8-8-2009

On mesh quality considerations for the discontinuous Galerkin method

Eric M. Collins

On mesh quality considerations for the discontinuous Galerkin method

Comments

manufactured solutions||mesh generation||solver verification

ON MESH QUALITY CONSIDERATIONS FOR THE DISCONTINUOUS GALERKIN METHOD

By

Eric M. Collins

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computational Engineering
in the Bagely College of Engineering

Mississippi State, Mississippi

August 2009

ON MESH QUALITY CONSIDERATIONS FOR THE

DISCONTINUOUS GALERKIN METHOD

By

Eric M. Collins

Approved:

Edward A. Luke Associate Professor of Computer Science and Engineering (Director of Dissertation)	David S. Thompson Associate Professor of Aerospace Engineering (Committee Member)
Pasquale Cinnella Professor of Aerospace Engineering (Committee Member)	Seth Oppenheimer Professor of Mathematics and Statistics (Committee Member)
J. Mark Janus Associate Professor of Aerospace Engineering (Committee Member and Graduate Coordinator for Computational Engineering)	Sarah A. Rajala Dean of the Bagley College of Engineering

Name: Eric M. Collins

Date of Degree: August 8, 2009

Institution: Mississippi State University

Major Field: Computational Engineering

Major Professor: Dr. Edward A. Luke

Title of Study: ON MESH QUALITY CONSIDERATIONS FOR THE DISCONTINUOUS GALERKIN METHOD

Pages in Study: 144

Candidate for Degree of Doctor of Philosophy

It is widely accepted that the accuracy and efficiency of computational fluid dynamics (CFD) simulations is heavily influenced by the quality of the mesh upon which the solution is computed. Unfortunately, the computational tools available for assessing mesh quality remain rather limited. This report describes a methodology for rigorously investigating the interaction between a flow solver and a variety of mesh configurations for the purposes of deducing which mesh properties produce the best results from the solver. The techniques described herein permit a more detailed exploration of what constitutes a quality mesh in the context of a given solver and a desired flow regime.

In the present work, these newly developed tools are used to investigate mesh quality as it pertains to a high-order accurate discontinuous Galerkin solver when it is used to compute inviscid and high-Reynolds number flows in domains possessing smoothly curving boundaries. For this purpose, two flow models have been generated and used to conduct parametric studies of mesh configurations involving curved elements. The results of these studies allow us to make some observations regarding mesh quality when using the discontinuous Galerkin method to solve these types of problems. Briefly, we have found that for inviscid problems, the mesh elements used to resolve curved boundaries should be at least third order accurate. For viscous problems, the domain boundaries must be approximated by mesh elements that are of the same order as the polynomial approximation of

the solution if the theoretical order of accuracy of the scheme is to be maintained. Increasing the accuracy of the boundary elements to at least one order higher than the solution approximation typically results in a noticeable improvement in the computed error norms. It is also noted that C^1-continuity of the mesh is not required at element interfaces along the boundary.

DEDICATION

For Jennifer, Katelyn, Isaac, and Miranda.

ACKNOWLEDGMENTS

I would like to acknowledge the assistance of my advisor, Dr. Edward Luke for his insights and encouragement and to the members of my committee for their patience and willingness to answer the occasional odd question. My thanks also go out to several of my co-workers who provided me with a sounding board when I just needed to get some thoughts out of my head.

I would like express my gratitude to the Computational Simulation and Design Center (formerly the MSU-NSF Engineering Research Center for Computational Field Simulations) for its nearly continual support of my work during my time at Mississippi State University. I am also grateful for the support of NASA's Constellation University Institutes Program (CUIP) under the management of Claudia Meyer and Jeff Ryback, and to Jeff West for serving as technical monitor and for providing helpful suggestions and support.

Last, but certainly not least, I am fortunate to have a very patient and supportive family. I deeply appreciate all that my wife has done for me over the years, and I try not to think of what it would have been like to go through this without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

General Notation

q, α	Scalar valued quantity
$\vec{u}$	Vector valued quantity
$\tilde{\sigma}$	Tensor valued quantity
$\mathbf{M}$	Matrix valued quantity
$\mathbf{q}$	Array of scalar values
$\vec{\mathbf{f}}$	Array of vector values
$\vec{u} \cdot \vec{v} = u_i v_i$	Inner product of two vectors (scalar valued)
$\vec{u}\vec{v} = u_i v_j \vec{a}_i \vec{a}_j$	Outer product of two vectors (tensor valued)
$\text{div}(\vec{u}) = \frac{\partial u_i}{\partial x_i}$	Divergence of a vector (scalar valued)
$\text{div}(\tilde{\sigma}) = \frac{\partial \sigma_{ij}}{\partial x_i} \vec{a}_j$	Divergence of a tensor (vector valued)
$\text{grad}(q) = \frac{\partial q}{\partial x_i} \vec{a}_i$	Gradient of a scalar (vector valued)
$\text{grad}(\vec{u}) = \frac{\partial u_i}{\partial x_j} \vec{a}_i \vec{a}_j$	Gradient of a vector (tensor valued)
x, y, z	Global Cartesian coordinates
$\hat{\imath}, \hat{\jmath}, \hat{k}$	Unit vectors for Cartesian coordinates
ξ, η, ζ	Local parametric coordinates
t	Time
δ_{ij}	Kronecker delta
$\tilde{I}$	Identity tensor
f_h	Polynomial approximation to some function f
$\vec{n}$	Unit normal vector (orthogonal to a surface)
$\mathcal{P}^k$	The space of all k^{th}-degree polynomials
$d\Omega$	Volume element
$d\Gamma$	Surface element

Fluid Dynamic Variables

a	Speed of sound
a_0	Speed of sound at stagnation conditions
c_p	Specific heat at constant pressure
c_v	Specific heat at constant volume
c_f	Skin friction
e	Total energy (thermal & kinetic)
e_0	Internal energy (thermal)
k	Coefficient of thermal conductivity
P	Absolute pressure
P_0	Stagnation pressure
P_{ref}	Reference pressure
R	Species gas constant
$\hat{R}$	Universal gas constant
Re	Reynolds number
T	Fluid temperature
T_0	Stagnation temperature
T_{ref}	Reference temperature
$\vec{u}$	Fluid velocity vector
u, v, w	Cartesian fluid velocity components
u_τ	Friction velocity
u^+	Scaled flow speed (parallel to wall)
y^+	Scaled wall distance
γ	Adiabatic exponent
μ	Coefficient of shear viscosity
ρ	Fluid density
ρ_0	Stagnation density
$\tilde{\sigma}$	Shear stress tensor
τ_w	Wall shear stress

CHAPTER 1

INTRODUCTION

1.1 Motivation

The past few decades have seen a remarkable growth in the availability of high-performance computing hardware, and the sophistication of Computational Fluid Dynamics (CFD) solver capabilities. Along with this growth, the demand from practicing engineers for simulations with greater fidelity over a wider range of flow regimes and domain complexity has also increased.

Despite this impressive progress, though, there still remains some fluid flow phenomena for which accurate simulations cannot be practically computed. One of the most difficult flow regimes in which to obtain highly accurate solutions is very high-Reynolds-number flows in and around complex geometry configurations. These flows typically exhibit unsteady turbulent behavior characterized by coherent vortex structures with widely varying length scales. While many existing computational tools can be brought to bear on some aspects of these problems, it still remains impractical to resolve much of the rich character of the physical phenomena resulting from the turbulent behavior of these flows.

The limitations on current solvers are mostly due to a sharp rise in computational cost associated with the increase in complexity of the fluid flow. At higher Reynolds numbers, the length scales of the coherent flow features diminish rapidly. Fully resolving the flow

field using Direct Numerical Simulation (DNS) [50] would require the use of highly refined meshes. This is perhaps the most intuitive way to approach the problem; however, the amount of computational resources required is often prohibitive. When explicit time integration is used, smaller mesh spacing will also incur a corresponding reduction in the size of allowable time steps. For implicit time integration, the reduction in mesh scales may cause the system matrix to become stiff. The increased stiffness may then make it more difficult for the linear system solver to reach a converged solution. Thus, a linear increase in Reynolds number will typically incur an exponential rise in computational expense.

This downward pressure on mesh resolution can be partially mitigated through the use of turbulence modeling. Turbulence modeling relies on the following observation: as large-scale turbulent flow structures decay into smaller-scale turbulent eddies, their dissipative behavior becomes increasingly uniform. Turbulence models attempt to approximate the more uniform viscous dissipation effects rather than to fully resolve them.

Among turbulence modeling techniques, Reynolds-Averaged Navier-Stokes (RANS) methods are the most frequently used for practical engineering problems [72]. Solvers based on the RANS equations utilize a modified set of governing equations which attempts to model the turbulent dissipation at all length scales. Depending on the specific turbulence model used to close the RANS equations, there will typically be one or more additional state variables that are evolved along with the conservative flow variables. RANS methods are fairly economical to compute and typically provide good results for steady and moderately steady flows with no separations. However, RANS is often less effective on flows with strong separations and large-scale coherent eddy structures [33]. The temporal

and spatial averaging schemes employed by RANS methods have a tendency to generate excessive dissipation in these regions. Unsteady flow features are typically over-damped and prematurely dissipated [47].

Large Eddy Simulation (LES) methods provide an alternative to RANS for simulations involving large unsteady features [31]. LES methods employ a spatial filter to distinguish between the large-scale unsteady flow structures and the smaller-scale eddies that are responsible for more uniform dissipative behavior. The large scale features are then fully resolved by the Navier-Stokes solver while the smaller scale features are modeled. The LES methods have been shown to produce superior results to that of RANS methods in regions involving large-scale unsteady features. However, when attempting to resolve the flow in regions of high solution anisotropy, such as those found in thin attached boundary layers, the LES method begins to encounter resource constraints similar to that of the DNS method.

In recent years, the hybrid RANS/LES approach has begun to show great potential for economically simulating a wide range of turbulent behavior with greater accuracy than previously possible [62]. In the hybrid approach, RANS is used to evolve the turbulent behavior in the thin attached boundary layers, while LES is used to capture large unsteady flow features. The LES subgrid-scale turbulence model is then used to model the behavior of the more isotropic turbulent decay in regions away from the boundaries where the relevant length scales remain relatively large.

The effective application of hybrid RANS/LES techniques to the simulation of high Reynolds number flows is currently an active area of research. The full potential of hybrid

RANS/LES remains difficult to assess, however, with the current generation of second-order flow solvers. On unstructured meshes, second-order solvers have a tendency to produce too much numerical dissipation over a wide range of mesh scales. It has been observed in some instances that the contribution of the LES sub-grid scale turbulence models are of the same order of magnitude as the numerical errors generated by the solver. This has prompted some researchers to forgo the sub-grid scale turbulence model in favor of allowing the numerical dissipation of the solver to control the rate of turbulent energy cascade as in the Monotone Integrated Large Eddy Simulation (MILES) approach [18].

While many researchers continue to search for practical ways to make LES work well with second-order solvers [33, 47], others have begun looking to higher-order accurate solvers to provide a more robust platform for evaluating the subgrid scale turbulence models [29, 64]. The increased rate by which errors are reduced in higher-order methods allows them to attain desired levels of error tolerances on much coarser meshes. The availability of higher-order derivatives has also been considered as an advantage for formulating more sophisticated turbulence models.

Yet, despite their favorable numerical properties, higher-order methods have still not seen wide acceptance for performing CFD calculations. In the past these methods have tended to be somewhat less flexible in terms of how the domain is discretized. High-order finite difference methods require nearly orthogonal alignment of the grid points, while some finite volume based methods (e.g. ENO, WENO) rely on wide mesh stencils to form high-order solution reconstructions. Because of their reliance on extended stencils, many of these methods require specialized treatment of boundary conditions in order to maintain

high-order accuracy. Methods which rely on extended stencils are also more difficult to parallelize due to their increased dependence on non-local data.

Recently, the Discontinuous Galerkin (DG) method has emerged as a promising candidate for computing high-order accurate solutions to unsteady flow fields on unstructured meshes [64]. Solvers based on the DG method have already been used to compute solutions for a wide variety of flow regimes including some fairly ambitious attempts at simulating turbulent, high-Reynolds number flows. To date, the DG method has been used to simulate turbulent flow using DNS [28, 29], LES [12, 27], and RANS equations with one- [46, 49], and two-equation [5, 36, 38] turbulence closure models. There has also been some recent progress in high-order shock capturing techniques [3, 52]. With these advancements, the DG method is well on its way to becoming a robust platform for the investigation of transonic and supersonic, high Reynolds number flow problems.

There is ample evidence in the literature that using linear elements to resolve curved boundaries is problematic for DG methods [7, 8, 36, 42, 43]. There is considerably less information, however, regarding the precise relationship of high-order elements to the overall accuracy of the solver. With the DG method being used to solve very high-Reynolds number flows, it is only natural to suspect that geometric accuracy of the mesh elements near the boundary will be even more critical, especially in regions of high solution anisotropy such as with thin attached boundary layers.

Numerous researchers have reported the need to use curved mesh elements near smoothly curving boundaries in order to maintain the desired accuracy of the DG solver [7, 8, 36, 42, 43]. For each report, there typically follows a description of how the problem was

overcome by curving the mesh elements adjacent to the boundary and adjusting the algorithms accordingly. Yet, despite the fact that this is clearly a mesh quality issue, very little information has been published regarding the minimum or optimum mesh configurations for discretizing the domain near smoothly curving boundaries. The present work will attempt to remedy this situation by developing a methodology for objectively comparing multiple meshing strategies for a given domain and fluid flow regime. This methodology is used to evaluate the quality of the solutions computed by a DG solver on various mesh configurations. The results of these numerical experiments are then used to make some recommendations for generating meshes for use with the high-order accurate DG method.

1.2 Mesh Quality

Mesh quality may be loosely defined as a set of criteria which computational meshes should satisfy in order to allow the solver to produce the most accurate numerical solution utilizing a reasonable amount of computing resources. Most of what is currently known about mesh quality derives from the analysis of the Finite Difference (FD), Finite Volume (FV) and Finite Element (FE) methods [66]. Guided by this analysis and decades of experience with numerical solvers based on these methods, the CFD community has arrived at a general consensus on what constitutes good mesh quality for both structured and unstructured meshes [65].

In general, the following mesh quality criteria will apply to most numerical CFD solvers [65, 66]:

- The spacing of mesh nodes should not change too rapidly in each coordinate direction (i.e. adjacent elements should be approximately the same size).

- Elements with large aspect ratios should be used sparingly, and only in locations where the relevant time and length scales require their use.
- When utilizing structured mesh topologies, elements should not be excessively sheared or twisted, (i.e. grid lines should be very nearly orthogonal).
- The mesh elements should conform to all domain boundaries as accurately as possible, and where possible they should also align with dominant flow features in the domain (e.g. strong shocks, thin boundary layers).

Although DG owes much of its heritage to both the FE and FV methods, there is considerably more redundant information present in the DG method which allows for a greater degree of decoupling of the degrees of freedom in each element. This increased decoupling allows for considerably more flexibility in how the domain is discretized. One particular advantage afforded by this is hp-adaptivity in which high-order accuracy may be traded with mesh resolution to improve the robustness and efficiency of the solver.

1.3 Research Objectives

The primary objective of this research is to establish a minimum set of meshing criteria which will preserve the order of accuracy of the solver. The secondary objective is to determine if there exist any additional quality criteria that can be shown to improve the accuracy of the solution and/or the stiffness of the numerical solver. One specific area that merits a focused investigation is how accurately curved domain boundaries must be approximated by the mesh geometry in order to maintain the accuracy of the scheme.

1.4 Overview

The following chapters will provide a description of the DG solver implementation and the diagnostic tools that have been developed to conduct the accuracy and stability

tests on the solver. The results of the verification studies are presented along with observations regarding the quality of the various meshing strategies as they pertain to the overall accuracy and stiffness of the solver for each problem.

In Chapter 2, the literature is reviewed and the relevant context for the current investigation is provided. Chapter 3 presents the governing equations and an overview of the DG method. The methodology utilized to verify the implementation of the solver is given in Chapter 4 along with a description of the techniques used to evaluate the accuracy and stiffness of the solver. The numerical results are presented in Chapter 5 along with a discussion of the mesh quality recommendations which are supported by the data. Some conclusions and suggestions for further study are provided in Chapter 6.

CHAPTER 2

RELATED WORK

2.1 Discontinuous Galerkin

The Discontinuous Galerkin (DG) method was originally formulated by Reed and Hill [54] in 1973 to solve the neutron transport problem. Some preliminary analysis of the method was conducted the following year by Lesaint and Raviart [40]. The method saw limited use for the next several years before being revived in the late 1980's and subsequently extended to solve many other types of problems.

In 1989, Cockburn and Shu began publishing a series of papers [19, 22, 23, 26] in which the capabilities of the DG method were steadily expanded upon. The method was extended from solving scalar, linear, and hyperbolic equations to systems of nonlinear, hyperbolic, conservation laws in multiple dimensions. The resulting Runge-Kutta Discontinuous Galerkin (RKDG) method enabled the development of the first generation of explicit solvers capable of computing solutions of the Euler equations [8, 19, 22, 24]. A detailed review of these developments may be found in the survey article by Cockburn *et al.* [20].

The first inter-element flux formulas for handling the viscous terms of the Navier-Stokes equations were proposed by Bassi and Rebay [7]. Despite the successful simulation of a number of convection-diffusion problems, this formulation (BR1) was later analyzed

and shown to have poor convergence rates for odd-order polynomial approximations and to be unstable for certain model problems. Bassi and Rebay then developed a new interface diffusion operator (BR2) which corrected these short-comings and also resulted in a more compact scheme [9].

Several alternative schemes for evaluating the diffusive fluxes were developed during this time by Cockburn and Shu [25, 26], Lomtev and Karniadakis [41], and Baumann and Oden [11]. Evaluation and analysis of these methods was carried out by Zhang and Shu [73], and Bassi and Rebay [10]. This analysis showed that Bassi and Rebay's initial scheme (BR1) was inconsistent for the Poisson problem and weakly unstable for the Laplace problem. Their second scheme (BR2), however, did not possess these limitations. Each of these schemes fall into the category of interior penalty (IP) methods. Interior penalty methods include a dissipative term which penalizes the solution for being discontinuous at the element interface. This dissipative term provides stabilization for the scheme.

More recently, van Leer [68, 69] has devised a diffusive operator based on a higher-order continuous reconstruction of the solution at the cell interface. Peraire and Persson have also revisited the Local Discontinuous Galerkin (LDG) method of Cockburn and Shu and have produced a more compact scheme referred to as the Compact Discontinuous Galerkin method (CDG) [51].

The DG method has also been extended to include turbulence modeling. The Reynolds Averaged Navier-Stokes (RANS) equations have been successfully integrated into DG solvers by Bassi and Rebay [4, 5, 6, 9]. In their method, the k-ω model is slightly modi-

fied and the turbulence parameters k and $\ln(\omega)$ are discretized using polynomial approximations in exactly the same manner as the conservative variables. The work of Bassi and Rebay has demonstrated that the DG method may be effectively utilized to simulate turbulent flows in complex domains and on problems of engineering interest.

Collis and Ghayour have utilized the DG method for computing complex, turbulent, compressible flows at relatively low Reynolds numbers using a Direct Numerical Simulation (DNS) approach [28, 29]. Their results demonstrate that even when computing turbulent behavior through DNS, the use of a high-order method can significantly reduce the mesh refinement and time step restrictions. Collis has also explored a variational multiscale (VMS) method for performing Large Eddy Simulations (LES) in the context of a DG solver [27].

Peraire and Persson have developed a RANS solver that utilizes the one-equation Spallart-Alamaras closure model [46]. They have also introduced a high-resolution artificial viscosity model for shock-capturing [52].

More recently, Bassi *et al.* continue to expand the capabilities of their solver to include unsteady, incompressible flows [2] and shock-capturing [3], which is based on an extension of Peraire and Persson's artificial viscosity model.

Landmann *et al.* have also been investigating the use of DG for turbulence simulations [38], solving the RANS equations using the k-ω closure model of Bassi and Rebay.

2.2 Curved Elements

In their earliest work on the DG method, Bassi and Rebay found that using linear elements near curved boundaries would give rise to non-physical unsteady flow features in the domain [8]. They observed that the accuracy of the solution was significantly improved if the mesh elements near the boundary were curved to match the actual geometry. To maintain the optimal order of accuracy of the DG method, Bassi and Rebay determined, by numerical experimentation, that the reflecting boundary condition must be computed using curved elements with the same order shape functions as the order of the solution basis. They found no significant improvement in their error norms when utilizing higher-order elements.

In their 1973 treatise on the finite element method, Strang and Fix [63] cited the approximation of smoothly curving domain boundaries by piecewise linear or polynomial mesh elements among the so-called *variational crimes*. In instances where these approximations could not exactly reproduce the shape of the boundary geometry, certain underlying assumptions for Rayleigh-Ritz criteria could not be satisfied. The fact that it was even possible to obtain a solution on these meshes was attributed to the ability of approximation and interpolation spaces to keep the errors bounded.

In the Computational Fluid Dynamics (CFD) community, the topic of mesh quality has traditionally been associated with analysis done on structured meshes during the late 1970's and early 1980's, and unstructured meshes during the 1990's [65, 66]. Nearly all of these studies were concerned with the use of simply shaped mesh elements, particularly those of a restricted set of polygonal (triangles, quadrilaterals) and polyhedral shapes

(tetrahedra, pyramids, prisms, hexahedra). The development of more advanced element shapes has occurred primarily within the finite element community. The use of higher-order element shapes in finite element analysis has roughly paralleled the development of methods for mathematical descriptions of general free-form surfaces [32]. The terms *sub-parametric*, *iso-parametric*, and *super-parametric* came into use to describe mesh elements with less than, the same as, or greater than the order of the polynomial basis used to approximate the solution. Analysis of these elements determined that isoparametric elements were sufficient for retaining the optimal convergence rate in the energy norm for elliptic problems [39].

Landmann, Kessler, *et al.* [37, 38] recently conducted several additional numerical experiments using a high-order accurate discontinuous Galerkin solver to model flows ranging from purely inviscid to laminar and turbulent viscous flows in two- and three-dimensions. Their results confirm that the use of linear elements to approximate a curved boundary significantly degrades the accuracy of the solution and, in many cases, will introduce non-physical, unsteady flow features into the solution. The authors suggest that an appropriate boundary approximation be at least C^1 continuous at all points of the smoothly curving surface. The authors utilize cubic Hermite surface patches to accomplish the desired continuity. They also provide some techniques for generating these patches when only the positions of the mesh nodes are provided [42, 43].

For practical simulations involving turbulent, high Reynolds number flows, the thin boundary layers are typically resolved with highly anisotropic mesh elements. Shephard, Flaherty, *et al.* [55, 59] have considered curved anisotropic mesh elements in the context

of their hp-adaption algorithms for general finite element methods. They composed a metric for guiding the remeshing of the domain based on the Hessian of the solution. They also addressed the issue of maintaining the fidelity of the underlying geometry of the boundaries during the remeshing step. By interfacing their mesh generator with an internal representation generated from topology and geometry data extracted from the CAD model, they are able to accurately refine the mesh and create curved elements which conform to the boundaries.

The use of curved elements is one of two ways to reduce the errors generated from the boundary conditions. The other method involves modifying the boundary data applied to the mesh approximation to account for the displacement error. Recently, Krivodonova and Berger [35] published an alternative to using explicitly curved mesh elements at reflecting boundary conditions. Traditional linear elements are used in conjunction with the exact normal information from the boundary geometry. Unlike traditional reflecting boundary conditions where a ghost cell state is established and a Riemann problem solved to establish the interface flux, Krivodonova and Berger allow the normal component of the flow to leave the domain and instantaneously reappear at another quadrature point symmetric to the first. This method appears to be limited to two-dimensional problems and mesh discretizations where the quadrature points can be symmetrically placed on the boundary. For general three-dimensional surface patches the flow streamlines may not exactly align with the orientation of the quadrature points. Also, if the shape of the surface patch is not symmetric, then this approach may not properly account for the production and destruction of flux at the boundary.

Mahajan [45] recently conducted a more detailed study of the specification of the boundary condition data in the context of second order elliptic problems in one- and two-dimensions. In this work, the boundary data on the computational mesh was recovered by integrating the flux functions through the region between the domain boundary and the mesh boundary. Mahajan was able to maintain the expected order of accuracy for the solver as long as the displacement of the mesh boundary from the domain boundary was less than the size of the adjacent mesh element. Since the analysis was conducted on isotropic meshes in one-dimension, it is not immediately clear how this would extend to anisotropic meshes in multiple dimensions where the displacement of the physical boundary may be on the order of several cell widths. The cases in the study also involved only meshes which were completely embedded inside the physical domain; hence, treatment of convex boundary shapes was not specifically addressed.

