required for the unsteady problem at hand, then a fast, explicit approach is preferable over an implicit one.

5.4 Unsteady, Viscous Flow over a Circular Cylinder

Unsteady, viscous flow over a circular cylinder is used to study the effect of curved elements on FC. This case has been studied by many researchers, with a fairly comprehensive review being written by Norberg [53]. Here we present results for flow over a cylinder with $M = 0.2$ and $Re = 100$. At this Reynold's number, vortices are shed alternately with a Strouhal number of $St = \frac{\omega L}{u} = 0.16 - 0.17$. Katz and Work [54] have already shown that FC is capable of predicting accurate Strouhal numbers on strand meshes. We use a triangularized version of the strand meshes for this case. Three meshes are employed: a 20×60, 4×60, and 60×60, where the first number gives the number of surface elements, and the second the number of radial elements. These meshes are subdivided according to the cubic procedure described. Each mesh is run using a linear version and a cubic version. Table 5.1 lists important mesh statistics. The mesh is extended out to a radial distance of 50 chord lengths. The course surface discretizations were purposefully chosen so as to exaggerate the effects of the curvature. The meshes used can be seen in figure 5.9. The meshes shown are the meshes *after* the subdivision process. A sample vorticity plot is shown in figure 5.8.

Solutions were obtained over a physical time duration of 10 seconds using a physical

Table 5.1: Cylinder Mesh Statistics

Parent Mesh Size	Total Surface Elements	Total Cells	NDOFS
20×60	60	21600	10860
40×60	120	43200	21720
60×60	180	64800	32580

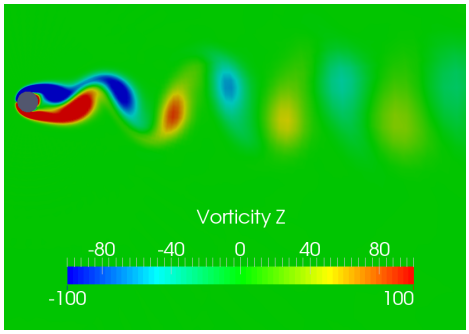
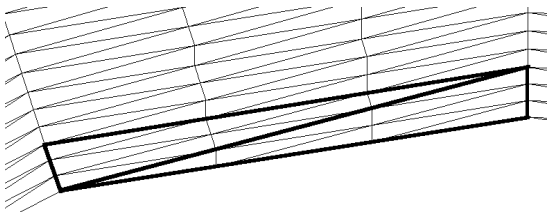
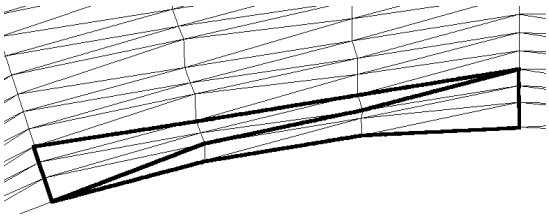


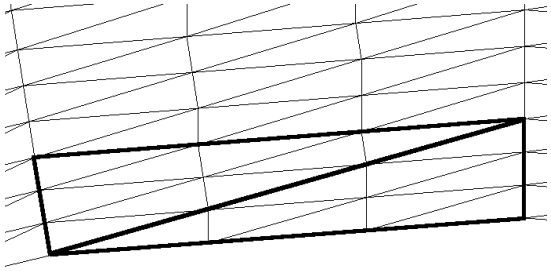
Fig. 5.8: Vorticity plot for unsteady cylinder