CHAPTER 3

IMPLEMENTATION OF A DISCONTINUOUS GALERKIN SOLVER

To facilitate the investigation of mesh quality as it pertains to the Discontinuous Galerkin (DG) method, a numerical solver for the compressible Navier-Stokes equations has been implemented and subjected to numerous code verification tests. Some of these verification tests form the basis for the research methodology utilized in the present work. This chapter provides the implementation details for the DG solver and then briefly describes the verification tests, which are relevant to this work. The next chapter describes how these tests have been extended for the purpose of investigating mesh quality.

3.1 Governing Equations

The problem domains of interest to the present work are inviscid and high-Reynolds number fluid flows. The pertinent governing equations are the Euler and Navier-Stokes equations for inviscid and viscous flows, respectively. For the purposes of this research, only single-species, perfect gases are considered. The thermodynamic state of the fluid at every point in the domain is described by a set of $d + 2$ variables (where d is the number of spatial dimensions being considered). The conservative variables of mass density (ρ), momentum density ($\rho\vec{u}$), and total energy density (ρe) are used for this purpose. To

evolve these quantities in time, a system of coupled Partial Differential Equations (PDEs) is solved.

The Euler equations, (3.1), (3.2), and (3.3), are a mathematical description of the conservation laws of mass, momentum and energy as applied to the motion of a inviscid, compressible fluid continuum. These equations provide satisfactory results for a wide range of convection dominated flow problems in which the effects of diffusion processes are negligible. If we let $\vec{u} = u\hat{\imath} + v\hat{\jmath} + w\hat{k}$ be the fluid velocity, then the Euler equations may be expressed as:

$$\frac{\partial}{\partial t}(\rho) + \text{div}(\rho\vec{u}) = 0 \tag{3.1}$$

$$\frac{\partial}{\partial t}(\rho\vec{u}) + \text{div}(\rho\vec{u}\vec{u}) = -\text{div}(P\tilde{I}) \tag{3.2}$$

$$\frac{\partial}{\partial t}(\rho e) + \text{div}(\rho e\vec{u}) = -\text{div}(P\vec{u}) \tag{3.3}$$

For problems in which the viscous and thermal properties of the fluid cannot be neglected, the Navier-Stokes equations, (3.4), (3.5), and (3.6) must be utilized.

$$\frac{\partial}{\partial t}(\rho) + \text{div}(\rho\vec{u}) = 0 \tag{3.4}$$

$$\frac{\partial}{\partial t}(\rho\vec{u}) + \text{div}(\rho\vec{u}\vec{u}) = -\text{div}(P\tilde{I}) + \text{div}(\tilde{\sigma}) \tag{3.5}$$

$$\frac{\partial}{\partial t}(\rho e) + \text{div}(\rho e\vec{u}) = -\text{div}(P\vec{u}) + \text{div}(\tilde{\sigma}\cdot\vec{u}) + \text{div}(k\ \text{grad}(T)) \tag{3.6}$$

These systems are closed by providing additional expressions which relate the primitive variables of temperature (T) and pressure (P) to the conservative variables.

$$P(\rho, e_0) = (\gamma - 1)(\rho e_0) \tag{3.7}$$

$$T(e_0) = \frac{\rho e_0}{c_v} \tag{3.8}$$

where $\gamma = c_p/c_v$ is the adiabatic exponent, c_p and c_v are the specific heats at constant pressure and volume, respectively, $e_0 = e - 1/2(\vec{u} \cdot \vec{u})$ is the specific internal energy $\tilde{\sigma}$ is the shear stress tensor:

$$\tilde{\sigma} = 2\mu\tilde{D} + \lambda \text{div}(\vec{u})\tilde{I} \tag{3.9}$$

$$\tilde{D} = \frac{1}{2}\left(\text{grad}(\vec{u}) + (\text{grad}(\vec{u}))^T\right) \tag{3.10}$$

$$\lambda = \mu_b - \frac{2}{3}\mu \text{div}(\vec{u}) \tag{3.11}$$

$$\mu = \mu_s + \mu_t \tag{3.12}$$

where μ_s, μ_t, and μ_b are the coefficients of shear, turbulent, and bulk viscosity, respectively. The thermal conductivity k is given by:

$$k = k_s + k_t \tag{3.13}$$

$$k_s = \frac{c_p Pr}{\mu_s} \tag{3.14}$$

$$k_t = \frac{c_p Pr_t}{\mu_t} \tag{3.15}$$

The laminar and turbulent Prandtl numbers are assumed to be constant with $Pr = 0.72$ and $Pr_t = 0.92$.

The system as a whole can be written more compactly as:

$$\frac{\partial \mathbf{Q}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = \mathbf{S}. \tag{3.16}$$

$$\vec{\mathbf{F}} = \vec{\mathbf{F}}_c(\mathbf{Q}) - \vec{\mathbf{F}}_d(\mathbf{Q}, \vec{\nabla}\mathbf{Q}) \tag{3.17}$$

Here, $\mathbf{Q}$ denotes the array of conserved quantities (mass, momentum, and total energy densities), and $\mathbf{S}$ represents any additional (non-conservative) source terms. $\vec{\mathbf{F}}_c$ and $\vec{\mathbf{F}}_d$ are

arrays of vector valued convective and diffusive fluxes, respectively. When expressed in Cartesian component form, the convective fluxes are given by:

$$\vec{\mathbf{F}}_c = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + P & \rho uv & \rho uw \\ \rho vu & \rho v^2 + P & \rho vw \\ \rho wu & \rho wv & \rho w^2 + P \\ (\rho e + P)u & (\rho e + P)v & (\rho e + P)w \end{bmatrix} \begin{bmatrix} \hat{\imath} \\ \hat{\jmath} \\ \hat{k} \end{bmatrix} \tag{3.18}$$

and the diffusive fluxes by:

$$\vec{\mathbf{F}}_d = \begin{bmatrix} 0 & 0 & 0 \\ \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \\ s_x + k\,\partial_x T & s_y + k\,\partial_y T & s_z + k\,\partial_z T \end{bmatrix} \begin{bmatrix} \hat{\imath} \\ \hat{\jmath} \\ \hat{k} \end{bmatrix} \tag{3.19}$$

where $\vec{s} = \tilde{\sigma} \cdot \vec{u} = s_x\hat{\imath} + s_y\hat{\jmath} + s_z\hat{k}$.

3.2 Discontinuous Galerkin

The Discontinuous Galerkin (DG) method is derived from both the Finite Element (FE) and Finite Volume (FV) methods. Like the FE method, DG attains a formal high-order of accuracy through the use of piecewise polynomial functions, which represent the solution state in each cell over the domain. Unlike most FE methods, however, the DG method permits a spatially discontinuous representation of the solution. Within each cell, the solution state is approximated by a continuous polynomial, but at the interface

between cells there are no formal continuity requirements explicitly enforced. As a result, the solution approximation will typically be double valued at the cell interfaces.

In the DG method, the polynomial solution in each element is completely described by a set of basis functions and a set of coefficients, which are unique to the element. Thus, the Degrees of Freedom (DOFs) describing the solution are localized to each element. The DOFs in each element are coupled to the DOFs in adjacent elements through interface fluxes defined on shared faces. These inter-element fluxes must properly account for the discontinuities in the solution which will typically arise at shared element interfaces. The convective interface flux terms are computed using upwinded numerical flux formulations, which are nearly identical to those used in the FV method. The diffusive interface fluxes are computed using either an interior penalty method or recovery method.

Since the solution DOFs are localized to each element and are only weakly coupled to the DOFs in adjacent elements, the resulting discretized equations possess very compact stencils. It is the compactness of the scheme that endows the method with a number of desirable numeric and algorithmic properties. The method is well suited for parallelization for both explicit and implicit time integration techniques. The relaxed continuity constraints between adjacent elements allow for much greater freedom in domain discretization strategies and also enable a number of techniques for solution adaptation (h- and p-refinement) and convergence acceleration (e.g. multigrid) to be implemented in a fairly straight-forward manner. Since extended stencils are not required to construct the solution approximation, there is also no need for special treatments of elements adjacent to domain boundaries. Although the DG method will typically utilize more DOFs than a

comparable continuous FE method on a similar mesh, it is perhaps this extra redundancy of information that permits the considerable flexibility of the scheme.

The DG method utilizes the method of weighted residuals to project the residual of the numerical solution onto the polynomial approximation space. These projected residuals are then used to advance the degrees of freedom of the solution in time by explicit or implicit methods. In the weighted residual approach, the governing equations (3.16) are multiplied by a weighting (or test) function ψ and then integrated over the entire domain Ω.

$$\int_\Omega \psi \frac{\partial \mathbf{Q}}{\partial t}\, d\Omega + \int_\Omega \psi(\vec{\nabla} \cdot \vec{\mathbf{F}})\, d\Omega = \int_\Omega \psi \mathbf{S}\, d\Omega \tag{3.20}$$

When solving the unmodified Navier-Stokes equations, $\mathbf{S} \equiv 0$.

Integration by parts is used to split the volume integral of the flux terms into a combination of surface and a volume integrals. This results in the weak form of the governing equations (3.21).

$$\int_\Omega \psi \frac{\partial \mathbf{Q}}{\partial t}\, d\Omega = \int_\Omega (\vec{\nabla}\psi \cdot \vec{\mathbf{F}})\, d\Omega - \oint_{\partial\Omega} \psi(\vec{n} \cdot \vec{\mathbf{F}})\, d\Gamma + \int_\Omega \psi \mathbf{S}\, d\Omega \tag{3.21}$$

3.2.1 Domain Decomposition

In order to represent geometrically complex domains in a manner which is conducive to efficient numerical computations, a partition of space, $\mathcal{T}_h$, is introduced. The physical domain, Ω, is decomposed into a collection of non-overlapping, simply connected elements, Ω^e, such that:

$$\bigcup_{e \in \mathcal{T}_h} \Omega^e = \Omega_h \approx \Omega. \tag{3.22}$$

where Ω_h is the mesh approximation to the physical domain, Ω. The weak form of the governing equations (3.21) must be satisfied on each element:

$$\int_{\Omega^e} \psi \frac{\partial \mathbf{Q}}{\partial t}\, d\Omega = \int_{\Omega^e} (\vec{\nabla}\psi \cdot \vec{\mathbf{F}})\, d\Omega - \oint_{\partial\Omega^e} \psi(\vec{n} \cdot \vec{\mathbf{F}})\, d\Gamma + \int_{\Omega^e} \psi \mathbf{S}\, d\Omega \tag{3.23}$$

for all $\Omega^e \in \Omega_h$.

The manner in which the domain is spatially decomposed into a computational mesh of discrete elements depends upon the type of flow problem under consideration and the specifics of the geometry of the domain boundaries. Ultimately, the accuracy of the solution and the efficiency of the solver are highly dependent upon the mesh elements being appropriately sized and shaped to resolve the domain geometry and all relevant physical flow phenomena.

In the following sections we describe the generation of finite element meshes for use with the discontinuous Galerkin solver. The meshing procedure begins with the generation of a finite volume mesh to establish an initial partition of space. These meshes are then p-refined (i.e. the polynomial order of the mesh elements are increased) using additional geometry information from the domain boundaries. Two strategies are pursued for interpolating the mesh to the boundary surfaces: one based on Lagrange interpolation, and the other based on Hermite interpolation.

3.2.1.1 Bezier Elements

In the current solver implementation, the shapes of the mesh elements are represented internally as Bezier volumes. The Bezier representation was selected for its well understood numerical properties, particularly the robust and efficient evaluation of positions and

derivatives within the element. Using the Bezier volume description, the shape of each element may be specified independently of all other elements, yet still be able to conform, in a natural way, to shared geometric constraints such as faces shared between adjacent elements. If greater continuity is required, Bezier elements can also be easily incorporated into larger B-Spline entities.

Figure 3.1 shows a series of computational meshes generated with linear, quadratic, and cubic mesh elements.

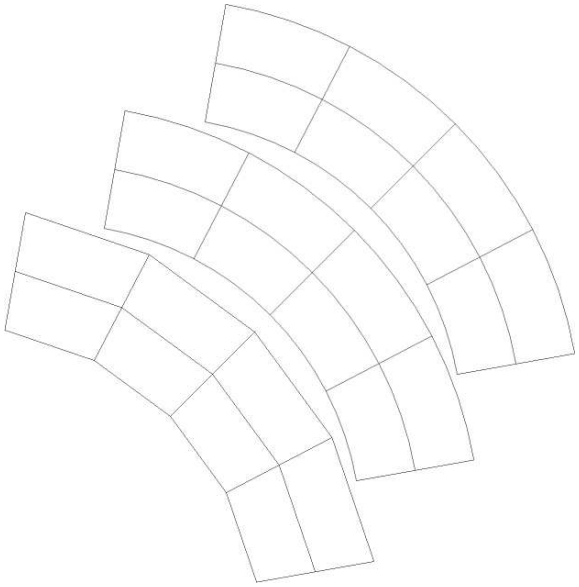

Figure 3.1

Sequence of p-refined meshes.

The Bezier representation consists of a set of basis functions and a corresponding set of control points. The basis functions are the Bernstein polynomials:

$$B_i^p(\xi) = \frac{1}{2^p}\frac{p!}{i!(p-i)!}(1-\xi)^{(p-i)}(1+\xi)^i \tag{3.24}$$

and tensor products thereof

$$B^p_{ij}(\xi,\eta) = B^p_i(\xi)B^p_j(\eta) \tag{3.25}$$

$$B^p_{ijk}(\xi,\eta,\zeta) = B^p_i(\xi)B^p_j(\eta)B^p_k(\zeta) \tag{3.26}$$

where p is the degree of the polynomial and $i,j,k \in [0,p]$. For greater flexibility, p may be chosen independently for each parametric direction:

$$B_{ijk}(\xi,\eta,\zeta) = B^{p_i}_i(\xi)B^{p_j}_j(\eta)B^{p_k}_k(\zeta) \tag{3.27}$$

for $i \in [0,p_i]$, $j \in [0,p_j]$, and $k \in [0,p_k]$.

The set of control points – also known as the control net – consists of $(p+1)^3$, or $(p_i+1)\times(p_j+1)\times(p_k+1)$, points in space. The positions of these points determine the shape of the entity. Figure 3.2 displays the representative control nets for linear, quadratic, and cubic approximations to an annular arc segment. Since the shape is linear in the radial direction, only two control points were required in that direction.

Bezier curves interpolate to the first and last control points, while surfaces and volumes tend to interpolate to the control net at their corners. This is a particularly useful property for the current application as it allows us to retain the original point spacing specified by a more traditional input mesh. Assuming that something like a typical finite volume mesh is provided, its nodes can be used as the corner nodes of the individual element control nets, thereby preserving the topology and spatial distribution of the original mesh.

For the purposes of curved mesh generation, Bezier allows a variable order representation that can easily be made to conform to domain boundaries while also maintaining suitable element shapes away from the wall. If necessary, the curvature of the boundary

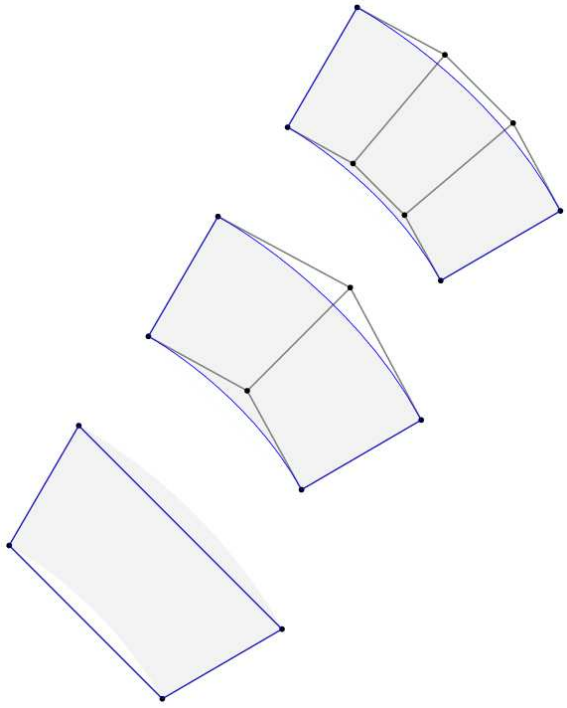

Figure 3.2

Linear, quadratic, and cubic control nets for an isotropic curved element.

can be propagated further into the domain by simply adjusting the positions of the interior control points until the desired element shapes are obtained.

In the case of high aspect ratio cells, for example, the curvature of the boundary may exceed the height of the element adjacent to the boundary. If only the boundary face is curved, then the overall shape of the resulting element is likely to be unacceptable for CFD simulations. In Figure 3.3 the shape of the elements on the right will most likely give rise to negative and singular determinants of the Jacobian of the mapping, whereas the elements on the left side of the figure will typically possess more favorable derivative behavior.

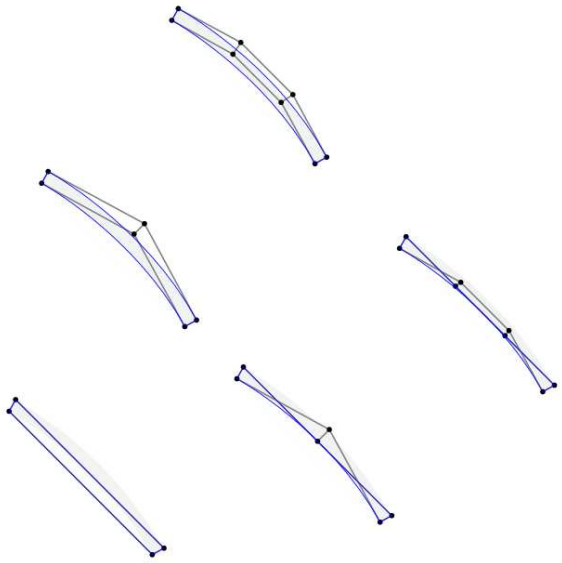

Figure 3.3

Linear, quadratic, and cubic control nets for an anisotropic curved element.

To avoid the generation of negative and singular volumes, it should be sufficient to ensure that there are no lines crossing in the control mesh. The process for smoothing and removing kinks from the mesh then becomes quite similar to that practiced in structured mesh generation for achieving the same purpose [65].

3.2.1.2 Lagrange Interpolation

For the Lagrange interpolation, a set of uniformly spaced points are evaluated on the boundary surface. A set of Bezier control points is then generated such that the reconstructed surface passes through each of the given points. The positions of the control points may be computed directly by solving the following system of equations:

$$S(\xi_i, \eta_j) = \vec{x}_{ij} \quad \forall i \in [0, p_i] \text{ and } j \in [0, p_j] \tag{3.28}$$

The surface definition $S(\xi, \eta)$ is a linear combination of the basis functions and control points, so the following linear system may be solved:

$$\sum_{m=0}^{p_i} \sum_{n=0}^{p_j} B_m^{p_i}(\xi_i) B_n^{p_j}(\eta_j) \vec{b}_{mn} = \vec{x}_{ij} \quad \forall i \in [0, p_i] \text{ and } j \in [0, p_j] \tag{3.29}$$

As previously mentioned, the corner points ($\vec{x}_{00}$, $\vec{x}_{0p}$, $\vec{x}_{p0}$, $\vec{x}_{pp}$) are taken to be the mesh nodes supplied to the solver from the input finite volume mesh. The parametric coordinates (ξ_i and η_j) are uniformly distributed in each parametric direction in the range $[-1, 1]$ including the endpoints. Intermediate points are generated from a bilinear interpolation of the four corner nodes.

$$\hat{x}_{ij} = \frac{1}{4} \quad (\quad (1 - \xi_i)(1 - \eta_j)\vec{x}_{00} + (1 + \xi_i)(1 - \eta_j)\vec{x}_{p0} + \tag{3.30}$$

$$(1 - \xi_i)(1 + \eta_j)\vec{x}_{0p} + (1 + \xi_i)(1 + \eta_j)\vec{x}_{pp} \;) \tag{3.31}$$

These points are then projected onto the boundary surface using an iterative method which minimizes the projected distance. This processes then yields the interpolation points, $\vec{x}_{ij}$.

In the present work, sets of self-similar grids generated from the Lagrange interpolation method will be designated as Cn, where the n represents the number of sample points taken in each direction (e.g. C2, C3, C4, etc.). Since the original nodes from the finite volume mesh are assumed to be in all sample sets as the corner points for each element, then the lowest interpolation order considered is second-order (C2). For the second-order interpolation, the original mesh nodes are the only points in the control net.

3.2.1.3 Hermite Interpolation

With the Hermite interpolation, both the location of the sample points, $\vec{x}_{ij}$, and their surface tangent directions, $(\vec{x}_{ij})_\xi$ and $(\vec{x}_{ij})_\eta$, are taken as input. The nodal locations and the tangent vectors are then used to generate a cubic Bezier control net. For smooth boundary surfaces, the mesh elements which result from this interpolation procedure have the property that the mapping functions for the adjacent elements are C^1-continuous in the direction normal to the shared face. Thus, for this interpolation strategy, there are no slope discontinuities at the element interfaces on the boundaries unless one already exists in the description of the boundary surface.

The system of equations to be solved arises from the conditions that must be satisfied at each of the sample points: namely, the positions, the tangential derivatives, and their cross-products.

$$S(\xi_i, \eta_j) \quad = \quad \vec{x}_{ij} \tag{3.32}$$

$$\partial_\xi S(\xi_i, \eta_j) = (\vec{x}_{ij})_\xi \tag{3.33}$$

$$\partial_\eta S(\xi_i, \eta_j) = (\vec{x}_{ij})_\eta \tag{3.34}$$

$$\partial^2_{\xi\eta} S(\xi_i, \eta_j) = (\vec{x}_{ij})_\xi \times (\vec{x}_{ij})_\eta \tag{3.35}$$

As with the Lagrange interpolation method, the system may be expressed as a linear system and the control points obtained from its solution.

This method will be used to generate mesh elements of fourth and sixth-order. For the sixth-order element, the extra sample points are generated at the center of the mesh edges and the mesh face and projected onto the exact geometry definition. The families of self-similar meshes generated with this method will be designated as Hn, where n denotes the number of interpolated points in each parametric direction (e.g. H2, H3).

3.2.2 Approximation of the Solution State

As with the finite element method, the solution state $\mathbf{Q}$ is represented by a set of piecewise polynomial functions. Unlike most finite element schemes, however, the DG method does not enforce any continuity constraints on the solution at the interface between adjacent elements. The result is that the polynomial solution representation within each element is completely independent of the solution representation in all other elements.

The number of linearly independent basis functions of degree p or less, in d independent variables required to span the space of p^{th}-degree polynomials, $\mathcal{P}^p$, is given by:

$$N(p, d) = \begin{pmatrix} p+d \\ \\ p \end{pmatrix} = \frac{(p+d)!}{(d!)(p!)} \tag{3.36}$$

The discontinuous solution approximation space for the DG method is obtained by selecting a set of N linearly independent basis functions for each element. These basis functions are defined such that they are non-zero only within their associated element, and zero everywhere else.

$$\Phi^e \equiv \{\phi^e \in \mathcal{P}^p \mid \alpha_i \phi_i^e \neq 0,\ i \in [1, N] \ \forall\ \alpha_i \neq 0\} \tag{3.37}$$

The polynomial approximation to the solution within an element is then a linear combination of the basis functions defined on that element[1].

$$\mathbf{Q}^e = \mathbf{q}_i^e \phi_i^e \tag{3.38}$$

3.2.2.1 Spectral Decomposition

For any given function of space $f(x, y, z)$, a piecewise polynomial approximation can be obtained over a region Ω by projecting the function into the polynomial space.

$$\int_\Omega f_h \phi_j \ d\Omega = \int_\Omega f \phi_j \ d\Omega \qquad \forall \phi_j \in \Phi \tag{3.39}$$

Here, $f_h = f_i \phi_i$ is the desired polynomial approximation. The polynomial coefficients, f_i, are obtained by solving the linear system:

$$\left[\int_\Omega \phi_i \phi_j \ d\Omega\right] [f_j] = \left[\int_\Omega f \phi_j \ d\Omega\right] \tag{3.40}$$

$$\mathbf{M}\mathbf{f} = \mathbf{r} \tag{3.41}$$

where

$$\mathbf{M} = \left[\int_\Omega \phi_i \phi_j \ d\Omega\right] \tag{3.42}$$

is the mass matrix, $\mathbf{f} = [f_i]$ is the vector of unknown polynomial coefficients.

[1]Here, and in the following sections, we use standard summation convention where repeated indices imply summation. The range of the summation should be obvious from the context.

3.2.2.2 Choice of Basis Functions

In theory, any set of linearly independent basis functions which span $\mathcal{P}^p$ will form a sufficient basis for obtaining $(p+1)^{th}$-order accurate solution approximations. In practice, the overall numerical stability and computational efficiency of the solver are almost always improved by selecting a set of orthogonal basis functions, which will produce diagonal mass matrices. The use of orthonormal basis functions is typically the most efficient. Since the associated mass matrix is the identity matrix, it does not need to be explicitly inverted and can be factored out of certain computations. In the current solver implementation, there are two sets of basis functions available: power series and orthonormal.

The power series basis functions are a set of monomials with the form:

$$\phi(\xi, \eta, \zeta) = \xi^i \eta^j \zeta^k \tag{3.43}$$

for all permutations of $i, j, k \in [0, p]$ such that $i + j + k \leq p$. The orthonormal basis set is obtained from the power series through the use of a Gram-Schmidt process [71].

Returning to the discretized governing equations (3.21), the Galerkin finite element method specifies that the test functions ψ are selected from the same set of basis functions used to describe the solution.

$$\int_{\Omega^e} \phi_i^e \frac{\partial \mathbf{Q}^e}{\partial t} \, d\Omega = \int_{\Omega^e} (\vec{\nabla} \phi_i^e \cdot \vec{\mathbf{F}}) \, d\Omega - \oint_{\partial\Omega^e} \phi_i^e (\vec{n} \cdot \vec{\mathbf{F}}) \, d\Gamma + \int_{\Omega^e} \phi_i^e \mathbf{S} \, d\Omega \tag{3.44}$$

for all ϕ_i and all elements $\Omega^e \in \Omega_h$.

The surface integrals in the above equation are evaluated at all bounding faces of each element. Due to the discontinuous nature of the solution approximation, the values of

the flux functions (and basis functions) in the integrands are not uniquely defined. The discontinuities are resolved using numerical flux formulations denoted by $\mathbf{H} \approx \vec{\mathbf{F}} \cdot \vec{n}$.

$$\int_{\Omega^e} \phi_i^e \frac{\partial \mathbf{Q}^e}{\partial t} \, d\Omega = \int_{\Omega^e} (\vec{\nabla} \phi_i^e \cdot \vec{\mathbf{F}}) \, d\Omega - \oint_{\partial\Omega^e} \phi_i^e \mathbf{H} \, d\Gamma + \int_{\Omega^e} \phi_i^e \mathbf{S} \, d\Omega \tag{3.45}$$

Due to the fundamental differences in numerical behavior of the convective and diffusive flux terms, the interface fluxes for each must be treated separately.

3.2.3 Convective Fluxes

For the convective terms, the numerical fluxes at the element interfaces and boundary faces are computed using an upwinded flux formulation similar to those developed for the finite volume method [67]. The current solver implementation includes Roe's approximate Riemann solver, van Leer's flux vector splitting method and the local Lax-Friedrichs flux difference splitting scheme. For the numerical experiments conducted for this report, the Roe scheme has been used to obtain the convective interface fluxes.

3.2.4 Diffusive Fluxes

The diffusive fluxes must be treated carefully: not only are the solution values discontinuous at the interface, but the gradients of solution are as well. In the literature there have been numerous formulations for the diffusive fluxes provided. Some of these methods include: the local discontinuous Galerkin method (LDG) of Cockburn and Shu [25], the interior penalty method of Oden and Baumann [11, 48], and the lifting operators of Bassi and Rebay (BR2) [9]. More recently, Peraire and Persson [51] have derived an update to the LDG method with a more compact stencil which they have appropriately named

the compact discontinuous Galerkin method (CDG). Also, van Leer *et al.* [68, 69] have recently formulated a diffusive operator based on a locally recovered solution approximation. The recovered solution, which smoothly interpolates the solution between two adjacent elements, is then used to compute the diffusive fluxes at the interface.

The local lifting operator formulation of Bassi and Rebay (BR2) was selected for use in the current solver implementation due to its compactness as well as the fact that it has already been shown to work in multiple dimensions and on complex flow problems, including those with highly anisotropic curved elements.

In the BR2 scheme, a local lifting operator $\vec{\mathbf{r}}_f^e$ is computed on each face for each of the conserved quantities:

$$\int_{\Omega^e} \phi^e \vec{\mathbf{r}}_f^e \, d\Omega = \int_{\Gamma_f} (\mathbf{Q}_0 - \mathbf{Q}^e) \phi^e \vec{n} \, d\Gamma \tag{3.46}$$

as well as a global lifting operator $\mathbf{R}$ for each element:

$$\int_{\Omega^e} \phi^e \vec{\mathbf{R}}^e \, d\Omega = \oint_{\partial\Omega^e} (\mathbf{Q}_0 - \mathbf{Q}^e) \phi^e \vec{n} \, d\Gamma = \sum_f \vec{\mathbf{r}}_f^e \tag{3.47}$$

where $\mathbf{Q}^e$ is the solution state in element e, and $\mathbf{Q}_0$ is the target solution value that could be obtained on the face if the local lifting operator were applied to the gradient of the solution within the element. Thus, the local lifting operator serves as a local correction to the gradient designed to bring the solution states in adjacent elements together at a common value $\mathbf{Q}_0$. On interior faces, the solution state in the current element, $\mathbf{Q}^e$, is extrapolated to the face and is averaged with the value from the adjacent element, $\mathbf{Q}^*$, extrapolated to the same location to obtain: $\mathbf{Q}_0 = (\mathbf{Q}^e + \mathbf{Q}^*)/2$. On boundary faces with

Dirichlet conditions this value is set to the prescribed boundary value $\mathbf{Q}_0 = \mathbf{Q}_b$, and on boundaries with Neumann conditions $\mathbf{Q}_0 = \mathbf{Q}^e$.

These two lifting operators are then utilized as a correction to the gradient for the purpose of computing the diffusive fluxes. It should be noted that the lifting operators are only applied to the gradients and are not utilized to recompute the solution values at the interface.

For each element, the volume integration of the diffusive fluxes is given by:

$$\int_{\Omega^e} \vec{\nabla}\phi_i \cdot \mathbf{F}_d(\mathbf{Q}^e, \vec{\nabla}\mathbf{Q}^e + \vec{\mathbf{R}}^e)\ d\Omega \tag{3.48}$$

and the surface integrations by:

$$\sum_{\Gamma^f \in \partial\Omega^e} \int_{\Gamma^f} \phi \mathbf{H}_d^f\ d\Gamma \tag{3.49}$$

where

$$\mathbf{H}_d^f = \frac{1}{2}\left(\vec{\mathbf{F}}_d(\mathbf{Q}^e, \vec{\nabla}\mathbf{Q}^e + \vec{\mathbf{r}}_f^e) + \vec{\mathbf{F}}_d(\mathbf{Q}^*, \vec{\nabla}\mathbf{Q}^* + \vec{\mathbf{r}}_f^-)\right) \cdot \vec{n} \tag{3.50}$$

3.2.5 Boundary Conditions

The discontinuous Galerkin method allows quite natural treatment of most boundary conditions by simply supplying the prescribed (or derived) boundary values for the solution $\mathbf{Q}_b$ (for Dirichlet conditions) or the normal flux $\vec{\mathbf{F}}_b \cdot \vec{n}$ (for Neumann conditions).

For Dirichlet boundaries, the gradients of the solution at the boundary are taken to be the same as that of the solution in the adjacent cell ($\vec{\nabla}\mathbf{Q}^b = \vec{\nabla}\mathbf{Q}^e$), and for Neumann boundaries, the solution values are taken from the interior state ($\mathbf{Q}^b = \mathbf{Q}^e$). Mixed conditions may be specified through the use of appropriate combinations of these quantities. The

prescribed values are applied at each quadrature point on the boundary faces where they may be subject to additional constraints depending on the types of boundary conditions imposed.

Solid-wall boundaries enforce a zero mass flux condition. Inviscid, slip-wall conditions (i.e. reflecting) prescribe a zero energy flux and a momentum flux based only on the static pressure directed normal to the wall. Viscous, no-slip boundaries impose a fluid velocity which matches the wall velocity, however, in this work we consider only cases with static geometry. Adiabatic and prescribed temperature conditions are also supported.

For flow-through boundaries, a characteristic decomposition is applied to the conditions across the boundary faces. Once the boundary values are obtained through a characteristic extrapolation procedure [34], they are subsequently used to compute the flux through the boundary using the numerical interface flux treatments described in Sections 3.2.3 and 3.2.4.

3.2.6 Time Integration

Once the residual source terms have been obtained from the discretized governing equations 3.45, they may be used to update the polynomial solution coefficients by using explicit or implicit time integration methods.

$$\frac{\partial \mathbf{Q}_i}{\partial t} = \mathbf{R}(\mathbf{Q}) \tag{3.51}$$

where

$$\mathbf{R} = \mathbf{M}^{-1}\left[-\oint_{\partial\Omega^e} \phi_i \mathbf{H}\, d\Gamma + \int_{\Omega^e} \vec{\nabla}\phi_i \cdot \vec{\mathbf{F}}\, d\Omega\right] \tag{3.52}$$

where

$$\vec{\mathbf{F}} = \vec{\mathbf{F}}_c(\mathbf{Q}^e) - \vec{\mathbf{F}}_d(\mathbf{Q}^e, \vec{\nabla}\mathbf{Q}^e) \tag{3.53}$$

and

$$\mathbf{H} = \mathbf{H}_c(\mathbf{Q}^e, \mathbf{Q}^*; \vec{n}) - \mathbf{H}_d(\mathbf{Q}^e, \vec{\nabla}\mathbf{Q}^e, \mathbf{Q}^*, \vec{\nabla}\mathbf{Q}^*; \vec{n}) \tag{3.54}$$

and $\mathbf{M}$ is the mass matrix (3.42). Since the basis functions are non-zero only in their prescribed elements, the mass matrix is block diagonal.

3.2.6.1 Explicit Methods

Cockburn and Shu have developed second and third-order accurate explicit Runge-Kutta algorithms (RK2, RK3) which are total variation bounded (TVB) [23]. Algorithm 3.1 describes the process for a k^{th}-order accurate integration.

Algorithm 3.1 (TVB Runge-Kutta integration)

Set $\mathbf{Q}^{(0)} = \mathbf{Q}(t_n)$

for $i = 1..k$

$$\mathbf{Q}^{(i)} = \sum_{l=0}^{i-1} \left(\alpha_{il} \mathbf{Q}^{(l)} + \Delta t_n \beta_{il} \mathbf{R}(\mathbf{Q}^{(l)}) \right)$$

$\mathbf{Q}(t_{n+1}) = \mathbf{Q}^{(k)}$

Here $\Delta t_n = t_{n+1} - t_n$ and the α and β coefficients are given in Table 3.1.

Table 3.1

Coefficients for the RK2 and RK3 schemes

k	α_{il}			β_{il}		
2	1			1		
	$\frac{1}{2}$	$\frac{1}{2}$		0	$\frac{1}{2}$	
3	1			1		
	$\frac{3}{4}$	$\frac{1}{4}$		0	$\frac{1}{4}$	
	$\frac{1}{3}$	0	$\frac{2}{3}$	0	0	$\frac{2}{3}$

A standard fourth-order Runge-Kutta (RK4) scheme has also been implemented [53].

$$\mathbf{k}_1 = \Delta t_n \mathbf{R}(\mathbf{Q}(t_n)) \tag{3.55}$$

$$\mathbf{k}_2 = \Delta t_n \mathbf{R}(\mathbf{Q}(t_n) + \frac{1}{2}\mathbf{k}_1) \tag{3.56}$$

$$\mathbf{k}_3 = \Delta t_n \mathbf{R}(\mathbf{Q}(t_n) + \frac{1}{2}\mathbf{k}_2) \tag{3.57}$$

$$\mathbf{k}_4 = \Delta t_n \mathbf{R}(\mathbf{Q}(t_n) + \mathbf{k}_3) \tag{3.58}$$

$$\mathbf{Q}(t_{n+1}) = \mathbf{Q}(t_n) + \frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4 \tag{3.59}$$

3.2.6.2 Implicit Methods

Equation (3.51), can be discretized in time as:

$$(1+\psi)\Delta\mathbf{Q}^n - \psi\Delta\mathbf{Q}^{n-1} = \Delta t\left[(1-\theta)\mathbf{R}^n(\mathbf{Q}^n) + \theta\mathbf{R}^{n+1}(\mathbf{Q}^{n+1})\right] \tag{3.60}$$

where $\Delta\mathbf{Q}^n = \mathbf{Q}^{n+1} - \mathbf{Q}^n$. Equation (3.60) enables the use of a family of time integration methods including the three point backward ($\theta = 1$, $\psi = \frac{1}{2}$), backward Euler ($\theta = 1$, $\psi = 0$), and Crank-Nicholson ($\theta = \frac{1}{2}$, $\psi = 0$) schemes [13, 44].

Due to the non-linearity of $\mathbf{R}$, updates to the solution are obtained from Equation (3.60) through the use of an iterative Newton method. The Newton method converges on the solution to $\mathcal{L}(\mathbf{Q}) = 0$ by iteratively solving:

$$\mathcal{L}'(\mathbf{Q}^n)\ \Delta\mathbf{Q}^{n,p} = -\mathcal{L}(\mathbf{Q}^{n,p}) \tag{3.61}$$

where

$$\mathcal{L}(\mathbf{Q}^{n+1}) = \mathbf{Q}^{n+1} - \mathbf{Q}^n - \frac{\Delta t}{1+\psi}[(1-\theta)\mathbf{R}(\mathbf{Q}^n) + \theta\mathbf{R}(\mathbf{Q}^{n+1})] - \frac{\psi}{1+\psi}(\mathbf{Q}^n - \mathbf{Q}^{n-1}) \tag{3.62}$$

The iteration is initialized with the current solution state $\mathbf{Q}^{n,p=0} = \mathbf{Q}^n$ and is advanced by $\mathbf{Q}^{n,p+1} = \mathbf{Q}^n + \Delta\mathbf{Q}^{n,p}$ until the norm of the right side of Equation (3.61) falls below a prescribed tolerance, or a maximum number of iterations is exceeded. The Jacobian of Equation (3.62) is given by:

$$\mathcal{L}'(\mathbf{Q}^n) = \mathbf{I} - \frac{\theta\Delta t}{1+\psi}\left[\frac{\partial}{\partial\mathbf{Q}}\mathbf{R}(\mathbf{Q}^n)\right] \tag{3.63}$$

3.2.7 Matrix Scaling

The ability of the iterative Newton method to rapidly converge on a solution to the implicit system given above may be adversely affected by poor matrix scaling. It is typically

advantageous to eliminate, to as much an extent as possible, the effects of poor matrix scaling prior to solving the system. The solver used in the present work makes use of the row-scaling, column-scaling, and row-column equilibration algorithms. In each case, the solver operates on a modified system matrix and right-hand-side.

$$(\mathbf{D}_1\mathbf{A}\mathbf{D}_2)\mathbf{y} = \mathbf{D}_1\mathbf{b} \tag{3.64}$$

where $\mathbf{D}_1$ and $\mathbf{D}_2$ are diagonal matrices which scale equations and unknowns, respectively, and $\mathbf{x} = \mathbf{D}_2\mathbf{y}$ is the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$. For row-scaling, $\mathbf{D}_2$ is taken to be the identity, while for column-scaling $\mathbf{D}_1$ is the identity.

When they are not the identity matrix, the scaling factors appearing on the diagonals of $\mathbf{D}_1$ and $\mathbf{D}_2$ are computed using the iterative algorithm of Ruiz [57]. In this algorithm, the maximum value in each row and column are determined. Each row and column are then multiplied by scaling factors which are the inverse of the square root of the maximum values (i.e. $(||\mathbf{r}_i||_\infty)^{-1/2}$, and $(||\mathbf{c}_j||_\infty)^{-1/2}$). This continues until the maximum values in each row and column converge within some prescribed tolerance of one. The resulting entries in the diagonal matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ consist of the product of the scaling factors from each iteration for each row and column, respectively.

Algorithm 3.2 (Iterative row-column equilibration)

Let $\hat{\mathbf{A}}$ be the matrix scaled by $\mathbf{D}_1$ and $\mathbf{D}_2$:
$\hat{\mathbf{A}}^{(0)} = \mathbf{A},\ \mathbf{D}_1^{(0)} = \mathbf{I},$ and $\mathbf{D}_2^{(0)} = \mathbf{I},$
for $k = 0, 1, 2, ...$, until convergence

$$\mathbf{D}_R = \text{diag}\left(\sqrt{||\mathbf{r}_i^{(k)}||_\infty}\right), \text{ and } \mathbf{D}_C = \text{diag}\left(\sqrt{||\mathbf{c}_j^{(k)}||_\infty}\right),$$
$$\hat{\mathbf{A}}^{(k+1)} = \mathbf{D}_R^{-1}\hat{\mathbf{A}}^{(k)}\mathbf{D}_C^{-1}$$
$$\mathbf{D}_1^{(k+1)} = \mathbf{D}_1^{(k)}\mathbf{D}_R^{-1}, \text{ and } \mathbf{D}_2^{(k+1)} = \mathbf{D}_2^{(k)}\mathbf{D}_C^{-1},$$

3.3 Spatial Integration and Meshing Considerations

The accurate spatial integration of the governing equations is essential to a proper implementation of the discontinuous Galerkin method. For most problems of engineering interest, two discrete approximations are commonly utilized to enable the efficient solution of the governing equations on complex domains: domain decomposition and numerical quadrature.

For the discontinuous Galerkin method, each element acts as a subdomain all to itself. Within each element, the solution is approximated with a polynomial function which attempts to satisfy the weak form of the governing equations (3.45) subject to the boundary conditions imposed on the element by the solutions in the adjacent elements. For a practical mesh, the elements must be appropriately distributed such that the resulting flow field can be adequately resolved by the element-wise polynomials and the numerical integration of the governing equations can be carried out with the required numerical accuracy.

The computational mesh is a partition of space consisting of a collection of simply shaped elements, which should ideally completely cover the domain with no gaps and no overlaps. However, depending on the specific geometry of the domain and the sizes and shapes of available mesh elements, the resulting mesh is often only an approximation to the exact domain.

In the present work, Bezier volumes are utilized as the internal representation of the mesh elements. Bezier elements offer a consistent representation of volume and surface entities which can easily be made to conform to prescribed boundary surfaces to within a desired order of accuracy. More details are provided in Section 3.2.1.1.

It is often not practical to evaluate the integrals in the governing equations (3.45) exactly. It is, therefore, common practice to evaluate these integrals using numerical quadrature rules with the appropriate order of accuracy. [2] Numerical quadrature rules approximate the continuous integration operation with a weighted sum of integrand evaluations.

$$\int f(x)\ dx \approx \sum_q f(x_q) w_q \tag{3.65}$$

Here the integrand $f(x)$ is evaluated at a finite number of locations given by x_q, and w_q is a precomputed weight associated with each quadrature point. The weighted sum of these evaluations is the desired approximation of the integral. The use of more quadrature points will usually result in a more accurate approximation. Each quadrature rule has an associated order which determines the rate at which the numerical error in the integration is reduced as the size of the elements are reduced.

In order to maintain the desired accuracy and stability of the discontinuous Galerkin method, the order of accuracy of the quadrature rule must be at least twice that of the polynomial approximation of the solution [61].

Quadrature rules have long been available for a number of common element shapes, and a comprehensive encyclopedia of quadrature and cubature formulas has recently been compiled by Cools [30]. The most commonly used element shapes are those for which the integration by quadrature rules are readily available. Typically, the quadrilateral/hexahedral and triangular/tetrahedral element shapes are selected because the quadrature rules for these elements are most easily obtained and are fairly trivial to implement. When these

[2] The term cubature is sometimes utilized when referring to spatial integration in three or more dimensions. However, it is generally accepted that numerical quadrature can be used, in a generic sense, to describe the procedure for approximating a continuous integral, of any dimension, with a weighted sum.

two element types are used together to discretize the same domain, then prism and pyramid elements may also be required to ensure a topologically valid mesh.

The quadrature rules specify a collection of points in a standard parametric domain and a set of weights, one for each point. For linear, quadrilateral, and hexahedral elements, the parametric domains are typically in the range $[-1, 1]$ in each coordinate direction. For triangular and tetrahedral elements, the points are either provided in Barycentric coordinates (i.e. a linear combination of of the vertexes) with parameters in the range $[0, 1]$, or the triangle (tetrahedra) is mapped to a degenerate quadrilateral (hexahedra). In the latter case, care must be taken to avoid evaluating the integrand at, or very near to, the elements' singularities.

To facilitate the evaluation of the integrals on curved mesh elements, a mapping from parametric to global coordinates is established. The determinant of the Jacobian of this mapping then becomes part of the integrand.

$$\int_{\Omega} f(x, y, z)\, dx\, dy\, dz = \int_{\Omega_h} \hat{f}(\xi, \eta, \zeta)\, J^{-1}\, d\xi\, d\eta\, d\zeta$$

where

$$\hat{f}(\xi, \eta, \zeta) = f(x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta))$$

and

$$\mathbf{J}^{-1} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \qquad J^{-1} = \det(\mathbf{J}^{-1})$$

The quantity $d\Omega_h = J^{-1}\, d\xi\, d\eta\, d\zeta$ can be interpreted as a local measure of volume. If J^{-1} is constant, then it may be factored out of the integral. The result of the integration

in the parametric space is then simply multiplied by the volume of the element. This is typically what is done in finite volume solvers which rely solely on midpoint rules (one point quadrature) for integration.

Unfortunately, the constant Jacobian assumption is really only valid for simplex element shapes. For non-simplex elements, a constant Jacobian constraint severely limits the allowable element shapes. For example, a hexahedral element with a constant Jacobian is a right parallel-piped (i.e. a cube stretched along its principal axes). Any shearing or twisting of the element caused by non-planar faces or non-orthogonal grid lines would result in a non-constant Jacobian.

If the Jacobian of the mapping is not constant, then its determinant must be evaluated, along with the rest of the integrand, at each quadrature point. Inclusion of the determinant of the Jacobian of the inverse mapping allows the integral to be evaluated in the parametric space using the aforementioned quadrature rules.

$$\int_{\Omega_h} \hat{f}\ d\Omega_h = \int_{\Omega_h} \hat{f}(\xi,\eta,\zeta)\ J^{-1}\ d\xi\ d\eta\ d\zeta \approx \sum_q \hat{f}(\xi_q,\eta_q,\zeta_q)\ J_q^{-1} w_q$$

If the mesh is static, then the determinant of the Jacobian can be precomputed and combined with the quadrature weight to form a set of weighted volume elements. The quadrature rule is still a weighted sum, but now the weights also account for variations in the shape of the element as well.

3.4 Code Verification

Code verification is the process whereby a specific software system is verified to be properly implemented. A flow solver is an instantiation in code of a set of numerical al-

gorithms intended to solve the system of governing equations. As such, verification of the flow solver requires that the implementation of the numerical algorithms be demonstrated to be accurate to within acceptable tolerances. Upper bounds on the acceptable tolerances are typically established by the theoretical accuracy of the solver with proper consideration given to the floating point accuracy limitations of the computational hardware on which the solver is to be run.

Proper verification of the software implementation of each of the numerical algorithms is essential to establishing the reliability of the flow solver as a whole. In some instances, however, it may be impractical to test individual algorithms in isolation. In such cases, it may be more expedient, and yet still sufficient, to establish the proper functioning of the flow solver as a whole. The methods of exact, nearby, and manufactured solutions offer an economical way to test the solver full-up while still being able to detect to the smallest of implementation mistakes.

3.4.1 Method of Exact Solutions

The Method of Exact Solutions (MES) is employed when there exist known analytic solutions to the governing equations. With the exact solution readily available, the true error in the numerical solution can be computed directly at every point in the domain. If the method is consistent and convergent, one would expect the magnitude of these errors to diminish as the mesh is refined. The observed rate by which the errors converge should be of order $\mathcal{O}(h^{p+1})$, where p is the degree of the polynomials used to approximate the so-

lution. The observed rate of convergence is determined by comparing the rate of reduction in error to the rate of reduction in mesh spacing.

$$\mathcal{O} = \frac{\ln\left(\frac{||\mathcal{E}_i||}{||\mathcal{E}_j||}\right)}{\ln\left(\frac{h_i}{h_j}\right)} \tag{3.66}$$

where $||\mathcal{E}_n||$ is the norm of the integrated error in the numerical solution at the n^{th} refinement level and h_n is proportional to the mesh spacing at the n^{th} level. Failure to converge at the expected rate would indicate either a mistake in the implementation or an improper choice of algorithm(s) for discretizing the problem and/or computing its solution.

The principal drawback of the method of exact solutions is that there exist precious few known analytic solutions to the Euler equations, even less for the laminar Navier-Stokes equations, and there are currently no known closed-form solutions to the turbulent Reynolds-averaged Navier Stokes equations. So, while the method of exact solutions can provide some very useful information regarding the accuracy of the solver implementation, its applicability is rather limited to a small set of cases for which analytic solutions are known to exist.

3.4.2 Method of Manufactured Solutions

The Method of Manufactured Solutions (MMS) was developed to address the lack of available analytic solutions for certain nonlinear PDEs. The MMS approach allows for a detailed evaluation of specific numerical solver implementations under a much wider range of conditions than was previously possible using MES, and with far greater resolution and accuracy than was ever possible when comparing numerical results to experimentally obtained data.