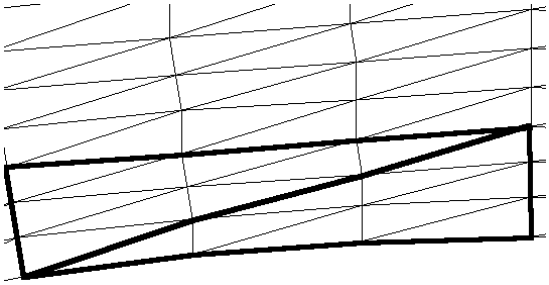
(a) 20×60 linear mesh



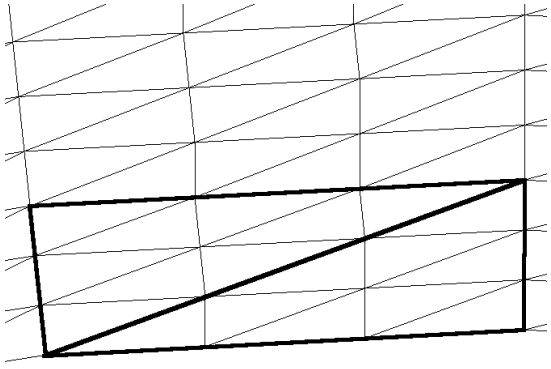
(b) 20×60 cubic mesh



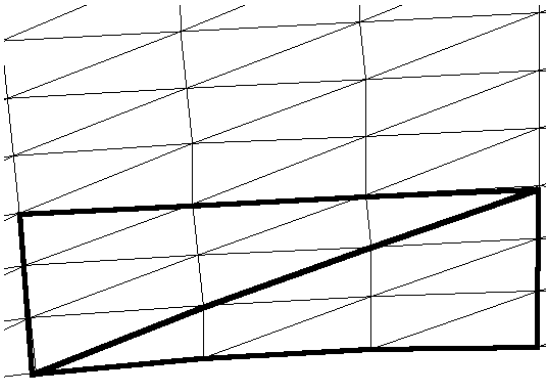
(c) 40×60 linear mesh



(d) 40×60 cubic mesh



(e) 60×60 linear mesh



(f) 60×60 cubic mesh

Fig. 5.9: Meshes used for unsteady flow over a cylinder

time step of 2×10^{-3} seconds, resulting in around 50 complete periods of oscillation with around 100 physical time steps per period. The flow is initialized with a small flow angle to initiate shedding quickly. Each test ran with 200 psuedosteps to initiate the solution, with 35 pseudosteps for each physical time step.

The first difference to be noticed is in the meshes, especially the difference between the linear and cubic meshes for the 20x60 case. With this coarse spacing, the difference between the two on the surface of the cylinder is easily seen. An unexpected side effect of the curving procedure, however, is the higher aspect ratio of the subdivided triangles in figure 5.9(b). In more extreme cases, the subdivision process could result in badly deformed cells, and this must be taken into consideration when generating meshes for FC. As the meshes are refined on the surface, this curvature effect is not seen as much, and doesn't present an issue.

The major effect of the deformed mesh was an effect on convergence. Figure 5.10 shows the density RMS for the linear 20×60 mesh and the cubic 20×60 mesh that failed to converge well. If the number of pseudosteps in the initiation step was increased to 300 from 200, then the residual for the cubic mesh closely followed that of the linear mesh.

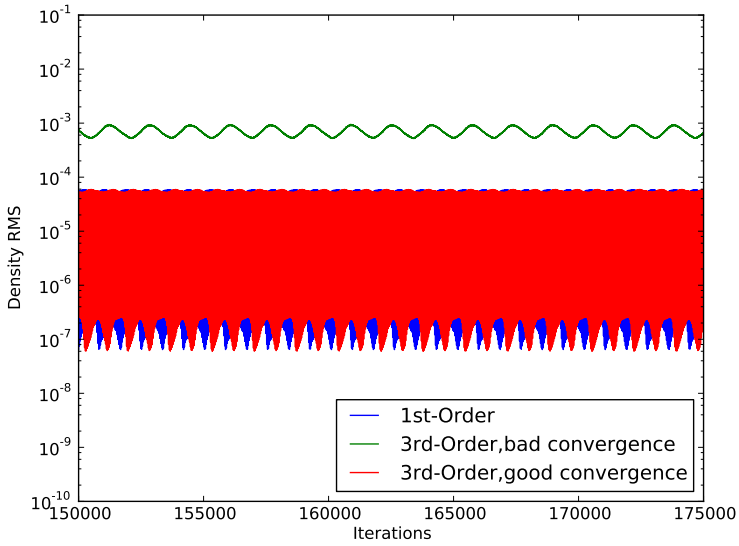


Fig. 5.10: Density convergence plot for the 20×60 mesh

The Strouhal number for each flow was computed using the lift history of the cylinder. The results are shown in table 5.2.

It is believed that the lower accuracy is due to the deformed elements generated by the subdivided triangles, and not necessarily FC itself. The subdivided triangles were linear, and didn't follow the interior reference lines of the curved element. This highlights an important feature of FC; it is highly dependent on the chosen gradient approximation.

Table 5.2: Strouhal Numbers

Mesh	St
1st-order 20×60	0.158
3rd-order 20×60	0.150
1st-order 40×60	0.164
3rd-order 40×60	0.161
1st-order 60×60	0.164
3rd-order 60×60	0.164

Chapter 6

Conclusions and Future Work

This work has examined the merits of Flux Correction. For inviscid flow, it has been compared against another high-order method known as Flux Reconstruction, a derivative of the Discontinuous Galerkin and Spectral Difference methods. The effect of curved elements on Flux Correction has also been examined.

For steady-state cases run in serial, Flux Correction is found to be much faster than Flux Reconstruction in terms of walltime and iterations to convergence. It also is shown to be superior in accuracy to Flux Reconstruction.

For unsteady cases however, the dual-time stepping method used is much slower, requiring convergence in psuedo-time for each physical time step. The explicit, direct timestepping method used in Flux Reconstruction results in much faster runtimes. If small time steps are required for temporal accuracy, then an explicit timestepping method is preferable over an implicit one, as the benefit of large time steps is lost. Dual-time stepping is a common practice in industry, thus this is not a great detriment.

Curved elements were found to have a huge impact on the accuracy of Flux Correction, which is thought to be because of the choice of gradient approximation. In the cylinder test case used, cubic meshes were found to reduce the accuracy of the estimated Strouhal number. It is believed that this is due mainly to bad elements generated in the subdivision process of the gradient approximation. This also highlights another property of Flux Correction; it is highly dependent on how high-order gradient estimates are made. Research needs to be done on high-order gradient techniques, and how Flux Correction reacts to them. High-order surface elements are still recommended, as they have been proven by others to be necessary for good accuracy. How curved surface elements affect the surrounding mesh and subsequent solution is a matter up for further research.

Flux Correction still has much to prove it's usefulness as a higher-order method in Computational Fluid Dynamics. The obvious next step is the extension to three dimensions. More work needs to be done in the mathematics, to determine if more error terms can be cancelled, therefore increasing the order of the method. Application of the method to turbulence, including LES is currently being explored. The major appeal of Flux Correction as a higher-order method is the minor changes that can be made to an existing code base in order to implement it. Thus, in the future, it is intended to see if such can be done on standard commercial and free CFD solvers, such as FLUENT or OpenFOAM.

References

[1] Wang, Z. J., "High-order Methods for the Euler and Navier–Stokes Equations on Unstructured Grids," *Progress in Aerospace Sciences*, Vol. 43, 2007, pp. 1–41.

[2] Bassi, F. and Rebay, S., "High-order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations," *Journal of Computational Physics*, Vol. 138, No. 2, 1997, pp. 251–285.

[3] Gao, H., Wang, Z. J., and Liu, Y., "A Study of Curved Boundary Representations for 2D High-order Euler Solver." *Journal of Scientific Computing*, Vol. 34, No. 3, 2008, pp. 260–286.

[4] Vincent, P. and Jameson, A., "Facilitating the Adoption of Unstructured High-order Methods Amongst a Wider Community of Fluid Dynamicists," *Mathematical Modelling of Natural Phenomena*, Vol. 6, No. 3, 2011, pp. 97–140.

[5] Jameson, A., "Advances in Bringing High-order Methods to Practical Applications in Computational Fluid Dynamics," *20th AIAA Computational Fluid Dynamics Conference*, No. AIAA 2011-3226, American Institute of Astronautics and Aeronautics, June 2011.

[6] Barth, T. and Frederickson, P., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," *American Institute for Astronautics and Aeronautics 28th Aerospace Sciences Meeting*, American Institute of Astronautics and Aeronautics, January 1990.

[7] Delanaye, M. and Liu, Y., "Quadratic Reconstruction Finite Volume Schemes on 3D Arbitrary Unstructured Polyhedral Grids," *American Institute for Astronautics and Aeronautics 14th Computational Fluid Dynamics Conference*, American Institute for Astronautics and Aeronautics, November 1999.

[8] Abgrall, R., "On Essentially Non-oscillatory Schemes on Unstructured Meshes: Analysis and Implementation," ICASE Report 92-74, National Aeronautics and Space Administration, December 1992.

[9] Barth, T. J. and Deconinck, H., *High-order Methods for Computational Physics*, Springer Verlag, Berlin, Germany, 1999.

[10] Friedrich, O., "Weighted Essentially Non-oscillatory Schemes for the Interpolation of Mean Values on Unstructured Grids," *Journal of Computational Physics*, Vol. 144, No. 1, July 1998, pp. 194–212.