Rather than try to obtain an exact solution to the governing equations, a solution is manufactured instead. There is no requirement for the prescribed solution to be an exact solution to the governing equations. However, if the prescribed functions are not an exact solution, then additional source terms must be generated and added to the system in order balance the equations. In the present work, the volume integrated source terms, $\mathcal{I}_m^e$, are obtained by substituting the prescribed functions, $\mathbf{Q}_m$, into Equation (3.20).

$$\mathcal{I}_m^e = \int_{\Omega^e} \phi^e \frac{\partial \mathbf{Q_m}}{\partial t}\, d\Omega + \int_{\Omega^e} \phi^e (\vec{\nabla} \cdot \vec{\mathbf{F}}(\mathbf{Q}_m))\, d\Omega - \int_{\Omega^e} \phi^e \mathbf{S}\, d\Omega \tag{3.67}$$

These terms are then added to the other numerical source terms already present in the solver.

This source term is then used to modify the system of equations shown in Equation (3.45).

$$\int_{\Omega^e} \phi^e \frac{\partial \mathbf{Q}^e}{\partial t}\, d\Omega = \int_{\Omega^e} (\vec{\nabla}\phi^e \cdot \vec{\mathbf{F}})\, d\Omega - \int_{\partial\Omega^e} \phi^e \mathbf{H}\, d\Omega + \int_{\Omega^e} \phi^e \mathbf{S}\, d\Omega + \mathcal{I}_m^e \tag{3.68}$$

Under the additional influence of this term, the manufactured solution becomes an exact solution to the modified equations. Thus, in the presence of the manufactured source terms, the solver will tend to drive the numerical solution towards the manufactured solution. The accuracy with which the solver recovers the manufactured solution on a family of refined meshes can then be used to determine the solver's rate of convergence using Equation (3.66).

If the PDEs to be solved are of order s (i.e. s^{th}-order derivatives of the solution are present), then the manufactured solution must be at least C^{s+1} continuous to ensure that

the manufactured source terms are continuous when the differential operators have been applied to the prescribed solution.

If one wishes to test a p^{th}-order-accurate solver, then the manufactured solution should also be at least C^{p+1}-continuous. If this condition is not satisfied, then it is possible that the solver will be able to exactly capture the manufactured solution with the available degrees of freedom. Once this occurs, there will likely be no further convergence of the error norms.

Knupp and Salari describe the MMS in greater detail and provide specific recommendations on functions which are admissible for use with the MMS [58]. Bond *et al.* have recently extended the technique to allow for the validation of several different kinds of boundary conditions [14, 15, 16, 17].

The method of manufactured solutions has been utilized extensively during the development of the solver used in this research. The MMS tests have been an invaluable tool for assessing the accuracy of the solver on nearly arbitrary solutions. The MMS is also very sensitive to a wide variety of errors in solver implementation and problem specification. It is for this reason that the MMS test is often considered to be among the most rigorous verification tests that a solver can be subjected to, second only to full theoretical analysis of the solver algorithms.

Despite the usefulness of the MMS for detecting errors in the solver implementation, the method lacks precision when it comes to identifying the source of the errors. Also, when utilizing arbitrary functions, it is possible for the magnitude of the manufactured source terms to be quite large. If the manufactured source terms are much larger than

the numerical source terms, then the convergence tests may only be showing the rate of convergence for the manufactured source terms. One must take care when selecting the manufactured solution not to inadvertently bias the results in this way. This condition can be easily tested for by purposefully introducing an inconsistency in the solver implementation and checking to see if the convergence tests confirm the presence of the anomaly.

Section 3.5 describes the form of the manufactured solution used to verify the current implementation of the DG flow solver. In Chapter 5, two additional flow models are presented. These models are meant to replicate the behavior of high-Reynolds number flows in a curved duct. These flow models provide the opportunity to evaluate several different mesh configurations by supplying precise error measures throughout the domain.

3.4.3 Method of Nearby Solutions

Similar to the Method of Exact Solutions is the Method of Nearby Solutions (MNS), also known as the Method of Nearby Problems [56]. In this method, the solution to a particular flow problem is obtained from a numerical simulation generated by an alternate solver (which has presumably already been verified and/or validated), typically on a refined mesh. A continuous solution is then generated by interpolating the numerical solution with piecewise high-order splines. Since there is no guarantee that the interpolated function will exactly satisfy the governing equations, it necessary to augment the solver with a set of source terms to balance the equations. The source terms are obtained in the same manner as for the MMS (See Section 3.4.2)..

One significant advantage of MNS is that the magnitude of the source terms used to modify the governing equations are much smaller than with MMS. It should, therefore, be much easier to detect errors generated by the solver since the manufactured source terms are exerting less influence. Another advantage is that the nearby solutions more accurately approximate flow conditions that are likely to be encountered by the solver in practice. The obtained solution is very nearly an exact solution to the PDEs and is, therefore, providing a much more accurate indicator of the kinds of errors that will be generated by the solver when it is used to solve the unmodified governing equations.

The primary challenge to using the MNS is the generation of high-order splined solutions which are required to test the high-order accuracy of the solver. There is a good chance that attempting to interpolate a high-order spline through potentially noisy data could result in oscillatory behavior in the reconstructed function. Another disadvantage to the method is that the nearby solution may be difficult to reproduce precisely. The nearby solution may vary significantly depending on which solver is used to obtain the numerical solution from which the splined solution is generated.

To address the issues mentioned in the preceding sections, it is necessary to take an approach somewhere between the methods of manufactured and nearby solutions. In the current work, a solution generated from a mathematical description is prescribed, but with the the function being specifically selected such that it closely resembles a physically realistic flow profile. The resulting function is somewhat nearby and, therefore, possesses relatively small manufactured source terms. The profile is also easier to reproduce since it is based on a precise mathematical formulation. With this method, it is also possible to

approximate certain specific flow phenomena which are of interest to the solver developer. In this way, the solver can be tested on conditions very near to its intended use while still having an exact representation of the desired output available to compare against and to generate precise error estimates.

3.5 Solver Verification with MMS

The method of manufactured solutions (MMS) has been used throughout the development of the solver for the purposes of code verification. These tests verify that the solver algorithms are operating as expected and establish confidence in the solver's ability to produce solutions of the desired order of accuracy. The MMS tests can be quickly conducted, and they provide data on the observed behavior of a specific solver implementation. To pass the MMS tests, the errors in the solution approximation must be reduced at an expected rate when the computational mesh is isotropically refined.

The numerical solver has been subjected to a number of verification tests designed to detect implementation errors during its development. The manufactured solution verification tests demonstrate that the solver is capable of attaining the desired order of accuracy on a wide variety of mesh and flow configurations when Dirichlet boundary conditions are imposed.

Following the recommendations of Salari and Knupp [58] the prescribed manufactured solutions have been chosen so that they possess continuous derivatives of arbitrarily high-

order. The baseline manufactured solutions utilized for solver verification in the present work have the following functional form:

$$f(x,y,z) = \bar{f} + f_x x + f_y y + f_z z + A\cos(\omega_x x + \psi_x)\cos(\omega_y y + \psi_y)\cos(\omega_z z + \psi_z) \quad (3.69)$$

For the general MMS verification tests, Equation (3.69) is used to generate descriptions for the primitive variables (pressure, temperature, and velocity). The eleven free parameters ($\bar{f}$, f_x, f_y, f_z, A, ω_x, ω_y, ω_z, ψ_x, ψ_y, ψ_z) are randomly generated constants, subject to some constraints which keep the functions bounded to physically realistic values (e.g. positive pressure and temperature), and reasonable frequency content. The latter constraint ensures that the solver can be expected to resolve most of the dominant features in the solution on the coarsest computational mesh. The goal is to be in the asymptomatic range[3] from the coarsest to finest meshes.

In these tests, the boundary conditions are obtained by evaluating the manufactured solution at the mapped physical coordinates of the mesh boundaries, and applying a characteristic decomposition of the flow state in the surface normal direction.

Given the analytic expression for the manufactured solution, the solver automatically computes the source terms and incorporates them into the residuals which are used to drive the evolution of the numerical solution. The solver is run to convergence and the errors in the resulting numerical solution are computed using L_1, L_2, and L_∞ norms. This process is repeated on successively finer meshes. The observed order of accuracy is then computed from the integrated error norms obtained from solutions on two meshes differing only in

[3]The asymptomatic range refers to a range of possible mesh discretizations within which the most significant flow features are properly resolved. Within this range, solution errors are expected to decrease at a rate commensurate with the mesh spacing h^p, where p is the order of the method.

their level of refinement using Equation (3.66). The results of these tests indicate that the DG method is able to maintain high-order accurate convergence on a wide variety of prescribed solutions and mesh configurations.

We present here the results of one such test involving the manufactured solution Equation (3.69) solved over a spherical domain. The discretization of the domain was accomplished by first decomposing the sphere into tetrahedra and then further subdividing each tetrahedron into four hexahedra. (See Figure 3.4) This mesh was then used to generate a family of isotropically refined meshes by subdividing the hexahedral elements. A sample of the error data and convergence rates are presented in Table 3.2 and Table 3.3.

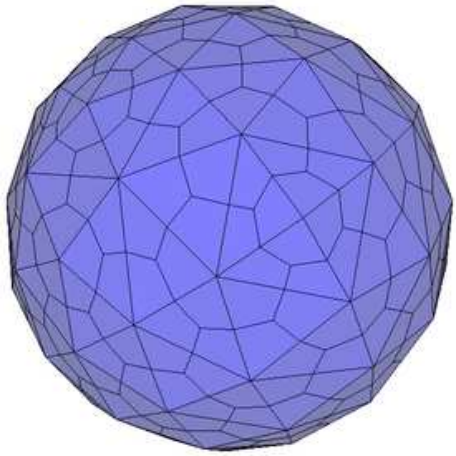

Figure 3.4

Spherical domain discretized with hexahedral elements.

Table 3.2

L_1 errors in the conservative variables for the 2^{nd}-order sphere MMS test

ρ	ρu	ρv	ρw	ρe
L_1-error norm				
0.00110515	0.189053	0.259318	0.294097	234.99
0.00028037	0.0420182	0.0607268	0.07018	60.9734
6.89971e-05	0.00829666	0.0132533	0.0153815	15.2542
Convergence rate				
1.9788	2.1697	2.0943	2.0672	1.9463
2.0227	2.3404	2.1960	2.1899	1.9990

Table 3.3

L_1 errors in the conservative variables for the 3^{rd}-order sphere MMS test

ρ	ρu	ρv	ρw	ρe
L_1-error norm				
0.000132775	0.0316843	0.0393676	0.045941	30.9613
1.64077e-05	0.00288955	0.00400252	0.00470231	4.03834
1.86469e-06	0.000145986	0.000295765	0.000348658	0.468995
Convergence rate				
3.0165	3.4549	3.2980	3.2883	2.9386
3.1374	4.3069	3.7584	3.7535	3.1061

CHAPTER 4

RESEARCH METHODOLOGY

As discussed earlier, there have been numerous reports in the Discontinuous Galerkin (DG) literature of researchers having difficulty obtaining stable solutions – or solutions which converge with the desired order of accuracy – to flow problems in and around domain boundaries with smoothly curving surfaces. In almost all instances, the proposed remedy for the problem involves using curved mesh elements to more accurately approximate the true boundary shape.

Despite the fact that this problem has been successfully diagnosed as being related to the quality of the mesh discretization, there has, as yet, been no detailed study of the precise relationship between the mesh representation and the accuracy and stability of the solver published in the literature. In this chapter we describe an experimental methodology for investigating and documenting this relationship.

4.1 Basic Approach

The verification techniques described in Section 3.4 are capable of serving as very sensitive error detectors; yet by themselves, they provide very little information regarding the nature of the fault that is preventing the solver from attaining its optimal convergence rate. For example, it can be rather difficult to distinguish between implementation errors

(caused by incorrect implementation of solver algorithms) and problem specification errors (errors in the data provided to the solver from which the numerical solution is obtained).

The most expedient course of action is typically to begin with simple verification tests to gain confidence in the core solver algorithms, and then steadily increase the scope and complexity of the tests until the all of the solver capabilities have been verified. Once the solver implementation has been sufficiently verified, however, one can take advantage of this knowledge to deliberately investigate the sensitivity of the solver to variations in the data supplied as input. With the Method of Manufactured Solutions (MMS), it is possible to target very specific flow conditions and, therefore, conduct very specific numerical investigations of the solver.

For the purposes of the current investigation, we will focus on the process of generating computational meshes of sufficient quality to be used for obtaining high-order accurate solutions to inviscid and very high-Reynolds number flow fields near smoothly curving boundaries. The MMS is used to obtain precise error norms for a prescribed solution. Several families of self-similar meshes with telescoping resolutions have been generated and used to obtain numerical solutions to the prescribed problem. The integrated mean error norms and estimates for the condition number of the system matrix are computed for each test case on each mesh.

The following sections provide additional details regarding the methodology used to conduct the numerical experiments. The specific flow models used in the experiments and the resulting data are provided in Chapter 5.

4.2 Specification of the Test Problems

The exact solution to each of the test problems considered in this investigation are specified by an analytic description of the geometry of the domain, Ω, and an exact, analytic description of the solution state at all points within the domain. The prescribed solution state is given as a set of continuous functions describing the primitive solution variables: $\mathbf{q} = [\rho, u, v, w, T]$. If the prescribed solution is not an exact solution to the governing equations, then manufactured source terms are generated as described in Section 3.4.2.

The numerical solution consists of polynomial approximations to the conservative variables: $\mathbf{Q}_h \approx [\rho, \rho u, \rho v, \rho w, \rho e]$, which is computed on a series of meshes with elements that approximate the exact geometry with varying orders of accuracy. The numerical solution is initialized by transforming the primitive variables in the prescribed solution into conservative variables and then projecting the resulting functions into the polynomial spaces within each of the elements as described in Section 3.2.2.1. The treatment of the boundary conditions is described in Section 3.2.5.

4.2.1 Coordinate Spaces

The topology of the hexahedral elements naturally gives rise to a local curvilinear coordinate system within each element in which the coordinate directions are aligned with the edges of the element. We assume the existence of one-to-one onto mapping functions, $\vec{f}_e : \Re^3 \rightarrow \Re^3$, from the parametric to global coordinates, and that these functions are invertible (i.e. the Jacobian of the mappings are non-singular). For the present study, the

parametric to global mapping functions are provided by the Bezier volume descriptions of the mesh elements. (See Section 3.2.1.1.)

The prescribed solutions used in this study have been specified in such a way as to precisely satisfy certain boundary conditions when evaluated at locations along the exact domain boundary. Due to the approximate nature of the mesh representation, however, there exists some ambiguity regarding the locations where the prescribed solution and the numerical solutions are to be evaluated (e.g. for implementing boundary conditions, evaluating source terms, and computing error norms). It is, therefore, necessary to consider an idealized mesh space in which the mesh elements exactly conform to the physical domain. The idealized mesh is the limit to which the approximate mesh will approach as the polynomial degree of the element shapes are increased.

Using the parametric space of each element as a reference, points on the approximate mesh can be associated with points on the idealized mesh. Any evaluation of the prescribed solution will be made in the idealized mesh space, while the numerical solution will be evaluated in the approximate mesh space. In this way, meaningful comparison can be made between the two solutions, and valid boundary condition data can be obtained regardless of the accuracy of the approximate mesh configuration.

All geometry data required by the solver is obtained from the approximate mesh element descriptions. When using the Bezier volume representation to describe the shape of the mesh elements, curved or otherwise, the mapping from parametric to physical coordinates will almost always be a high-order polynomial function. This implies that the determinant of the Jacobian of the inverse mapping and the element surface normals – both

of which are required to integrate the governing equations (3.45) – will be typically be non-constant.[1] Thus, it will be necessary to evaluate these quantities with at least the order of accuracy required by the scheme in all numerical computations (e.g. flux calculations and quadrature integrations).

4.2.2 Domain Discretization

The principal mesh parameters to be varied in these experiments are those which determine the mesh resolution, the distribution of elements within the domain, the polynomial degree of the mesh element shape functions, and the continuity constraints between adjacent elements. The resolution and the placement of the mesh elements within the domain are established for the coarsest mesh such that the polynomial solution approximations in each element stand a good chance of resolving nearly all of the relevant flow features. As the mesh is refined, the relative distribution of the elements remains fixed while each element is further subdivided.

A standard finite volume type mesh is used to establish an initial distribution of linear mesh elements. These mesh nodes, along with the geometric description of the domain boundaries, are then used to generate a collection of mesh elements, which have been curved to approximate the domain boundaries with the desired order of accuracy.

The shapes of the individual elements are represented internally as Bezier volume elements. The principal requirement for the mesh representation is that it forms a valid partition of space. That is, the set of all mesh elements should ideally cover the entire

[1]The only circumstance in which the Bezier representation of a hexahedral element will produce a constant Jacobian is when the shape of the element is a right-parallelepiped.

domain with no gaps and no overlaps. This condition is trivial to implement in the interior of the domain and slightly more challenging to satisfy near curved boundary surfaces. In the interior of the domain, C^0 mesh continuity can be accomplished by simply requiring adjacent elements to reference the same set of control points for each shared edge and face. At the domain boundaries the surface geometry may only be approximated with an order of accuracy commensurate with the order of the elements' Bezier polynomial representations.

For the present work, the control points for the approximating Bezier surface patches in each element are obtained from one of two techniques. The first technique – referred to as Lagrange interpolation – generates m^{th} order surface patches using an $m \times m$ set of sample points evaluated on the provided surface. This technique establishes only C^0-continuity at the interface between elements at the boundary. The second technique – referred to as Hermite interpolation – creates a $2m^{th}$ order patch from the position and normal data evaluated for a set of $m \times m$ sample points on the surface. The Hermite interpolation method allows for the enforcement of a C^1-continuity constraint at the element interfaces along the curved boundary surfaces. In each case, the original mesh nodes provided by the finite volume input mesh are retained as the corner points of each mesh element. For $m > 2$, the remaining sample points are obtained by generating a set of uniformly spaced points between the mesh nodes and then projecting them onto the exact boundary surface. For more details regarding the algorithms used to generate the Lagrange and Hermite surface patches, refer to Sections 3.2.1.2 and 3.2.1.3.

4.3 Error Evaluation

The accuracy of a given solver implementation is often difficult to assess without access to the exact solution. The investigation makes use of the method of manufactured solutions to obtain reliable measures of solution error. The basic approach of this method is given in Section 3.4.2.

The solver is initialized by projecting the prescribed solution into the polynomial approximation space within each element as described in Section 3.2.2.1. The solver is then allowed to run (under the influence of the manufactured source terms if applicable) until the magnitude of the residual falls below some tolerance or the integrated error norms have converged to greater than four digits of precision.

The error metrics used in the present work are the integrated L_s-norms of the difference between the prescribed solution and the numerical solution.

$$\mathcal{E}_s = \left(\int_\Omega \left(|\mathbf{Q} - \mathbf{Q}_h| \right)^s \, d\Omega \right)^{\frac{1}{s}} \tag{4.1}$$

In the solver, this integral is computed using numerical quadrature over each element. The volume-weighted element-wise errors are summed up and divided by the total volume of the mesh to obtain the mean total integrated error norm. The use of the mean error (volume weighted and normalized) allows a more meaningful comparison with the L_∞-norm and mean error norms computed on other mesh configurations, which may have a different distribution of mesh elements and/or total mesh volume.

$$E_s^e = \sum_{q=1}^{N_q} \left(\left(|\mathbf{Q}^e - \mathbf{Q}_h^e| \right)^s \left(J^{-1} \right)^e \right)_q w_q \tag{4.2}$$

$$\mathcal{E}_s = \left(\frac{1}{V} \sum_{e \in T_h} E_s^e \right)^{\frac{1}{s}} \tag{4.3}$$

Here V is the volume of the entire mesh, $\mathbf{Q}^e$, $\mathbf{Q}_h^e$, and $(J^{-1})^e$ are evaluations of the prescribed solution, the numerical solution, and the determinant of the Jacobian of the inverse coordinate mapping at each quadrature point within each element.

The error norms reported in the next chapter have been computed using the L_1-norm. Errors computed with the L_2 and L_∞-norms have also been computed and observed to converge at approximately the same rate as the L_1 error unless specifically noted otherwise.

4.4 Condition Number Evaluation

The stability of the solver as well as the ability of the solver to reach an accurate, converged solution with a reasonable amount of computational effort are issues of much practical concern. When considering the solver implementation, obtaining the solution to the linearized equations is often where the greatest amount of computing effort is expended. The stiffness of the problem under consideration influences the stiffness of the system matrix which, in turn, influences the efficiency and stability of the solver. The condition number of the system matrix – the ratio of its maximum to minimum eigenvalues – is usually a fairly good measure of the ability of the solver to efficiently obtain a solution to a given flow problem. For large matrices it is often impractical to compute the eigenvalues directly. Instead, we use an estimate of the condition number, which we take as a lower bound on the actual condition number.

In addition to the essential stiffness of the flow problem under consideration, there exist a number of other non-essential sources of matrix stiffness which can affect the mag-

nitude of the computed or estimated condition number. One such source of non-essential stiffness is caused by poor matrix scaling. In the present solver implementation, most of the adverse matrix scaling effects are likely due to the choice of units for the conserved solution variables. For non-dimensional codes, the choice of reference quantities would have a similar effect.

Since we would like to infer something about the stiffness of the problem from an estimate of the condition number, we must attempt to remove as much of the non-essential stiffness from the matrix as possible. The removal, or reduction, of the influences of matrix scaling on the condition number should permit a more reliable evaluation of the effects of the essential stiffness of the flow problem. The matrix scaling algorithm described in Section 3.2.7 should provide a consistent matrix representation regardless of any particular choice of units or reference quantities. The scaled matrix is then used to obtain the condition number estimate.

To compute the condition number estimate, we let $\hat{\mathbf{A}} = (\mathbf{D}_1\mathbf{A}\mathbf{D}_2)$ be the scaled system matrix resulting from Algorithm 3.2. For the linear system $\hat{\mathbf{A}}\mathbf{z} = \mathbf{w}$, the condition number may be estimated by the following expression:

$$\kappa_1(\hat{\mathbf{A}}) \geq \frac{||\hat{\mathbf{A}}||_1||\hat{\mathbf{A}}^{-1}\mathbf{w}||_1}{||\mathbf{w}||_1} = \frac{||\hat{\mathbf{A}}||_1||\mathbf{z}||_1}{||\mathbf{w}||_1} \tag{4.4}$$

Equality will occur when the sample vector $\mathbf{z}$ is aligned with the direction of maximum elongation of the matrix. Since it may be difficult to determine this direction *a priori*, we utilize an iterative approach where a large number of randomly generated sample vectors, $\mathbf{z}^{(k)}$, are transformed by the system matrix, $\mathbf{w}^{(k)} = \hat{\mathbf{A}}\mathbf{z}^{(k)}$. The condition number estimate

is then computed from Equation (4.4), with the largest value taken as the lower bound on the actual condition number.

$$\kappa_1(\hat{\mathbf{A}}) \geq \max_k \left(\frac{||\hat{\mathbf{A}}||_1 ||\mathbf{z}^{(k)}||_1}{||\mathbf{w}^{(k)}||_1} \right) \tag{4.5}$$

In the present work we use a sample size of five-hundred randomly generated vectors. We then examine these estimates and use them as a somewhat crude tool for evaluating the relative stiffness of the system matrix when the same flow problem is evaluated on multiple meshes. We seek to determine if such variations in mesh configurations have any significant effects on the condition of the system matrix.

CHAPTER 5

NUMERICAL RESULTS

5.1 Scope of the Numerical Investigation

The present work seeks to determine the effects of mesh discretization of smoothly curving solid wall boundaries on the accuracy and stiffness of the computed solution. To accomplish this, two model flow fields inside of a curved duct domain are investigated. The first model prescribes an exact solution to the inviscid Euler equations (3.1), (3.2), and (3.3), while the second model prescribes a flow profile which approximates the behavior of a turbulent boundary layer. The latter profile is designed to be a nearby solution to the Navier-Stokes equations (3.4), (3.5), and (3.6). The boundary layer profile has been carefully selected such that the prescribed solution possesses many of the expected properties of a high-Reynolds number flow near a curved solid wall boundary.

There are several variations in the flow profiles that could be generated by varying the input parameters to each model. In the following sections, the implementation details for each model is provided, followed by the numerical results from a select few instantiations of these flow models.

Figure 5.1

Flow schematic for the curved duct domain geometry.