[11] Hu, C. and Shu, C.-W., "Weighted Essentially Non-oscillatory Schemes on Triangular Meshes," *Journal of Computational Physics*, Vol. 150, No. 1, March 1999, pp. 97–127.

[12] Sherwin, S. J. and Karniadakis, G. E., "A New Triangular and Tetrahedral Basis for High-order (hp) Finite Element Methods," *International Journal for Numerical Methods in Engineering*, Vol. 38, No. 22, November 1995, pp. 3775–3802.

[13] Karniadakis, G. E. and Sherwin, S. J., *Spectral/hp Element Methods for Computational Fluid Dynamics*, Oxford Scientific Publications, Oxford, United Kingdom, 2005.

[14] Brooks, A. N. and Hughes, T. J. R., "Streamline Upwind/Petrov-Galerkin formulations for Convection dominated Flows with particular emphasis on the Incompressible Navier-Stokes Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, No. 1-3, September 1982, pp. 199–259.

[15] Cockburn, B. and Shu, C.-W., "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, pp. 2440–2463.

[16] Cockburn, B., Karniadakis, G. E., and Shu, C.-W., *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Springer, Berlin, Germany, 2000.

[17] Wang, Z. J. and Liu, Y., "Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids II: Extension to Two-dimensional Scalar Equation," *Journal of Computational Physics*, Vol. 179, No. 2, July 2002, pp. 665–697.

[18] Wang, Z. J. and Gao, H., "A Unifying Lifting Collocation Penalty Formulation Including the Discontinuous Galerkin, Spectral Volume/Difference Methods for Conservation Laws on Mixed Grids," *Journal of Computational Physics*, Vol. 228, No. 21, November 2009, pp. 8161–8186.

[19] Liang, C., Jameson, A., and Wang, Z. J., "Spectral Difference Method for Compressible Flow on Unstructured Grids with Mixed Elements," *Journal of Computational Physics*, Vol. 228, No. 8, May 2009, pp. 2847–2858.

[20] Liu, Y., Vinokur, M., and Wang, Z. J., "Spectral Difference Method for Unstructured Grids I: Basic Formulation," *Journal of Computational Physics*, Vol. 216, No. 2, August 2006, pp. 780–801.

[21] Huynh, H. T., "A Flux Reconstruction Approach to High-order Schemes Including Discontinuous Galerkin Methods," *18th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2007.

[22] Huynh, H. T., "A Reconstruction Approach to High-order Schemes Including Discontinuous Galerkin for Diffusion," *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, January 2009.

[23] Katz, A. and Sankaran, V., "An Efficient Correction Method to Obtain a Formally Third-order Accurate Flow Solver for Node-Centered Unstructured Grids," *Journal of Scientific Computing*, Vol. 51, No. 2, 2012, pp. 375–393.

[24] Pincock, B. and Katz, A., "High-order Flux Correction for Viscous Flows on Arbitrary Unstructured Grids," *21st AIAA Computational Fluid Dynamics Conference*, The American Institute of Aeronautics and Astronautics, June 2013.

[25] Roache, P. J., "Code Verification by the Method of Manufactured Solutions," *Journal of Fluids Engineering*, Vol. 124, March 2002, pp. 4–10.

[26] Allaneau, Y. and Jameson, A., "Connections Between the Filtered Discontinuous Galerkin Method and the Flux Reconstruction Approach to High Order Discretizations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 200, No. 4952, 2011, pp. 3628 – 3636.

[27] Castonguay, P., Williams, D. M., Vincent, P. E., Lopez, M., and Jameson, A., "On the Development of a High-order, Multi-GPU Enabled, Compressible Viscous Flow Solver for Mixed Unstructured Grids," *20th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2011.

[28] Dey, S., O'Bara, R. M., and Shepard, M. S., "Curvilinear Mesh Generation in 3D," *Proceedings of the Eighth International Meshing Roundtable*, John Wiley & Sons, Hoboken, New Jersey, 1999, pp. 407–417.

[29] Allen, C., "Parallel Flow-Solver and Mesh Motion Scheme for Forward Flight Rotor Simulation," *24th AIAA Applied Aerodynamics Conference*, No. AIAA 2006-3476, American Institute of Aeronautics and Astronautics, June 2006.

[30] Erwin, T., Anderson, W. K., Kapadia, S., and Wang, L., "Three Dimensional Stablilized Finite Elements for Compressible Navier-Stokes," *20th AIAA Computational Fluid Dynamics Conference*, No. AIAA 2011-3411, AIAA, June 2011.

[31] Solin, P., Segeth, K., and Dolezel, I., *Higher-order Finite Element Methods*, Chapman & Hall/CRC, Boca Raton, Florida, 2004.

[32] Jameson, A., Baker, T., and Weatherill, N., "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," *AIAA paper* 83-0103, AIAA 24th Aerospace Sciences Meeting, Reno, NV, January 1986.

[33] Haselbacher, A., *Grid-transparent Numerical Method for Compressible Viscous Flows on Mixed Unstructured Grids*, Ph.D. thesis, Loughborough University, 1999.

[34] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-stepping Schemes," *AIAA 14th Fluid and Plasma Dynamic Conference*, No. AIAA 1981-1259, AIAA, June 1981.

[35] Jameson, A. and Mavriplis, D., "Finite Volume Solution of the Two-dimensional Euler Equations on a Regular Triangular Mesh," *AIAA Journal*, Vol. 24, 1986, pp. 611–618.

[36] Brandt, A., "Multi-level Adaptive Solutions to Boundary-value Problems," *Mathematics of Computation*, Vol. 31, No. 138, April 1977, pp. 333–390.

[37] Allmaras, S., "Lagrange Multiplier Implementation of Dirichlet Boundary Conditions in Compressible Navier-Stokes Finite Element Methods," *17th AIAA Computational Fluid Dynamics Conference*, No. AIAA 2005-4714, AIAA, June 2005.

[38] Folkner, D. E., *Improvement in Computational Fluid Dynamics through Boundary Verification and Preconditioning*, Master's thesis, Utah State University, 2013.

[39] Williams, D. M., *Energy Stable High-order Methods for Simulating Unsteady, Viscous, Compressible Flows on Unstructured Grids*, Ph.D. thesis, Stanford University, 2013.

[40] Castonguay, P., Vincent, P. E., and Jameson, A., "A New Class of High-order Energy Stable Flux Reconstruction Schemes for Triangular Elements," *Journal of Scientific Computing*, Vol. 51, No. 1, April 2012, pp. 224–256.

[41] Hesthaven, J. S. and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Springer Verlag, Berlin, Germany, 2007.

[42] Peraire, J. and Persson, P., "The Compact Discontinuous Galerkin (CDG) Method for Elliptic Problems," *SIAM Journal on Numerical Analysis*, Vol. 30, 2009, pp. 1806–1824.

[43] Arnold, D. N., "An Interior Penalty Finite Element Method with Discontinuous Elements," *SIAM Journal on Numerical Analysis*, Vol. 19, 1982, pp. 742–760.

[44] Bassi, F. and Rebay, S., "A High-order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations," *Journal of Computational Physics*, Vol. 131, 1997, pp. 267–279.

[45] Bassi, F., Rebay, S., Mariotti, G., Pedinotti, S., and Savini, M., "A High-order Accurate Discontinuous FInite Element Method for Inviscid and Viscous Turbomachinery Flows," *2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics*, 1997.

[46] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[47] Rusanov, V. V., "Calculation of Interaction of Non-Steady Shock Waves with Obstacles," *Journal of Computational Math and Physics USSR*, Vol. 1, 1961, pp. 261–279.

[48] Castonguay, P., Vincent, P. E., and Jameson, A., "Application of High-order Energy Stable Flux Reconstruction Schemes to the Euler Equations," *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, No. AIAA 2011-686, American Institute of Aeronautics and Astronautics, January 2011.

[49] Taylor, M. A., Wingate, B. A., and Bos, L. P., "Several New Quadrature Formulas for Polynomial Integration in the Triangle," Online, Feb 2007, http://arxiv.org/abs/math/0501496v2, Accessed October 2013.

[50] Carpenter, M. H. and Kennedy, C. A., "Fourth-order 2N-Storage Runge-Kutta Schemes," NASA Technical Memorandum 109112, NASA, June 1994.

[51] Nastase, C. R. and Mavriplis, D. J., "High-order Discontinuous Galerkin Methods Using an hp-Multigrid Approach," *Journal of Computational Physics*, Vol. 213, March 2006, pp. 330–357.