5.2 Supersonic Vortex

The first test case is commonly referred to as the supersonic vortex [1, 60]. This exact solution to the inviscid equations is an idealized vortex flow. The domain is a curved duct bounded by two circular arcs (See Figure 5.1). The internal flow is tangentially directed around the bend by pressure and density variations in the radial direction. The prescribed density distribution is given by the following expression (Figure 5.2).

$$\rho(r) = \rho_i \left[1 + \frac{\gamma - 1}{2} M_i^2 \left\{ 1 - \left(\frac{r_i}{r} \right)^2 \right\} \right]^{\frac{1}{\gamma - 1}} \tag{5.1}$$

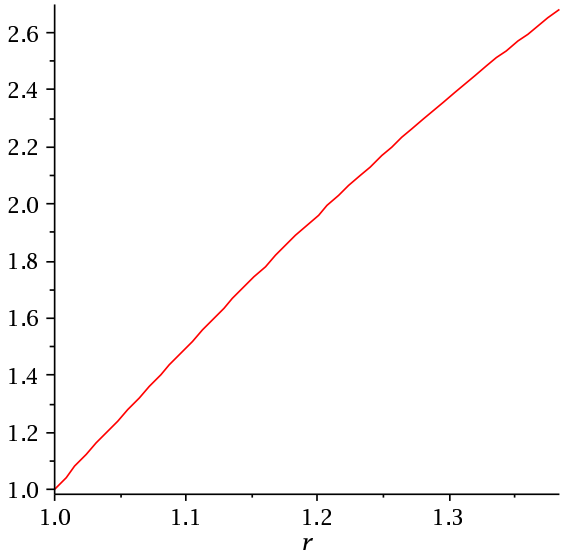

Figure 5.2

Density profile for the SSV test case.

The velocity of the flow is oriented tangential to the cylinder at all points in the domain (no flow along the axis or in the radial direction), while the magnitude of the flow velocity is inversely proportional to the radius.

$$\vec{u}(r,\theta) = u_i \left(\frac{r_i}{r}\right) (r\sin\theta\ \hat{\imath} - r\cos\theta\ \hat{\jmath}) \tag{5.2}$$

Here r_i is the radius of inner arc, and $u_i = M_i a_i$ is the magnitude of the flow velocity at the inner channel wall (Figure 5.3).

The remaining solution state is obtained from the isentropic conditions:

$$\frac{T_0}{T} = \left(1 + \frac{\gamma - 1}{2} M^2\right) = \left(\frac{a_0}{a}\right)^2 = \left(\frac{P_0}{P}\right)^{\frac{(\gamma-1)}{\gamma}} = \left(\frac{\rho_0}{\rho}\right)^{(\gamma-1)}, \tag{5.3}$$

and the equations of state for a perfect gas:

$$P = \rho RT, \tag{5.4}$$

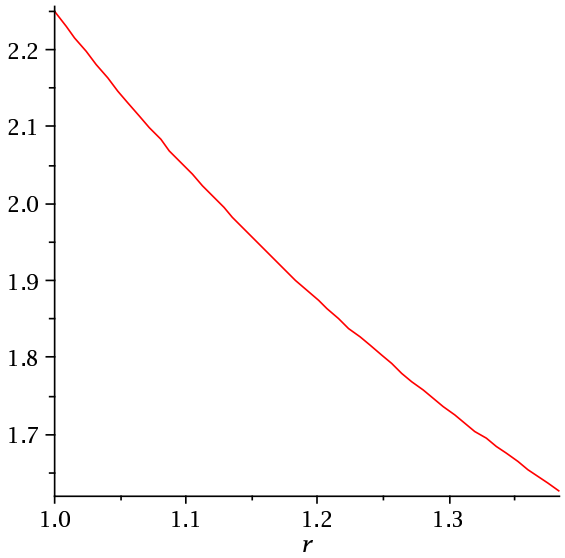

Figure 5.3

Mach number profile for the SSV test case.

$$P = (\gamma - 1)\left(\rho e - \frac{1}{2}\rho\vec{u}\cdot\vec{u}\right), \tag{5.5}$$

$$a = \sqrt{\gamma RT} \tag{5.6}$$

The radial variations in pressure and temperature are plotted in Figures 5.4 and 5.5.

The results presented in the next section are for runs of the supersonic vortex test case with the following conditions:

$$r_i = 1.0 \text{ m} \quad r_o = 1.384 \text{ m} \quad M_i = 2.25 \quad T_i = 300 \text{ K} \quad \rho_i = 1.0 \text{ kg/m}^3 \tag{5.7}$$

Figure 5.6 displays the first three mesh refinements used in the SSV test cases. Although the meshes shown have linear element shapes, all curved meshes also utilize the same distribution and spacing of mesh elements.

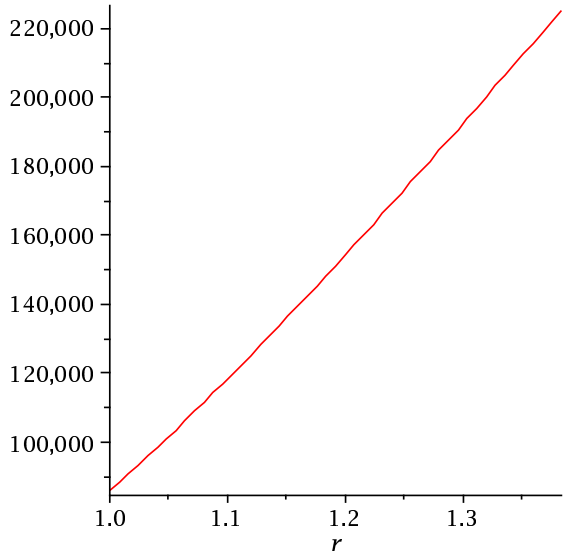

Figure 5.4

Pressure profile for the SSV test case.

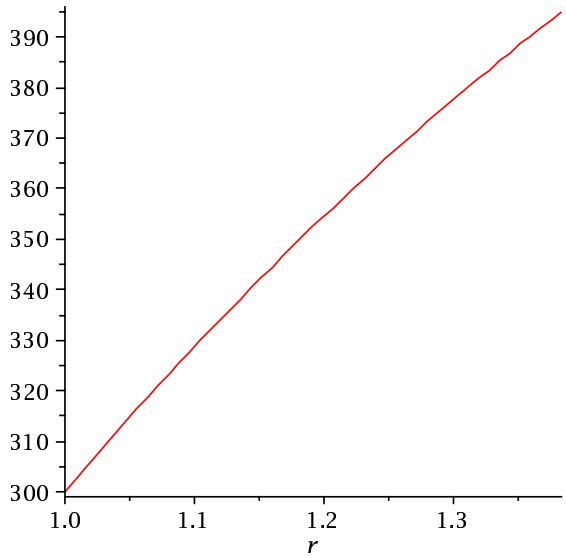

Figure 5.5

Temperature profile for the SSV test case.

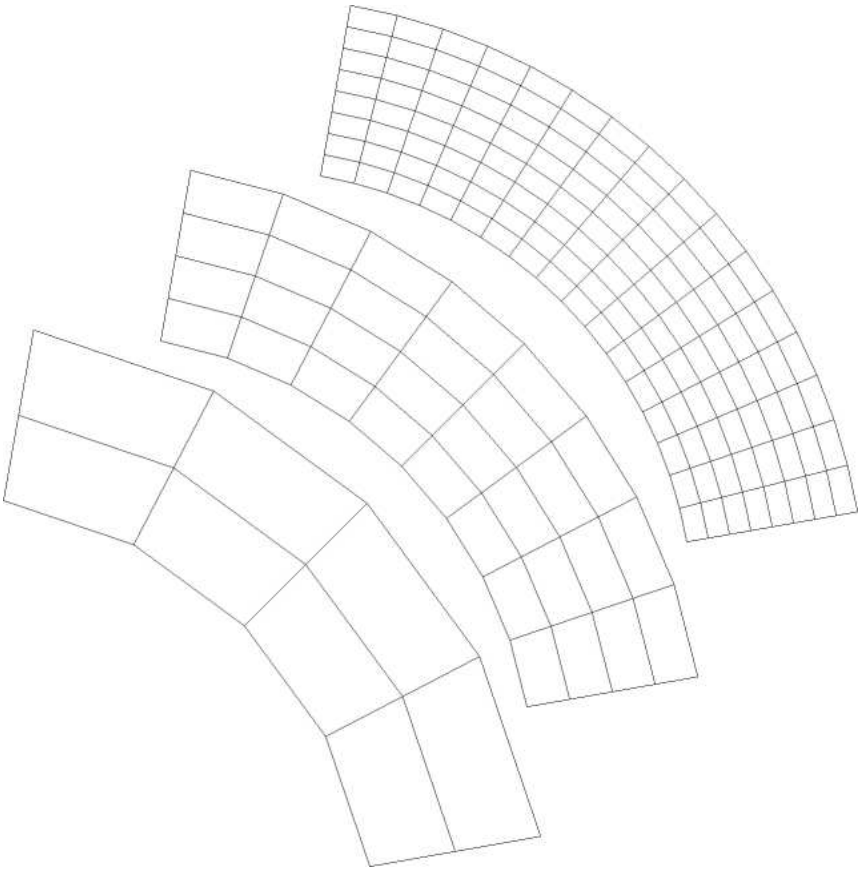

Figure 5.6

First three mesh refinements for the SSV test cases

Time integration is performed using the backward Euler method. Since this is an exact solution to the inviscid equations, there are no manufactured source terms present. The solver is run to convergence, where the residuals are allowed to drop to 10^{-10}.

The solver computed the solution to the supersonic vortex case with first through fifth-order solution approximations, and on seven different mesh configurations (C2, C3, C4, C5, C6, H2, H3). The L_1-error norm data for these runs are plotted on the following pages. The raw data is provided in Appendix A.

5.3 Results for the Supersonic Vortex Cases

Figures 5.7 and 5.8 display logarithmic plots of the L_1-error norms in density and x-momentum against the reciprocal of mesh spacing for the supersonic vortex case run with a first-order (piecewise constant) solution approximation. The solid line is provided for reference and represents the slope at which first-order convergence is attained. Since there is no discernible difference in computed error norms for these solutions, then it would appear that the choice of mesh discretization is irrelevant for first-order solutions.

Figures 5.9 and 5.10 plot the L_1 density and momentum error norms computed with a second-order solution approximation. While all of the solutions converge at second-order rate, the errors are clearly greater when the linear (C2) elements are used. No clear advantage is gained, however, by using elements with greater than third-order accuracy.

Figures 5.11 and 5.12 show the L_1 density and momentum error norms for the third-order solutions. Once again, it can be seen that the curved mesh elements are providing much more accurate solutions. This time, however, the computed error norms for the linear

elements (C2) are only converging at a second-order rate, while all of the meshes which use curved elements (C3, C4, C5, C6, H2, H3) appear to be converging at the expected third-order rate. As before, there are no clear improvements in the error norms when using higher than third-order elements.

The fourth-order errors are plotted in Figures 5.13 and 5.14. The linear elements are still generating a significant amount of error and limiting the solution to second-order convergence. The curved elements are all converging at a fourth-order rate; however, now there appears to be some small difference in the magnitude of the computed error norms amongst the higher-order curved elements. In order of least to most accurate: Hermite cubic (H2), Lagrange quadratic (C3), Lagrange cubic (C4), then all higher-order elements (C5, C6, H3). With the exception of the Hermite cubic elements, the data indicate that there is a slight reduction in error with higher-order geometry representation, although beyond fifth-order (C5), there does not appear to be any noticeable improvement.

Since the quadratic elements appear to be attaining fourth-order accurate convergence rates, there would seem to be a violation of the iso-parametric element criteria which is typically cited in the literature. To determine if this trend continues, the supersonic vortex case was run again with a fifth-order solution approximation. The L_1-error norms are plotted in Figures 5.15 and 5.16. From this data, it can clearly be seen that both the quadratic (C3) and cubic (C4, H2) elements are limited to fourth-order convergence, while the fifth-order (C5), and above (C6, H3) elements maintain the expected fifth-order rate.

This data clearly shows that the linear elements are incapable of delivering higher than second-order accuracy, regardless of the polynomial order of the solution represen-

tation. Figures 5.17 and 5.18 show the behavior of the error norms for just the linear (C2) elements when used with first through fifth-order solution approximations. The second through fifth-order solutions appear to be limited by the second-order accuracy of the mesh representation. It is also worth noting that, for a given mesh resolution, the magnitudes of the error norms actually increase with higher-order solution approximations. This data is consistent with observations in the literature which suggest that the use of linear mesh elements may cause non-physical unsteady phenomena to develop in response to the faceted surface geometry. Since the higher-order schemes produce much less numerical dissipation, they become less and less capable of damping out these non-physical flow features before they contaminate the rest of the flow field.