[52] Shu, C.-W., "Essentially Non-oscillatory and Weighted Non-oscillatory Schemes for Hyperbolic Conservation Laws," *Lecture Notes in Mathematics*, Vol. 1697, 1998, pp. 325–432.

[53] Norberg, C., "Fluctuating Lift on a Circular Cylinder: Review and New Measurements," *Journal of Fluids and Structures*, Vol. 17, 2003, pp. 57–96.

[54] Katz, A. and Work, D., "High-order Flux Correction/Finite Difference Schemes for Strand Grids," *AIAA 52nd Aerospace Sciences Meeting*, No. AIAA 2014-0937, AIAA, January 2014.

Appendix

This appendix describes a simple derivation of the 1D linear Galerkin node-centered method for a hyperbolic equation. This method starts by discretizing the domain into several elements, each of which starts and ends with a node, as demonstrated in figure 1. The governing equation is multiplied by an arbitrary test function ϕ, and then integrated over the entire domain Ω with boundary Γ:

$$\int_\Omega \phi u_t \, \mathrm{d}\Omega + \int_\Omega \phi f_x \, \mathrm{d}\Omega = \int_\Omega \phi S(x) \, \mathrm{d}\Omega. \tag{1}$$

For an arbitrary node i we define ϕ as a "hat" function which is linear over the two elements touching node i and zero everywhere else in the domain. This causes the integral over the domain to only exist in the vicinity of i, and breaks the domain integral into two integrals of elements Ω_A and Ω_B. The terms in equation (1) are labeled, from left to right: The unsteady term, the flux term, and the source term.

Flux Term

The flux term is integrated by parts:

$$\int_{\Omega_{AB}} \phi u_t \, \mathrm{d}\Omega + [\phi f]_{\Gamma_{AB}} - \int_{\Omega_{AB}} \phi_x f \, \mathrm{d}\Omega = \int_{\Omega_{AB}} \phi S(x) \, \mathrm{d}\Omega. \tag{2}$$

The boundary term only exists if node i is on a boundary. If the node is on a left boundary, Ω_A does not exist, and thus only the boundary term for Ω_B exists. If the node is on a right boundary, Ω_B does not exist, leaving only element Ω_A. The boundary flux term can be

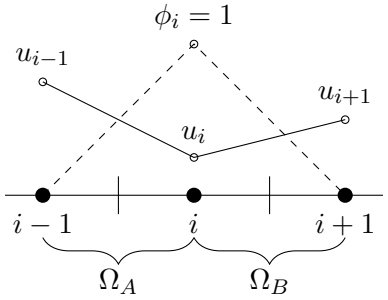


Fig. 1: Discretization used in the derivation of the Linear Galerkin Method

summarized as follows:

$$[\phi f]_{\Gamma_A} = f_i, \tag{3a}$$

$$[\phi f]_{\Gamma_B} = -f_i, \tag{3b}$$

$$[\phi f]_{\Gamma_{AB}} = 0. \tag{3c}$$

Handling the integrated flux term is done by recognizing that the derivative of ϕ is constant, where

$$\phi_x|_A = \frac{1}{x_i - x_{i-1}} = \frac{1}{\Delta x_A}, \tag{4a}$$

$$\phi_x|_B = -\frac{1}{x_{i+1} - x_i} = -\frac{1}{\Delta x_B}, \tag{4b}$$

where $\Delta x_A = x_i - x_{i-1}$ and $\Delta x_B = x_{i+1} - x_i$. The integrated flux terms for elements Ω_A and Ω_B become

$$\int_{\Omega_A} \phi_x f \, \mathrm{d}\Omega = \phi_x|_A \int_{\Omega_A} f \, \mathrm{d}\Omega, \tag{5a}$$

$$\int_{\Omega_B} \phi_x f \, \mathrm{d}\Omega = \phi_x|_B \int_{\Omega_B} f \, \mathrm{d}\Omega. \tag{5b}$$

The trapezoidal rule is used to integrate f because f has been chosen to be linear:

$$\int_{\Omega_A} f \, \mathrm{d}\Omega = \frac{1}{2} \left(f_i + f_{i-1} \right) \Delta x_A, \tag{6a}$$

$$\int_{\Omega_B} f \, \mathrm{d}\Omega = \frac{1}{2} \left(f_{i+1} + f_i \right) \Delta x_B. \tag{6b}$$