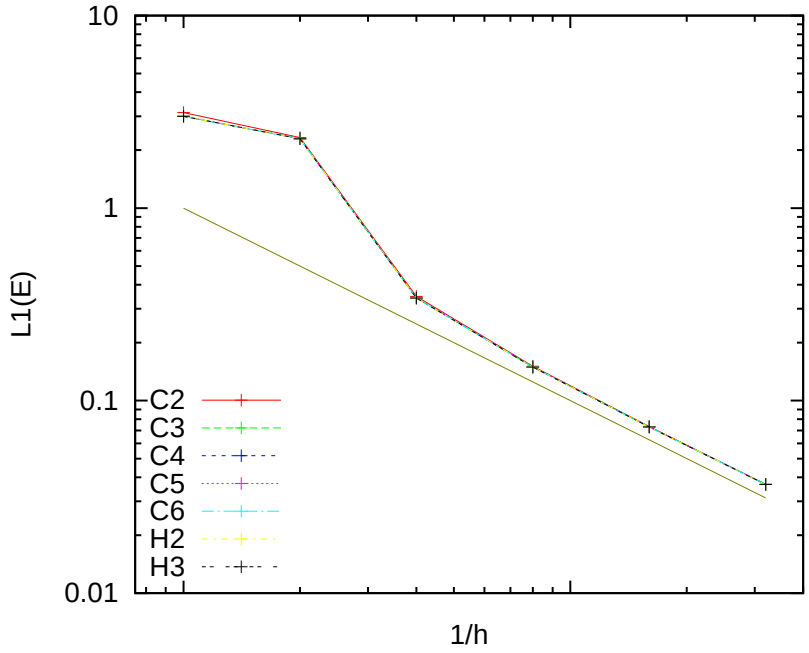

Figure 5.7

L_1-errors in ρ for a 1^{st}-order solution of the SSV

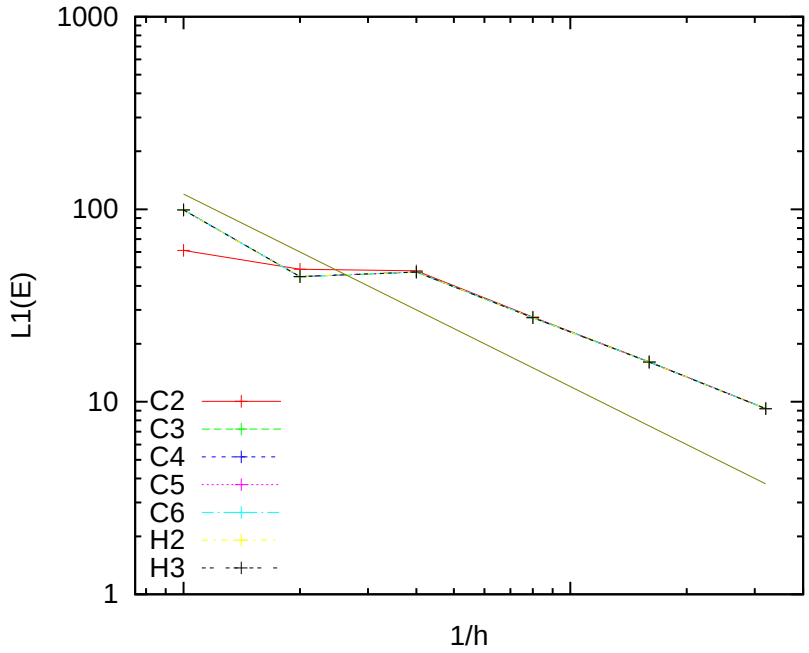

Figure 5.8

L_1-errors in ρu for a 1^{st}-order solution of the SSV

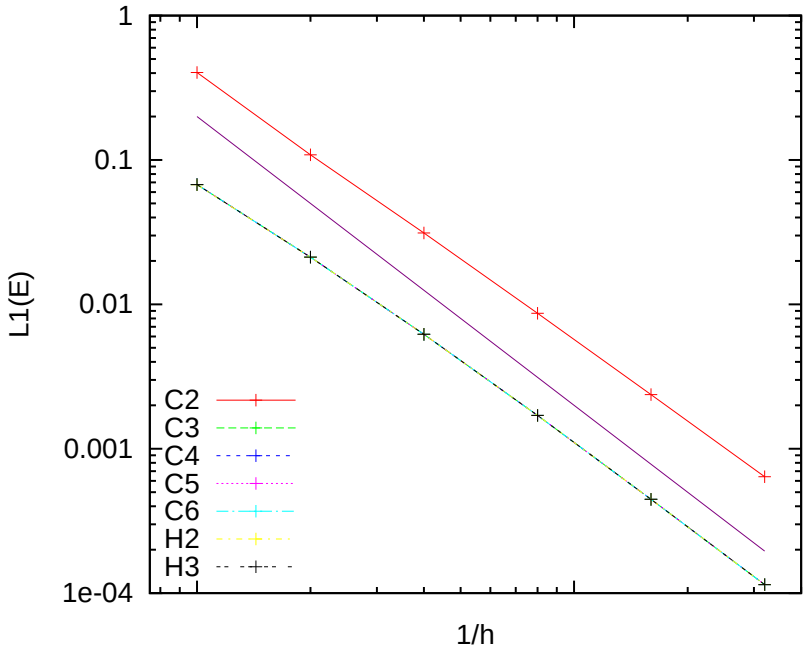

Figure 5.9

L_1-errors in ρ for a 2^{nd}-order solution of the SSV

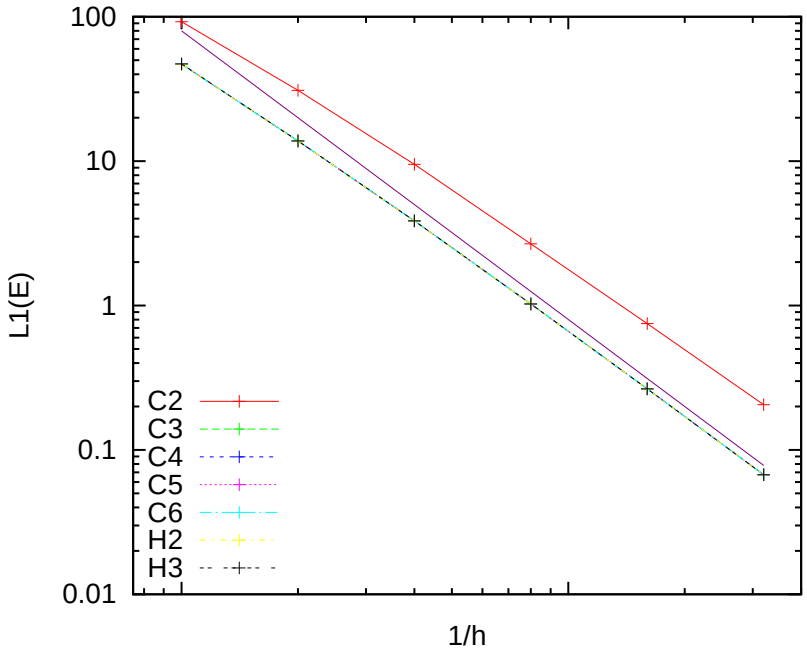

Figure 5.10

L_1-errors in ρu for a 2^{nd}-order solution of the SSV

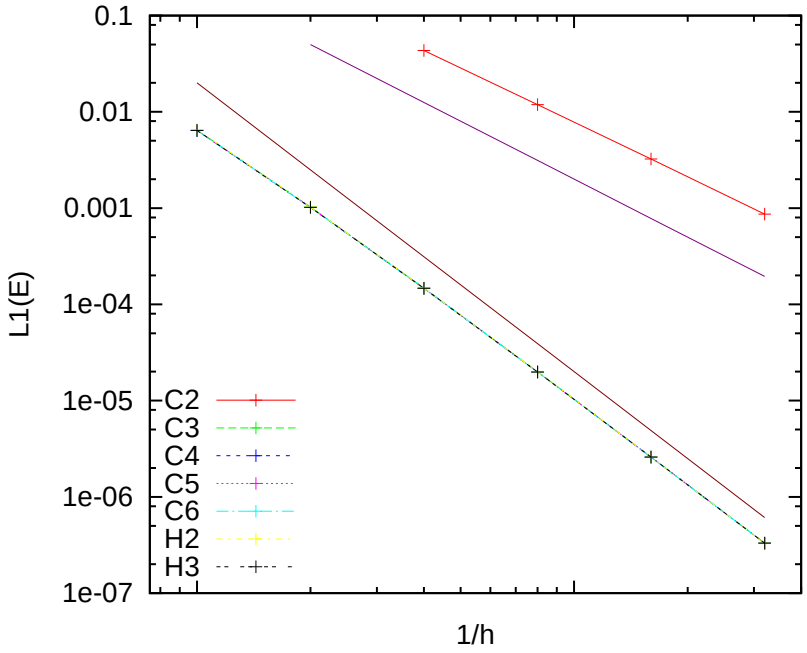

Figure 5.11

L_1-errors in ρ for a 3^{rd}-order solution of the SSV

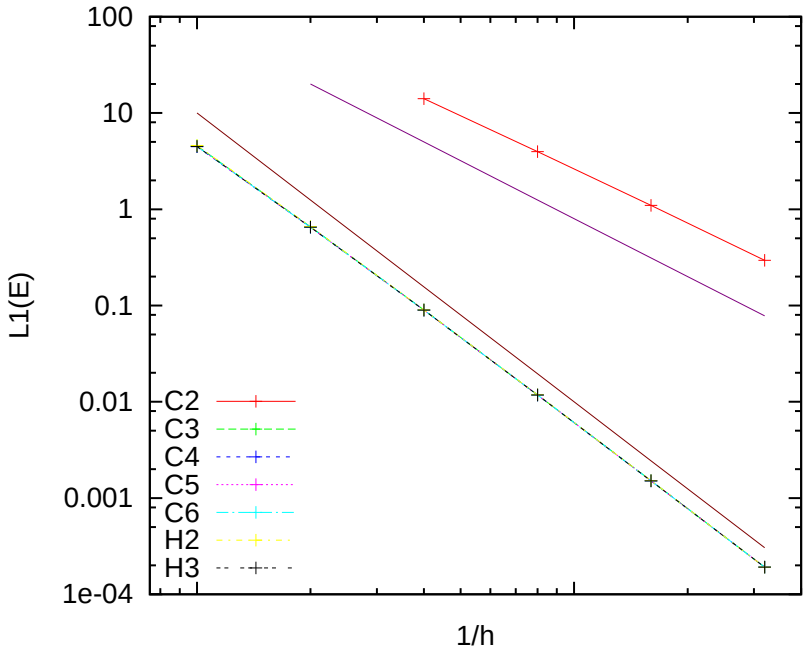

Figure 5.12

L_1-errors in ρu for a 3^{rd}-order solution of the SSV

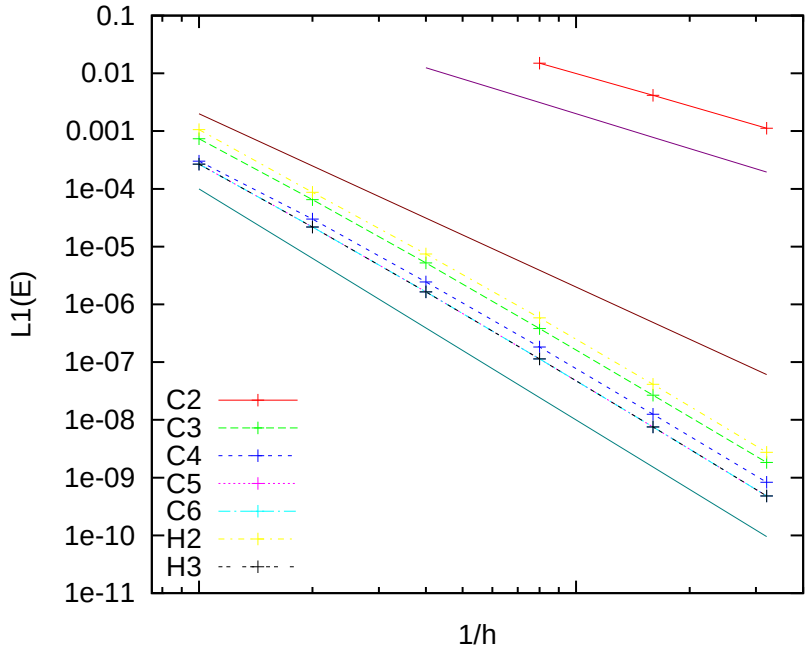

Figure 5.13

L_1-errors in ρ for a 4^{th}-order solution of the SSV

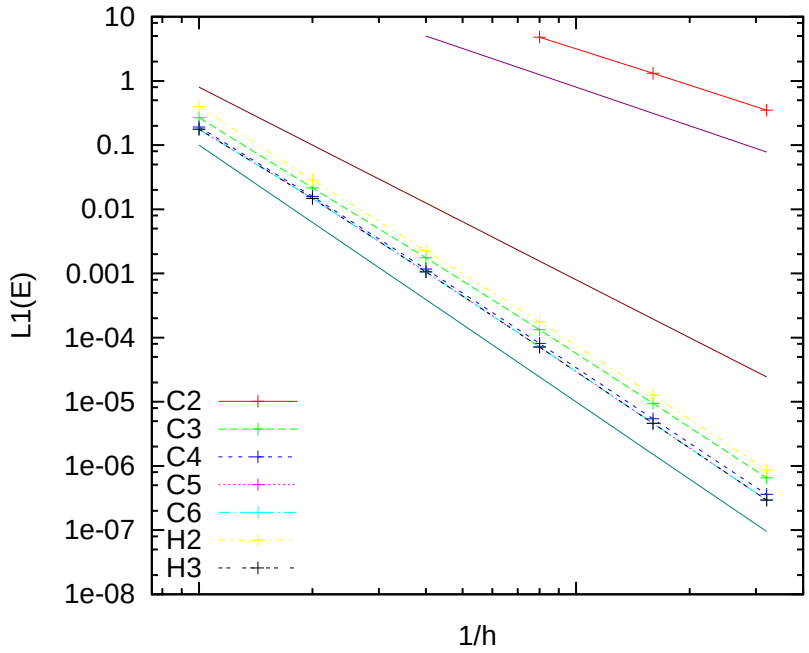

Figure 5.14

L_1-errors in ρu for a 4^{th}-order solution of the SSV

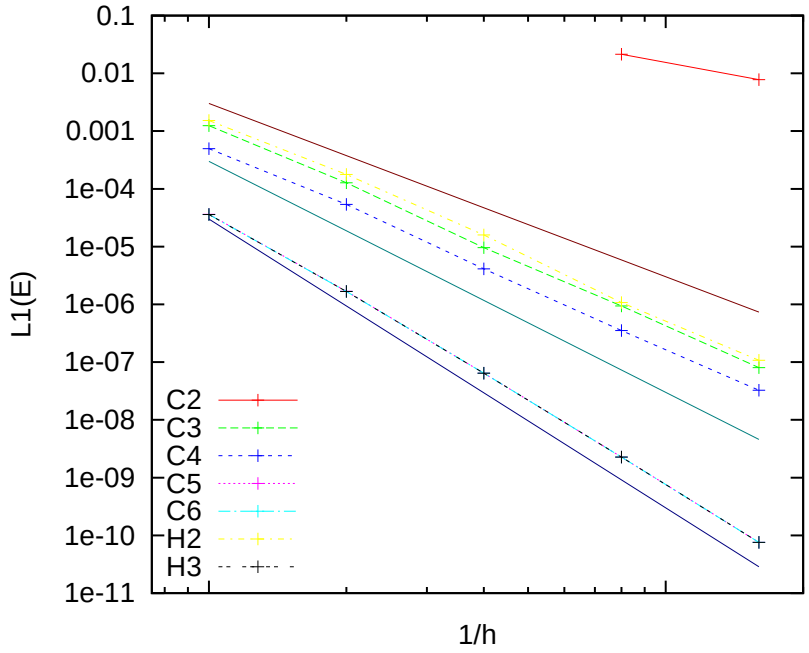

Figure 5.15

L_1-errors in ρ for a 5^{th}-order solution of the SSV

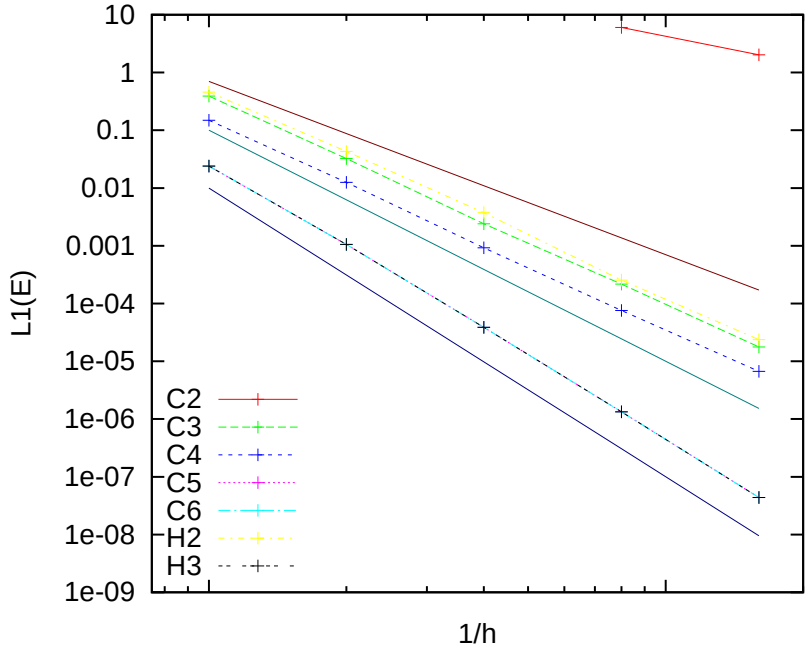

Figure 5.16

L_1-errors in ρu for a 5^{th}-order solution of the SSV

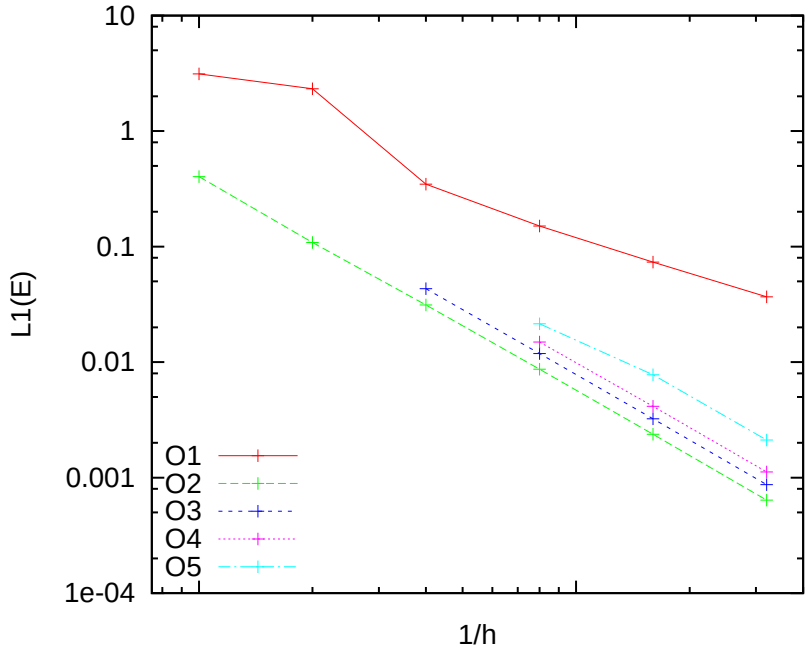

Figure 5.17

L_1-errors in ρ for all solution orders (SSV)

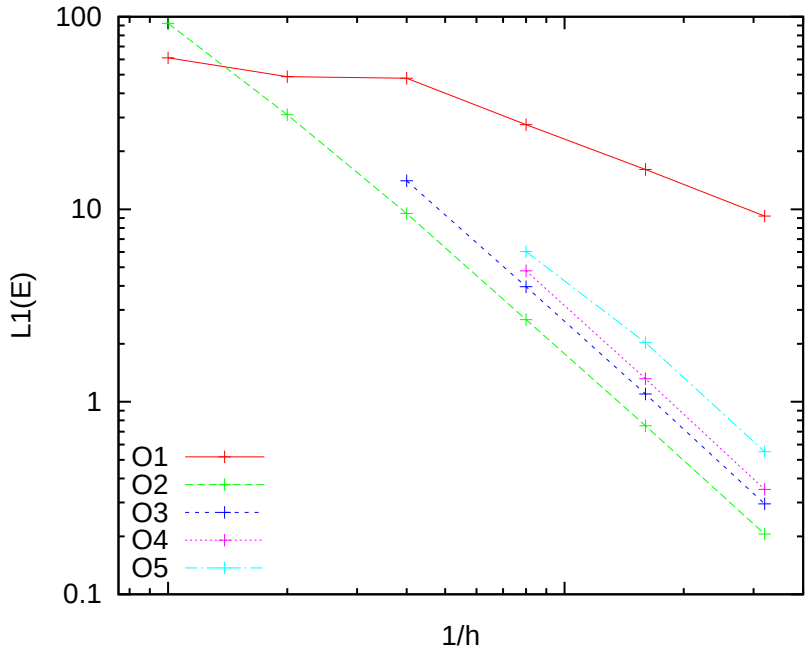

Figure 5.18

L_1-errors in ρu for all solution orders (SSV)

5.4 High-Reynolds Number Turbulent Boundary Layer

We present here a new nearby, manufactured solution which attempts to mimic many of the features typically found in turbulent boundary layers. As with the method of nearby solutions, the goal is to make the source terms as small as possible by prescribing a solution state which is as close to a real solution as is practical. Unlike the traditional method of nearby solutions, which utilize a splined interpolation of a solution computed from another flow solver, the prescribed flow state in this test case will be derived from the analytically and empirically derived expressions which comprise the well known law-of-the-wall relations [70, 72].

Although the law-of-the-wall relations which form the basis of this profile were originally derived for a flat plate, they are adapted here for use with the same curved duct geometry used in the supersonic vortex case (Figure 5.1). Using this configuration will permit the investigation of the behavior of the solver when highly anisotropic flow features are present near smoothly curving domain boundaries. To accomplish this objective with minimal added complexity to the flow profile model, a single boundary layer cross-section, generated from the flat-plate boundary layer expressions, is mapped into the radial direction of the curved duct domain. In this configuration, the inner duct wall serves as an adiabatic, no-slip, viscous wall.

It should be noted here that no attempt is being made to precisely model the actual flow that would develop in the duct. Rather, we are specifying a flow profile model, which is based on some well known relations from boundary-layer theory. This model has a fairly

convenient mathematical description and possesses some of the general characteristics of attached turbulent boundary layers.

Beginning with the density and temperature profiles for the supersonic vortex profile described in Section 5.2, the flow velocity profile is modified to approximate the rate of growth of the flow speed in a viscous boundary layer. The radial cross-section of the velocity profile is taken to be that of a turbulent, flat plate boundary layer that has traveled a distance of x along the flat plate before entering the curved duct. The Reynolds number of the flow is given by:

$$Re = \frac{\rho_\infty u_\infty x}{\mu_\infty} \tag{5.8}$$

and the width of the boundary layer is given by:

$$\delta = 0.074\ x\ Re^{-\frac{1}{5}} \tag{5.9}$$

The velocity profile consists of a piecewise function which maintains at least C^3-continuity throughout the domain, as required by the MMS technique. The functional form of the velocity profile is specified in two parts: the viscous sublayer is represented as a sixth-order polynomial, and the log-layer is of the standard law-of-the-wall form [70].

$$u^+(y^+) = \begin{cases} \sum_{i=0}^{5} a_i (y^+)^i & \text{for } y^+ \leq 30 \\ \frac{1}{\kappa}\ln(y^+) + C & \text{for } 30 < y^+ \leq y_\delta^+ \end{cases} \tag{5.10}$$

The scaled wall distance (y^+) is related to the radial distance by:

$$y^+ = \frac{\rho_w u_\tau}{\mu_w}(r - r_i) \tag{5.11}$$

and the scaled velocity (u^+) is related to the magnitude of the tangential velocity ($\hat{u}$) by:

$$u^+ = \frac{\hat{u}}{u_\tau} \tag{5.12}$$

Here, the friction velocity (u_τ) is:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho_w}} \tag{5.13}$$

Terms sub-scripted with a w denote quantities evaluated at the wall (ρ_w: density, μ_w: viscosity, τ_w: shear stress). In terms of the skin friction (c_f), freestream density (ρ_∞), and freestream velocity ($\hat{u}_\infty$), the wall shear stress is:

$$\tau_w = \frac{1}{2}\rho_\infty \hat{u}_\infty^2 c_f \tag{5.14}$$

where

$$c_f = 0.36\ Re^{-\frac{1}{5}}. \tag{5.15}$$

The spline fit to the viscous sublayer must satisfy two conditions at the wall and four conditions at $y^+ = 30$. The polynomial coefficients for the resulting sixth-order spline are thus derived from the following conditions:

At $y^+ = 0$:

$$u^+ = 0, \tag{5.16}$$

$$\frac{\partial u^+}{\partial y^+} = 1, \tag{5.17}$$

$$\tag{5.18}$$

and at $y^+ = 30$:

$$u^+ = \frac{1}{\kappa}\ln(30) + C, \tag{5.19}$$

$$\frac{\partial u^+}{\partial y^+} = \frac{1}{\kappa}\left(\frac{1}{30}\right), \tag{5.20}$$

$$\frac{\partial^2 u^+}{(\partial y^+)^2} = \frac{1}{\kappa}\left(-\frac{1}{30^2}\right), \tag{5.21}$$

$$\frac{\partial^3 u^+}{(\partial y^+)^3} = \frac{1}{\kappa}\left(\frac{1}{30^3}\right). \tag{5.22}$$

Since these derivatives are satisfied in the (u^+,y^+)-space, only one set of coefficients are required, regardless of the effects of scaling the boundary layer. If we let $\kappa = 0.41$ and $C = 4.9$, then we have:

$$\begin{aligned}
a_0 &= 0 \\
a_1 &= 1 \\
a_2 &= \frac{10}{369}\ln(30) - \frac{11083}{110700} \\
a_3 &= \frac{877}{184500} - \frac{2}{1107}\ln(30) \\
a_4 &= \frac{1}{22140}\ln(30) - \frac{6913}{66420000} \\
a_5 &= \frac{403}{467015625} - \frac{1}{2490750}\ln(30)
\end{aligned}$$

Since there is no turbulence model currently implemented, a turbulent viscosity profile is provided as part of the prescribed solution. There are two principal parts to the turbulent viscosity profile; the inner and outer layers. The inner viscosity profile derives from the constant shear stress assumption and has the form:

$$\mu_{t_i} = \frac{\tau_w}{\frac{d\hat{u}}{dr}} - \mu_s \tag{5.23}$$

The outer viscosity profile is based on the Cebeci-Smith model [72]:

$$\mu_{t_o} = \alpha\ \rho\ \hat{u}_\infty\ \delta_v^*\ F_{kleb}(y;\delta), \tag{5.24}$$

where

$$l_{mix} = \kappa y\left[1 - e^{-y^+/A^+}\right], \tag{5.25}$$

$$F_{kleb}(y;\delta) = \left[1 + \left(\frac{y}{\delta}\right)^6\right]^{-1}, \tag{5.26}$$

$$\delta_v^* = \int_0^\delta (1 - \hat{u}/\hat{u}_\infty)\, dy, \tag{5.27}$$

$$\kappa = 0.41, \qquad \alpha = 0.0168, \qquad A^+ = 26 \tag{5.28}$$

The combined profile is typically treated as a piecewise C^0-continuous function:

$$\mu_t = \begin{cases} \mu_{t_i} & y^+ < y_m^+ \\ \mu_{t_o} & y^+ > y_m^+ \end{cases} \tag{5.29}$$

with y_m^+ being the location where the values of the two functions intersect. Figure 5.19 plots the inner and outer turbulent viscosity profiles (in the radial direction) as the red and blue traces, respectively. Note that the radial component of shear stress ($\tau = (\mu_s + \mu_t)\partial_r \hat{u}$), plotted in Figure 5.20, is only C^0-continuous. This results in a discontinuity in the radial component of the manufactured source term for momentum ($\nabla \cdot \tilde{\sigma} = \partial_r \tau\ \hat{r}$) as shown in Figure 5.21. In order to meet the smoothness requirements for the MMS, the source term should be at least C^1-continuous. As stated in Section 3.4.2, quantities in the profile should be at least one order higher in continuity than the highest derivative taken of that quantity. Since the Navier-Stokes equations posses first-order derivatives of viscosity, the viscosity profile must therefore be at least C^2-continuous.

One way to generate a smooth viscosity profile is to utilize a blending function to smoothly transition from one function to the next.

$$\mu_{t_{kleb}} = F_{kleb}(y^+; y_m^+)\mu_{t_i} + (1 - F_{kleb}(y^+; y_m^+))\mu_{t_o} \tag{5.30}$$

This blended profile is plotted in Figure 5.19, as the green trace. In Figure 5.20, one can see that the shear stress is continuous, as is the source term in Figure 5.21. Although this form of the viscosity function does provide a smooth transition between the inner and outer

viscosity profiles, the resulting manufactured source term possess a significant oscillation near y_m^+. As the Reynolds number increases, this feature becomes even more anisotropic; the amplitude of the oscillation grows while it's width diminishes. The presence of highly anisotropic features in the source terms is not necessarily a show-stopper; however, it does impact the requirement that the mesh be made fine enough to enable accurate integration of the source terms.

Rather than concern ourselves with adapting the mesh to resolve arbitrary features in the source terms, an alternative blending technique is utilized which does not result in a substantial oscillation in the source term.. In this approach, a sixth-order spline is used to patch together the two viscosity layers. The spline function is constructed to match the inner and outer viscosity profiles, and their first two derivatives, at a distance of $\epsilon_m y_m^+$ on either side of y_m^+.

$$\mu_{t_s} = \sum_{i=0}^{5} c_i \, (y^+)^i \tag{5.31}$$

$$\mu_t = \begin{cases} \mu_{t_i} & y^+ \leq (1-\epsilon_m) y_m^+ \\ \mu_{t_s} & (1-\epsilon_m) y_m^+ < y^+ \leq (1+\epsilon_m) y_m^+ \\ \mu_{t_o} & y^+ > (1+\epsilon_m) y_m^+ \end{cases} \tag{5.32}$$

In the present work, $\epsilon_m = 0.5$, and the spline coefficients, c_i, which satisfy the continuity constraints, are numerically computed. As can be seen in Figure 5.21, the source terms due to the splined viscosity profile (red, black, and blue traces) maintain the constant shear stress assumption of the inner viscosity layer further into the domain before smoothly transitioning to the outer layer without the oscillatory behavior observed with the previous approach.

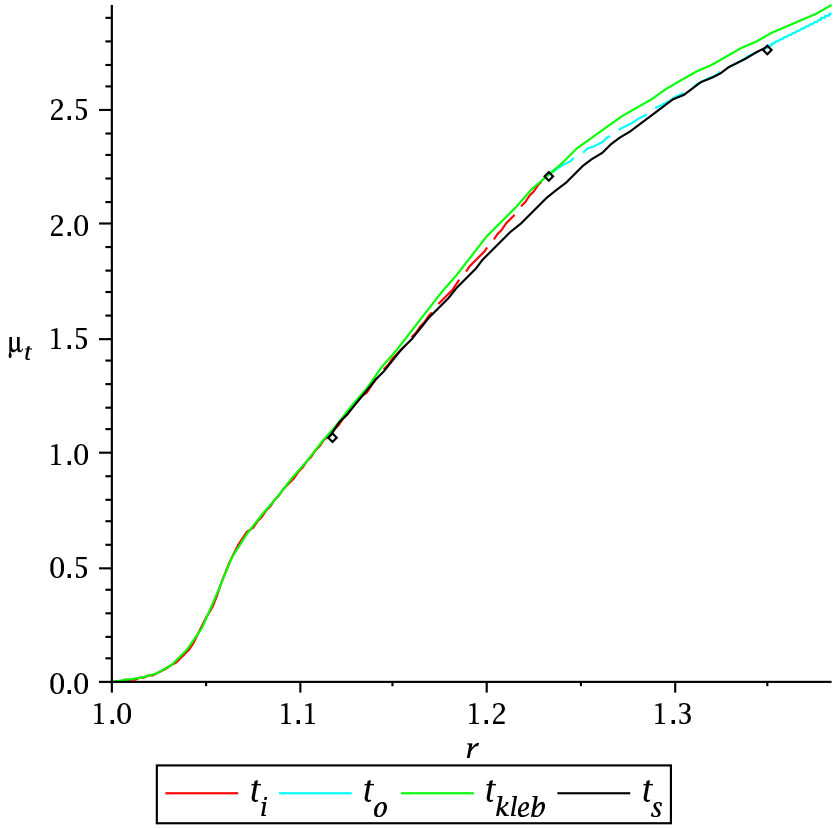

Figure 5.19

Turbulent viscosity profiles

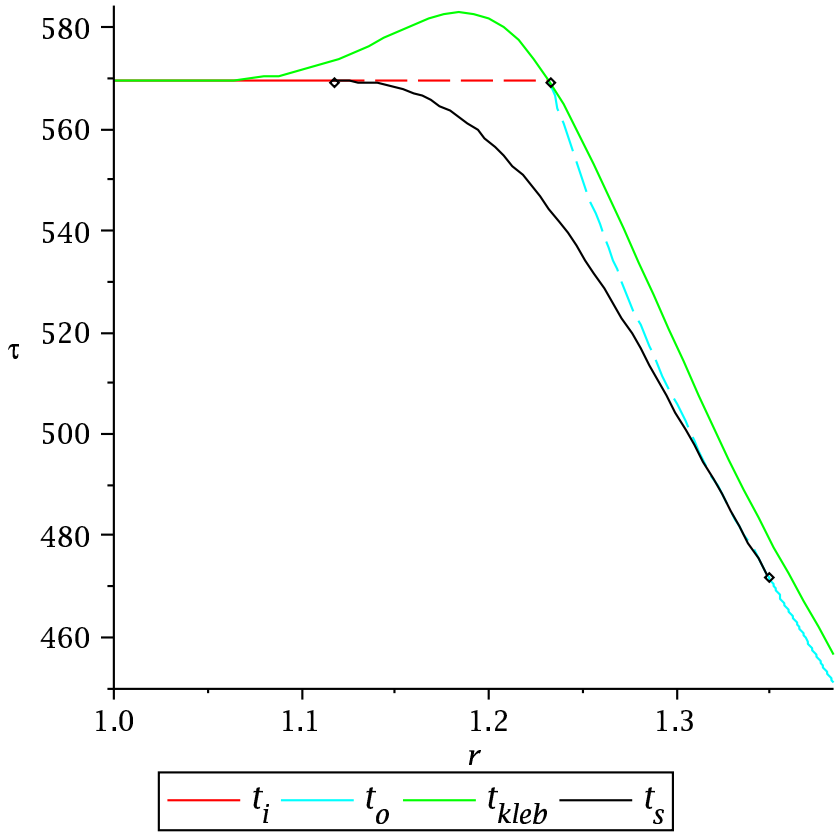

Figure 5.20

Radial component of shear stress

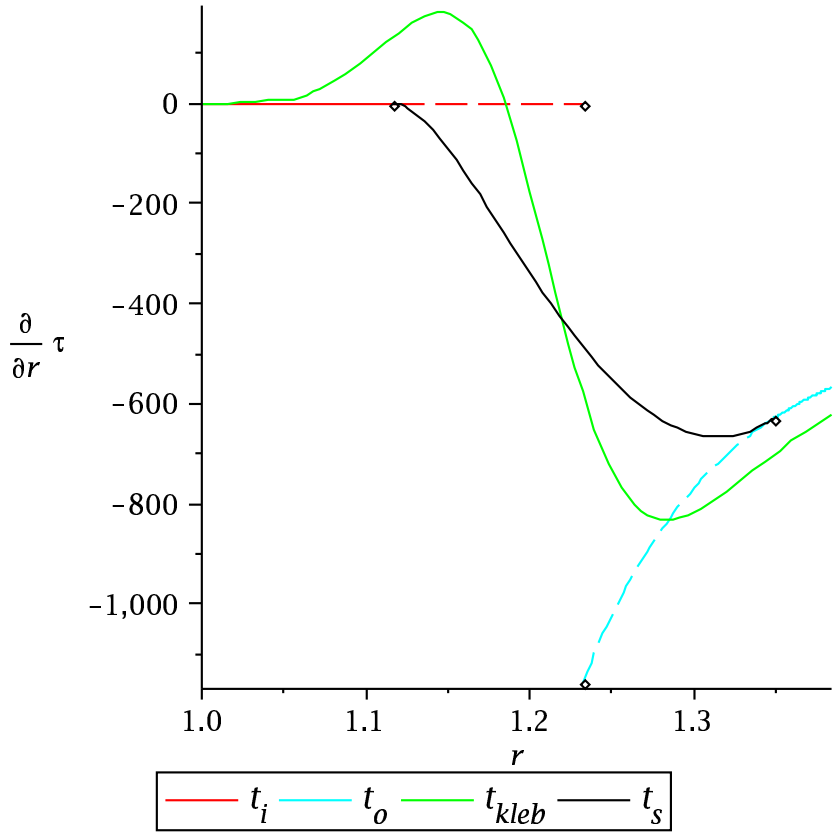

Figure 5.21

Radial component of the MMS source term for momentum due to viscous effects

The prescribed solution functions, consisting of the density and temperature profiles from the supersonic vortex case and the modified velocity and viscosity profiles given above, are substituted into the original governing equations (3.20) and the manufactured source terms (3.67) are obtained. These source terms are added to the numerical source terms already present in the solver (3.68). Under the influence of these source terms the solver will evolve the solution towards the prescribed solution. Once the solver has converged, error norms are computed (4.2) which are then utilized to determine the observed order of accuracy of the scheme (3.66).

The High-Reynolds Number Turbulent Boundary Layer (HRNTBL) test case has been run at four different Reynolds numbers ($Re \in \{5 \cdot 10^5, 1 \cdot 10^6, 2 \cdot 10^6, 4 \cdot 10^6\}$) on seven different mesh configurations (C2, C3, C4, C5, C6, H2, H3). The mesh configurations denoted by Cn utilize n^{th}-order Bezier volumes which have been generated by a Lagrange interpolation through n points in each parametric direction. The configurations denoted by Hn make use of a Hermite interpolation procedure and produce $2n^{th}$-order Bezier mesh elements. These interpolation algorithms are described in more detail in Sections 3.2.1.2 and 3.2.1.3.

For each Reynolds number case, an initial coarse mesh is generated such that the location of the first point off the wall is $y^+ \approx 5$. An exponential stretching function (5.33) is then used in the radial direction to relax the point spacing away from the wall, such that the mesh is nearly isotropic near the outer boundary.

$$r(z) = r_i + \frac{e^{\alpha z} - 1}{e^{\alpha} - 1}(r_o - r_i), \qquad 0 \leq z \leq 1 \tag{5.33}$$

From the initial coarse mesh, a series of self-similar meshes are generated in which the number of elements in each direction is doubled for each additional level of refinement. The point spacing in the radial direction is obtained by splitting the uniform point distribution in parametric space (z) prior to applying the stretching function (5.33). In this way, the rate of growth for the elements remains smooth.

Table 5.1 lists the resolutions of the coarsest meshes, the exponential stretching factors used in Equation 5.33, and the minimum and maximum element aspect ratios.

Table 5.1

Parameters for the coarsest mesh in each HRNTBL test case

Re	$N_\theta \times N_r$	α	$\min(h_\theta/h_r)$	$\max(h_\theta/h_r)$
$5 \cdot 10^5$	5×11	2.0	2.18516	25.4179
$1 \cdot 10^6$	5×11	3.2	1.66581	58.3019
$2 \cdot 10^6$	5×15	4.0	1.86636	143.754
$4 \cdot 10^6$	5×34	4.0	2.18516	328.881

In the next section, the L_1-errors norms for density and x-momentum are presented. The data is plotted for each mesh configuration over a series of refined meshes. The data shows the L_1-error norms ($\mathcal{E}_1$) on the y-axis versus the inverse of the mesh spacing ($1/h$) on the x-axis. The solid lines on the plots are provided to illustrate the slopes which correspond to second, third, and forth-order rates of convergence. The raw data for all four conservative variables for each set of runs are given in Appendix B.

5.5 Results for the Turbulent Boundary Layer Cases

As with the supersonic vortex case, the data indicates that computing second-order accurate solutions may be obtained with any of the mesh configurations; however, there is a noticeable improvement in the computed error norms when curved meshes (C3, C4, C5, C6, H2, H3) are employed. There is no additional improvement in error norms when greater than third-order elements are used. (See Figures 5.22, 5.23, 5.24, 5.25, 5.26, 5.27, 5.28, and 5.29)

When the solver is run with a third-order solution approximations, the linear mesh elements (C2) still show only second-order convergence of the error norms. Third-order convergence may be obtained with quadratic or higher mesh elements. Unlike the supersonic vortex, however, there now appears to be a noticeable difference in error norms computed on the third-order geometry (C3) when compared to the higher-order mesh elements (C4, C5, C6, H2, H3), particularly on the coarser meshes. The errors norms computed for the higher-order mesh shapes (C4, C5, C6, H2, H3) are effectively the same. (See Figures 5.30, 5.31, 5.32, 5.33, 5.34, 5.35, 5.36, and 5.37)

When fourth-order solution approximations are used, it becomes very difficult for the solver to arrive at a converged solution on meshes with linear elements (C2). When solutions are obtained on the linear meshes, the convergence rate is still only second-order. With quadratic elements, the error norms in density and energy are converging at a fourth-order rate; however, the error norms for momentum appear to only be third-order accurate. Fourth-order convergence is only obtained when the solver runs on cubic meshes (C4, H2) or better (C5, C6, H3). It is worth noting that there is a slight improvement in error norms

on meshes with greater than fourth-order elements (C5, C6, H3), but once again, we see that there is no further improvement in error norms once the geometry is represented at one order higher than the solution. (See Figures 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.44, and 5.45)

In general, it seems that the Hermite cubic meshes (H2) seem to generate slightly greater error norms than the Lagrange cubic meshes (C4). This is most likely due to the fact that the Lagrange interpolated elements pass through more points on the actual boundary surface for a given order of accuracy. Thus, the Lagrange elements are more likely to conform more closely to the boundary than the Hermite interpolated geometry which only passes through half as many points on the boundary. Although, there does not appear to be any noticeable difference in errors generated from the sixth-order interpolations, experience with the cubic elements suggests that a sixth-order solution may reveal a preference for the $C6$ elements over the $H3$ type. (See again Figures 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.44, and 5.45)