Substituting equations (6) and (4) into (5),

$$\phi_x|_A \int_{\Omega_A} f \, \mathrm{d}\Omega = \frac{1}{2} \left(f_i + f_{i-1} \right), \tag{7a}$$

$$\phi_x|_B \int_{\Omega_B} f \, \mathrm{d}\Omega = -\frac{1}{2} \left(f_i + f_{i+1} \right). \tag{7b}$$

Equation (7) can be seen as averages of the flux at the midpoints of the elements. Thus, the final form can be written as:

$$\int_{\Omega} \phi_x f \, \mathrm{d}\Omega = -\left(F_{i+1/2} - F_{i-1/2}\right). \tag{8}$$

where $\Delta x_i = \frac{1}{2}\left(\Delta x_A + \Delta x_B\right)$. This form looks very similiar to a FV method, and we interprete it as a finite volume surrounding node i and bounded by the faces at $i + 1/2$ and $i - 1/2$. This formulation will be different at boundaries. The case of i at the boundary can easily be derived from the same methodology.

Unsteady/Source Term

The unsteady and source terms have the same form, and are treated the same. For a fixed mesh, the time derivative can be moved outside the integral, as follows:

$$\int_{\Omega_{AB}} \phi u_t \, \mathrm{d}\Omega = \frac{\partial}{\partial t} \int_{\Omega} \phi u \, \mathrm{d}\Omega \tag{9}$$

In a traditional FE fashion, ϕ and u are interpolated in an element Ω_k with left and right boundary values Γ_k^L and Γ_k^R using shape functions:

$$\phi_k = N_1(\eta)\, \phi_{\Gamma_k^L} + N_2(\eta)\, \phi_{\Gamma_k^R}, \tag{10a}$$

$$u_k = N_1(\eta)\, u_{\Gamma_k^L} + N_2(\eta)\, u_{\Gamma_k^R}. \tag{10b}$$

The shape functions are defined on a standard element using η as a parameterizing variable. On a standard element, η varies from 0 to 1, starting at the left and moving to the right. The shape functions for a linear element are

$$N_1(x) = 1 - \eta, \qquad N_2(\eta) = \eta. \tag{11}$$

The transformation of the elements Ω_A and Ω_B from x to η is given by:

$$\left.\frac{dx}{d\eta}\right|_A = x_i - x_{i-1} = \Delta x_A, \tag{12a}$$

$$\left.\frac{dx}{d\eta}\right|_B = x_{i+1} - x_i = \Delta x_B. \tag{12b}$$

Transforming the integrals from x to η, substituting the approximations from equation (10) into equation (9), applying the exact values of ϕ, and integrating separately over elements Ω_A and Ω_B, the following discretized unsteady/source term is arrived at:

$$\frac{\partial}{\partial t}\int_{\Omega_{AB}} \phi u \, \mathrm{d}\Omega = \frac{\partial}{\partial t}\left[\frac{2}{3}u_i \Delta x_i + \frac{1}{6}\left(\Delta x_A u_{i-1} + \Delta x_B u_{i+1}\right)\right], \tag{13}$$

where $\Delta x_i = \frac{1}{2}\left(\Delta x_A + \Delta x_B\right)$. Equation (13) is the equation used for the source terms. (Drop the time derivative and change u to S.) For the unsteady terms though, this would result in a full matrix to be solved at each time step. This set of equations can be decoupled by summing the coefficents onto u_i and eliminating the diagonals. This approximation is called *mass lumping* in the more rigorous FE derivation. This approximation makes the final discretization become

$$\frac{\partial}{\partial t}\int_{\Omega_{AB}} \phi u \, \mathrm{d}\Omega = \Delta x_i \frac{\partial}{\partial t}\left(u_i\right). \tag{14}$$

The coefficients are different with i on a boundary.

Final Form

The differential equation in Galerkin form can be written by substituting equation (14), (13), and (8) into (2).

$$\frac{\partial u_i}{\partial t} - \frac{1}{\Delta x_i}\left(F_{i+1/2} - F_{i-1/2}\right) = \frac{1}{\Delta x_i}\left[\frac{2}{3}\Delta x_i S_i + \frac{1}{6}\left(S_{i-1}\Delta x_A + S_{i+1}\Delta x_B\right)\right]. \tag{15}$$

This equation is then solved for all nodes i. Both the traditional Galerkin method and flux correction are identical up until this point. Please see chapter 3.