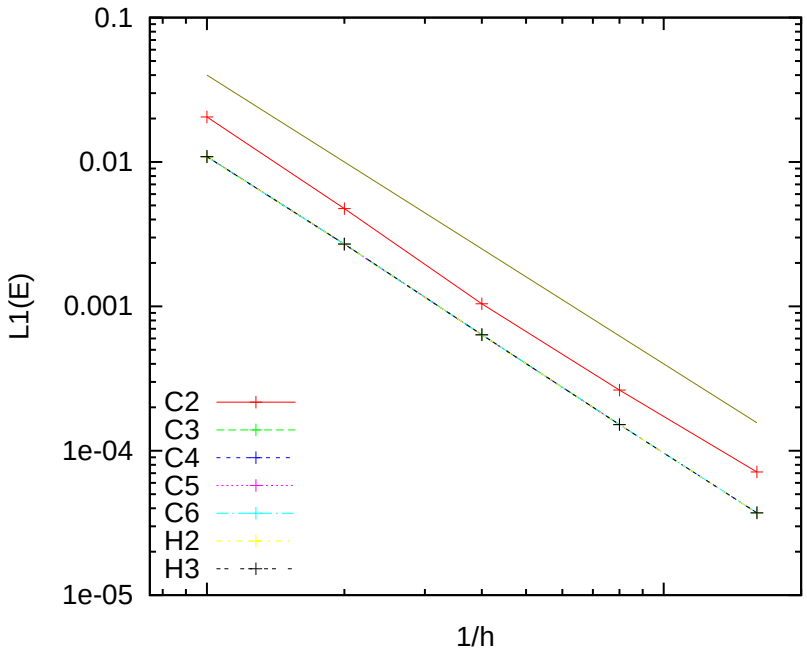

Figure 5.22

L_1-errors in ρ for a 2^{nd}-order solution at $Re = 5e5$

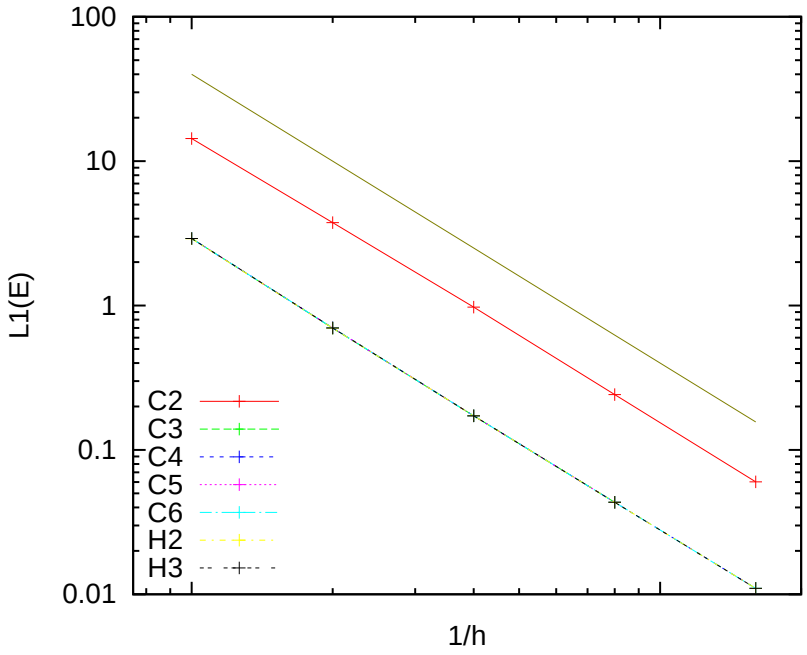

Figure 5.23

L_1-errors in ρu for a 2^{nd}-order solution at $Re = 5e5$

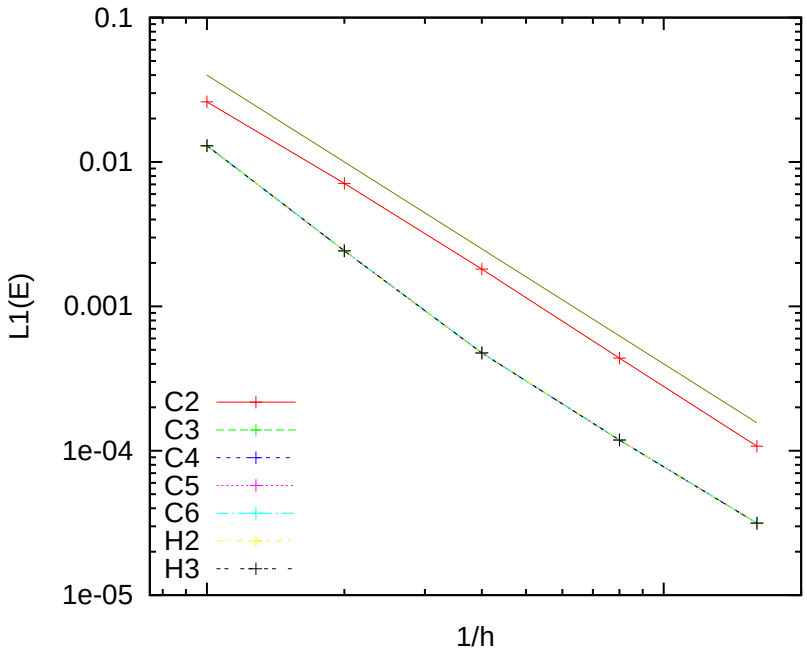

Figure 5.24

L_1-errors in ρ for a 2^{nd}-order solution at $Re = 1e6$

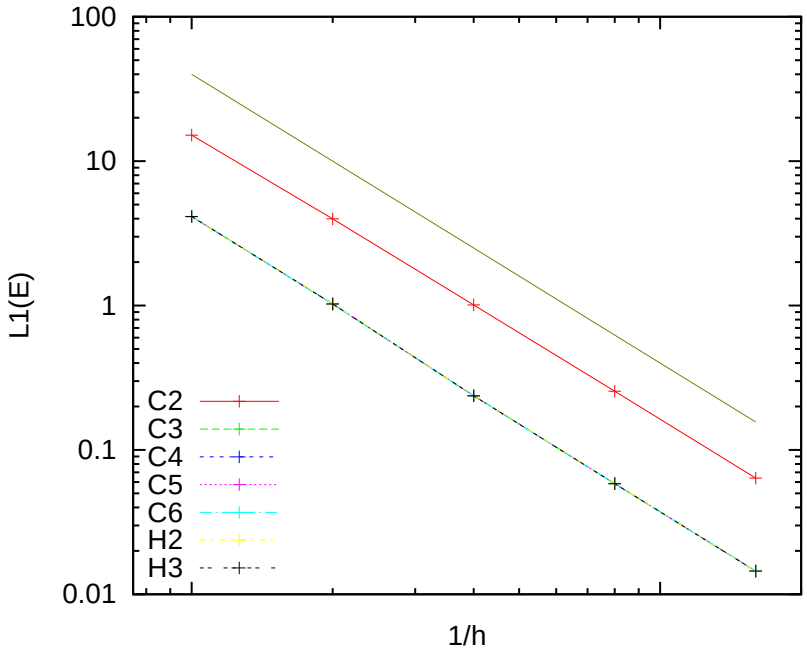

Figure 5.25

L_1-errors in ρu for a 2^{nd}-order solution at $Re = 1e6$

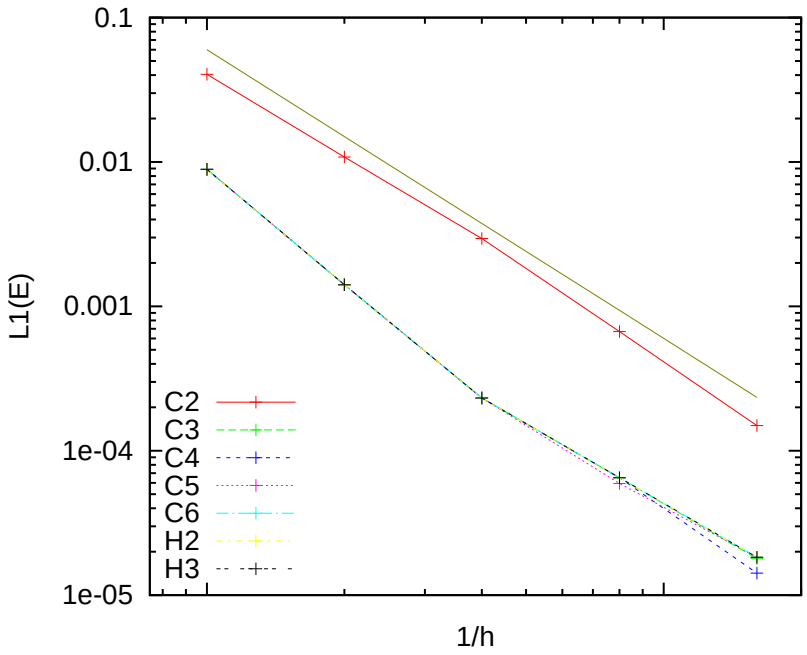

Figure 5.26

L_1-errors in ρ for a 2^{nd}-order solution at $Re = 2e6$

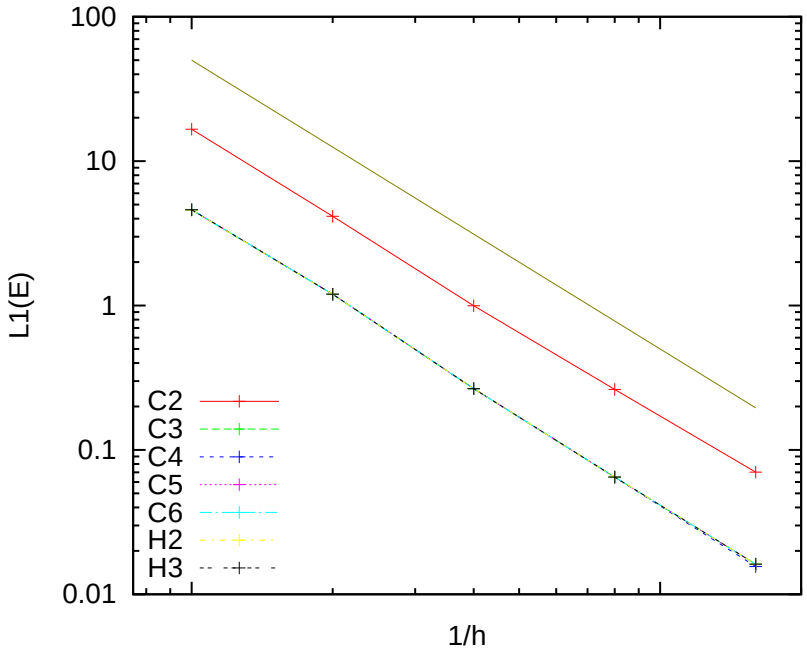

Figure 5.27

L_1-errors in ρu for a 2^{nd}-order solution at $Re = 2e6$

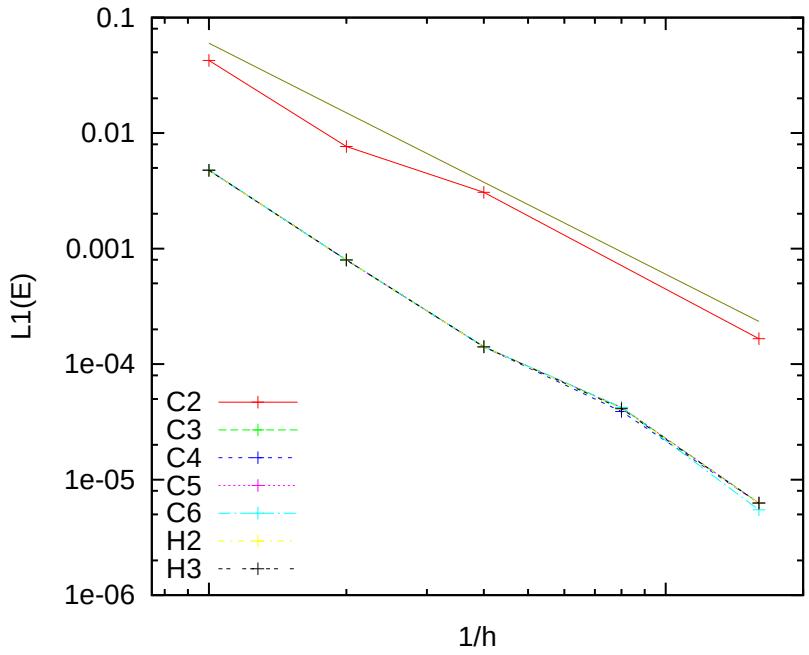

Figure 5.28

L_1-errors in ρ for a 2^{nd}-order solution at $Re = 4e6$

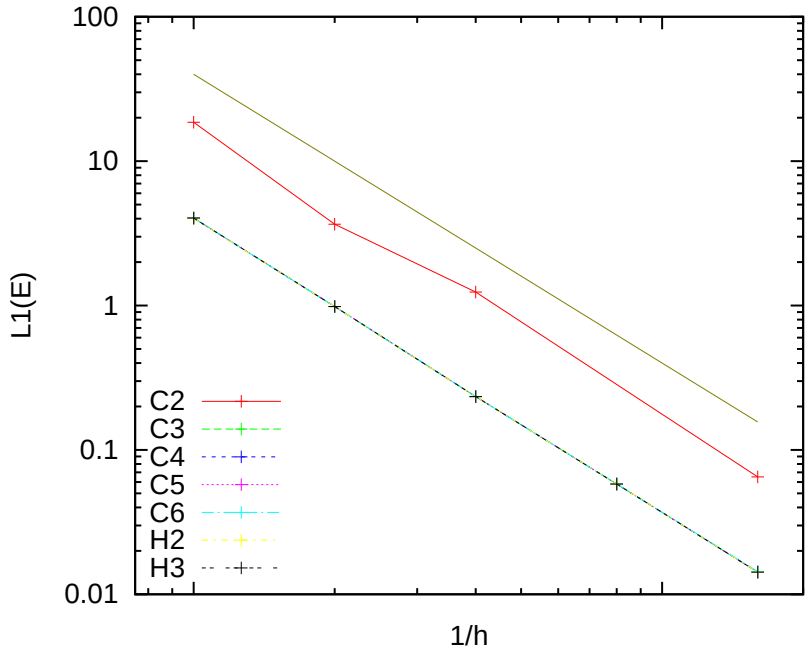

Figure 5.29

L_1-errors in ρu for a 2^{nd}-order solution at $Re = 4e6$

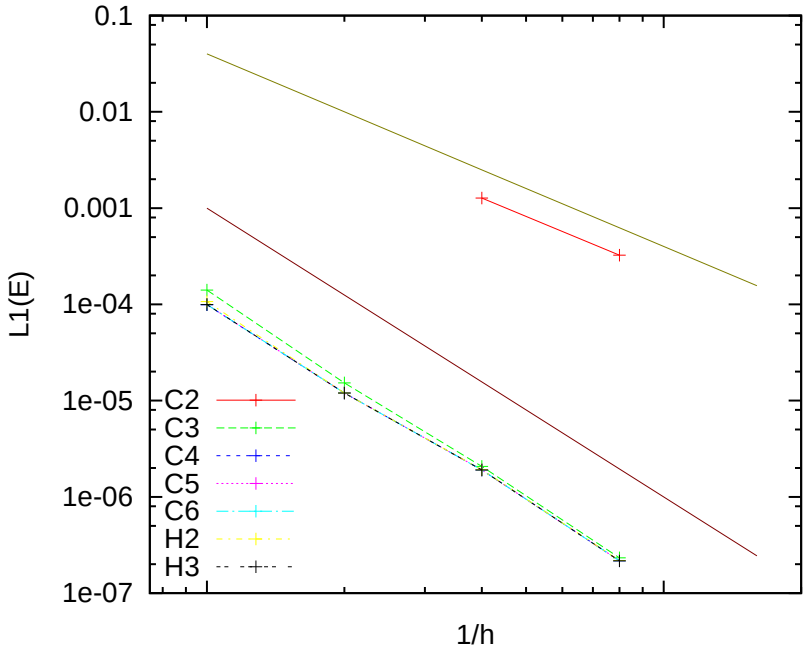

Figure 5.30

L_1-errors in ρ for a 3^{rd}-order solution at $Re = 5e5$

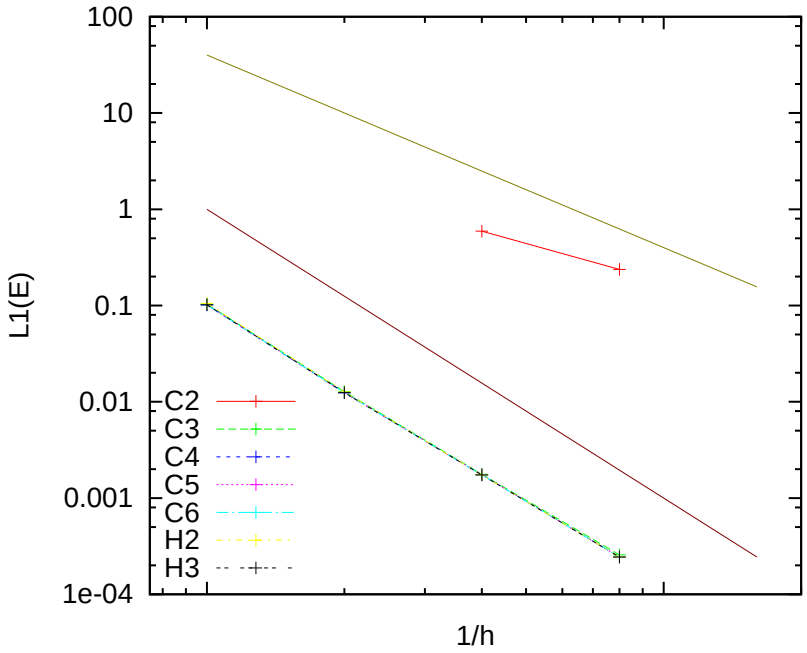

Figure 5.31

L_1-errors in ρu for a 3^{rd}-order solution at $Re = 5e5$

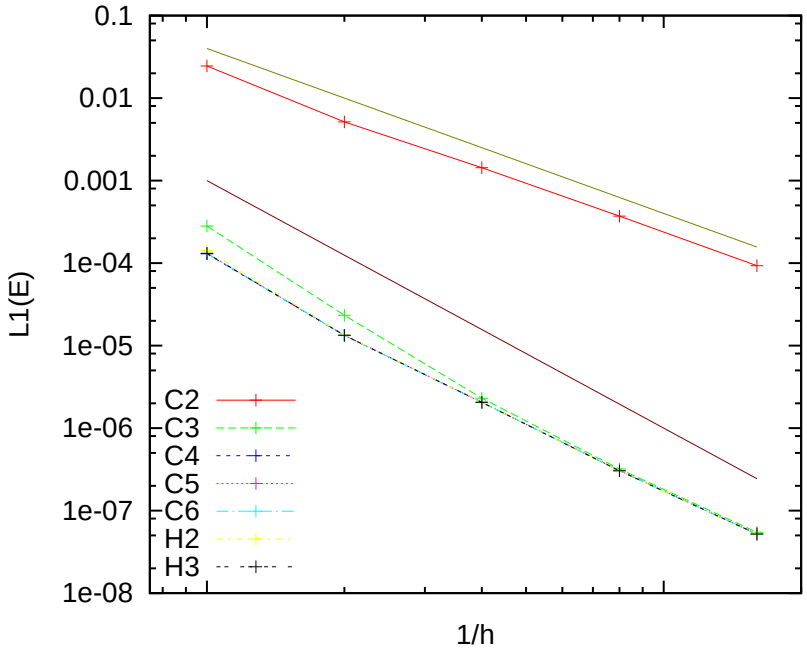

Figure 5.32

L_1-errors in ρ for a 3^{rd}-order solution at $Re = 1e6$

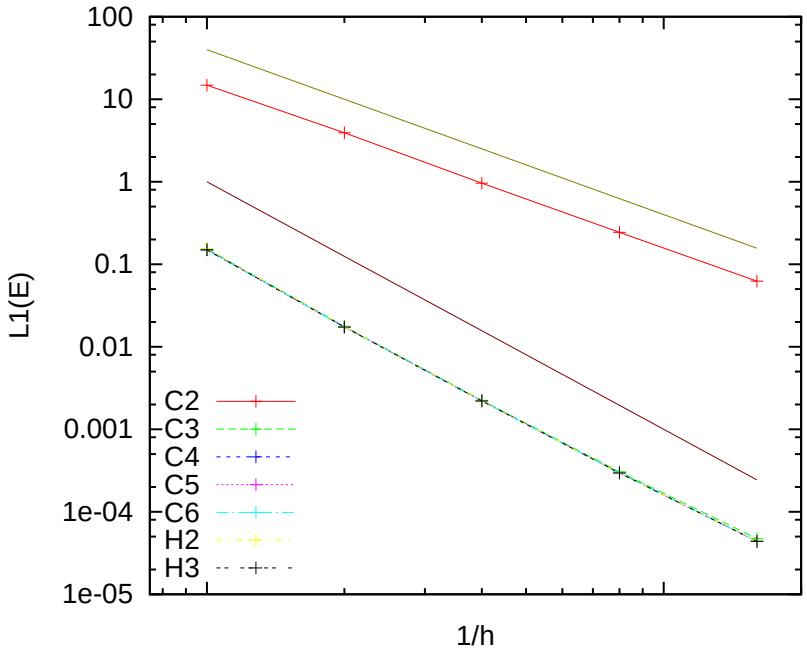

Figure 5.33

L_1-errors in ρu for a 3^{rd}-order solution at $Re = 1e6$

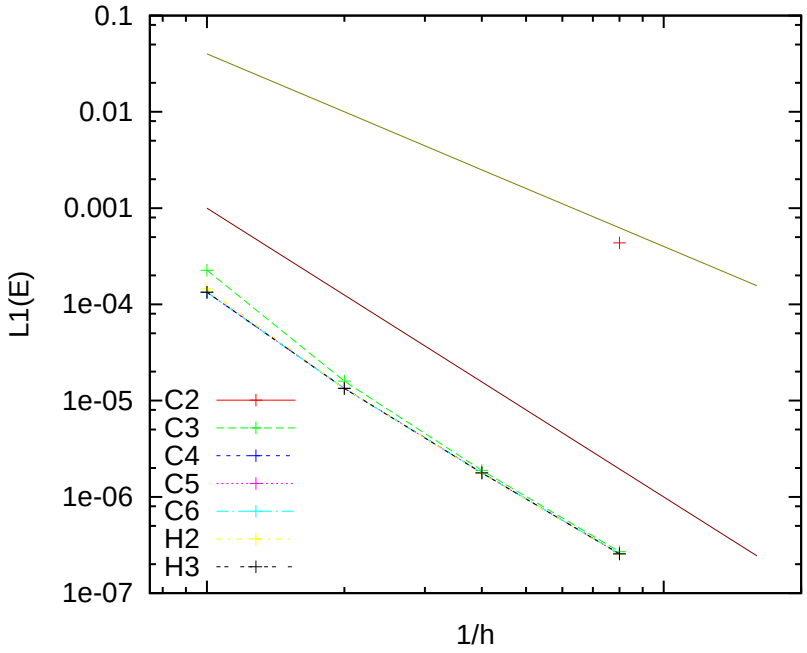

Figure 5.34

L_1-errors in ρ for a 3^{rd}-order solution at $Re = 2e6$

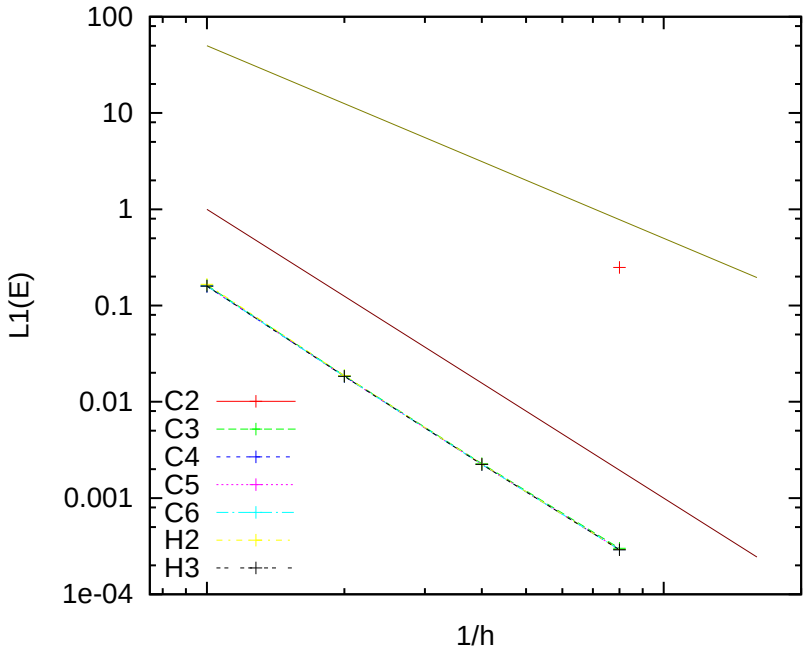

Figure 5.35

L_1-errors in ρu for a 3^{rd}-order solution at $Re = 2e6$

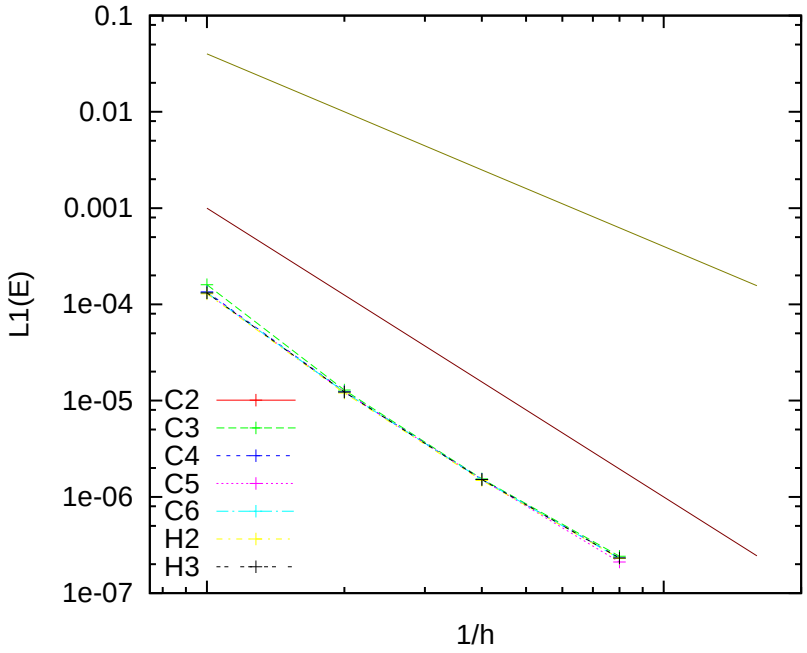

Figure 5.36

L_1-errors in ρ for a 3^{rd}-order solution at $Re = 4e6$

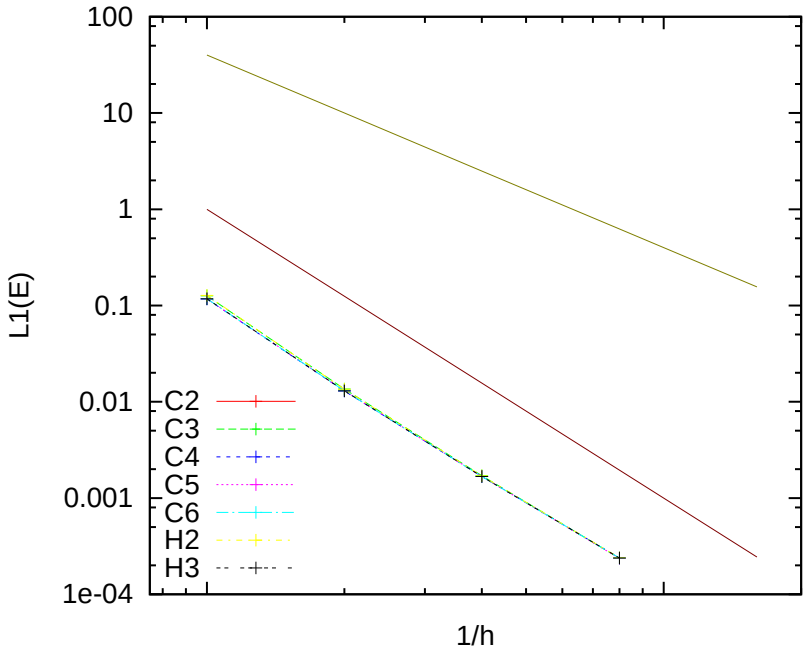

Figure 5.37

L_1-errors in ρu for a 3^{rd}-order solution at $Re = 4e6$

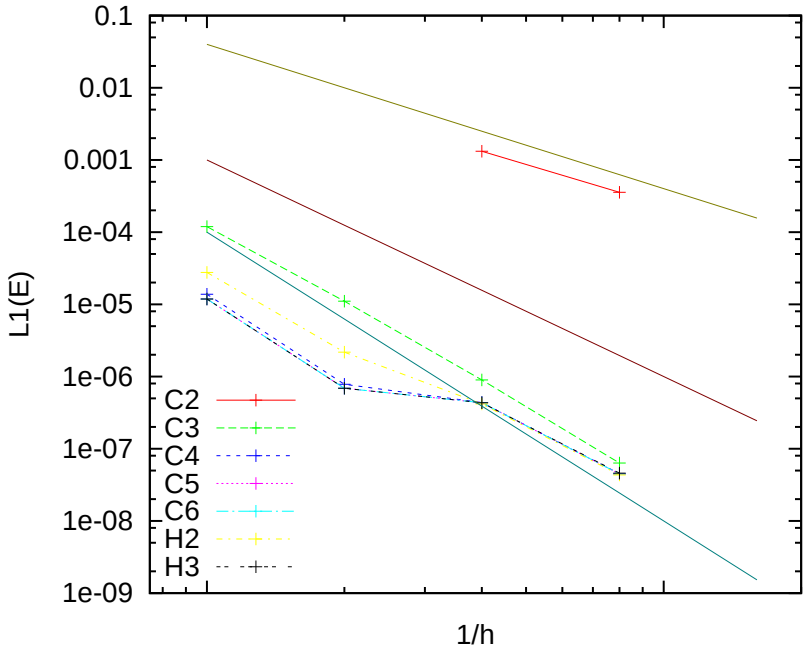

Figure 5.38

L_1-errors in ρ for a 4^{th}-order solution at $Re = 5e5$

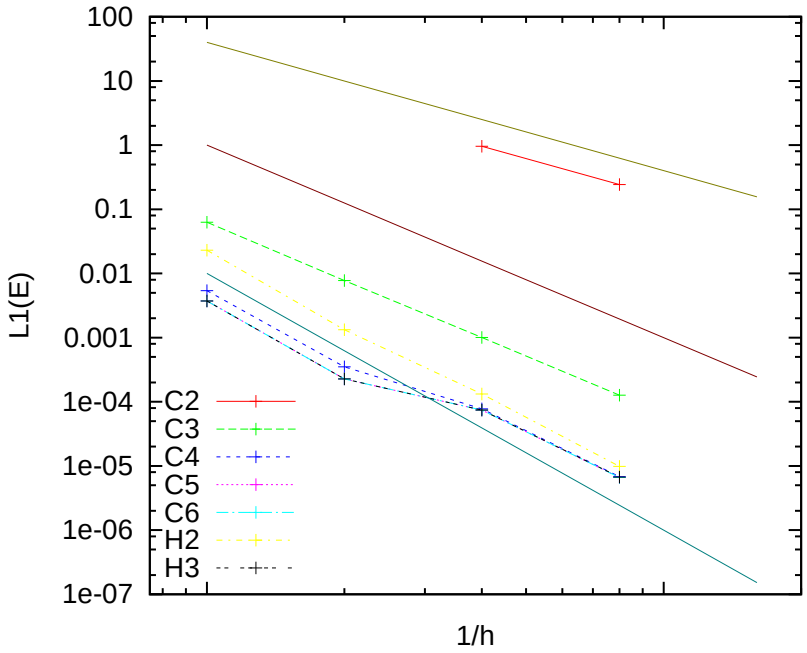

Figure 5.39

L_1-errors in ρu for a 4^{th}-order solution at $Re = 5e5$

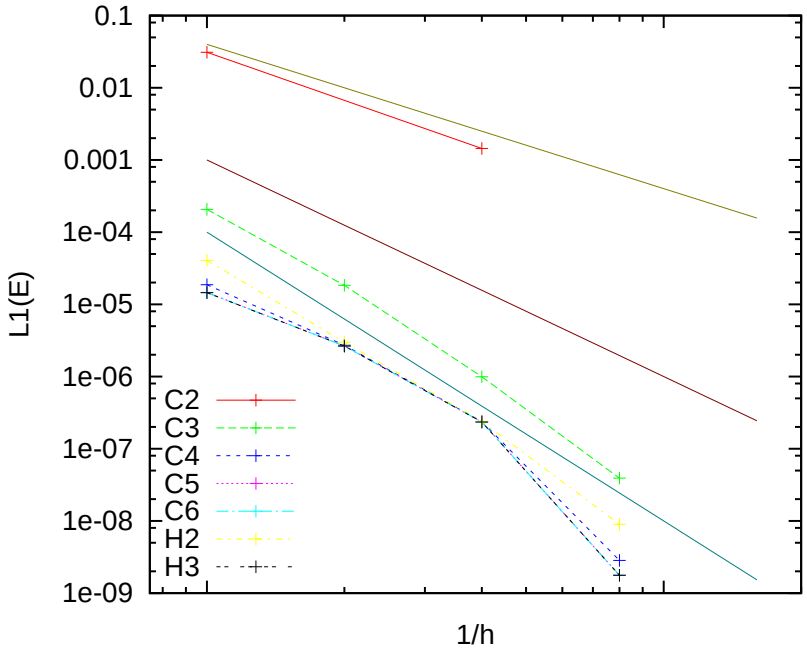

Figure 5.40

L_1-errors in ρ for a 4^{th}-order solution at $Re = 1e6$

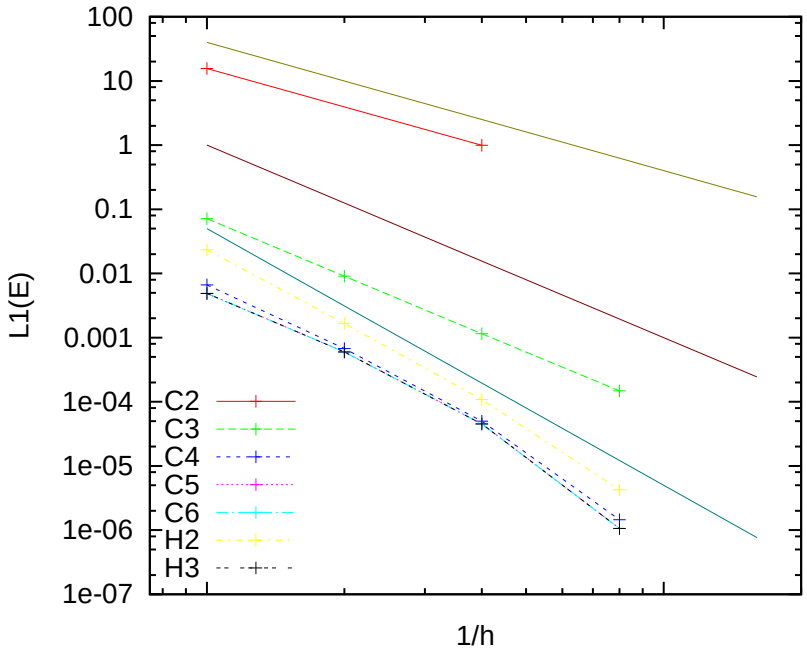

Figure 5.41

L_1-errors in ρu for a 4^{th}-order solution at $Re = 1e6$

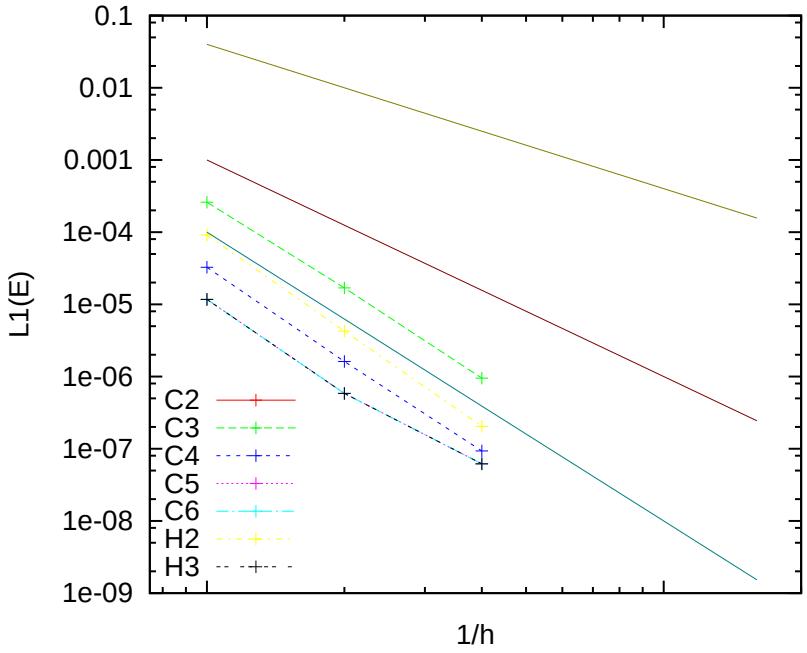

Figure 5.42

L_1-errors in ρ for a 4^{th}-order solution at $Re = 2e6$

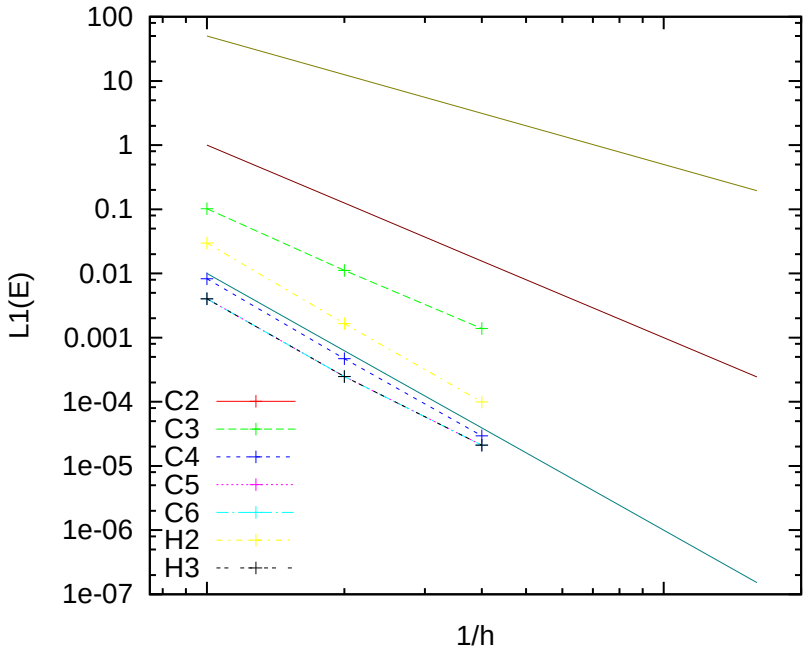

Figure 5.43

L_1-errors in ρu for a 4^{th}-order solution at $Re = 2e6$

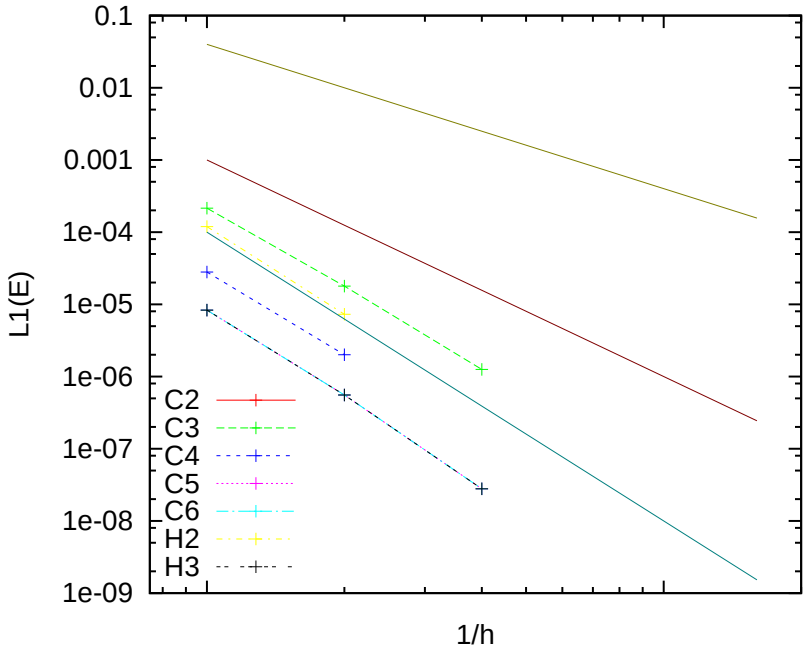

Figure 5.44

L_1-errors in ρ for a 4^{th}-order solution at $Re = 4e6$

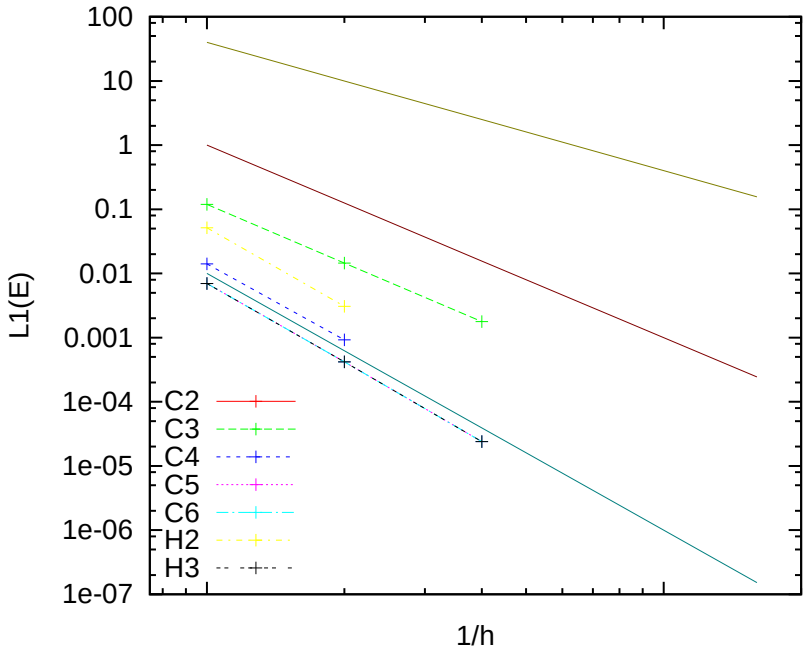

Figure 5.45

L_1-errors in ρu for a 4^{th}-order solution at $Re = 4e6$

5.6 Matrix Condition

The condition number for the scaled system matrices for each of the HRNTBL cases have been estimated and are plotted on the following pages. Since these are only estimates of the condition number, one can expect a small amount of variance from the actual values.

In general, the condition number tends to increase as the mesh is refined, although for the $Re = 5e5$ case, there opposite appears to be true. The most noticeable trend shown in these plots is the difference in condition number estimates for linear elements versus curved elements, particularly on the coarser meshes. As the meshes are refined, however, the condition number estimates for all element types tend towards the same values. Among the curved elements, there appears to be very little difference in condition number for a given mesh resolution. Of the significant variances which do exist, there appears to be one particular mesh for which the condition numbers diverge. The source of these deviations and their significance have not yet been determined.

From the data provided, it would appear that poor matrix conditioning is not a serious obstacle for these types of problems. While preparing these test cases to run, a variety of mesh resolutions and distributions were evaluated to find specific configurations on which converged solutions could be obtained. Based on that experience, and the data presented here, it appears as though the condition number estimate is more strongly affected by the mesh resolution than the specific shapes of the elements. In fact, for a sufficiently refined mesh, the order of the mesh element shapes appears to have almost no affect on the condition number estimate.

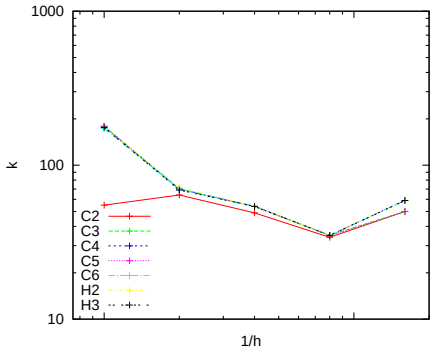

Figure 5.46

Condition numbers for 2^{nd}-order HRNTBL solutions at $Re = 5e5$

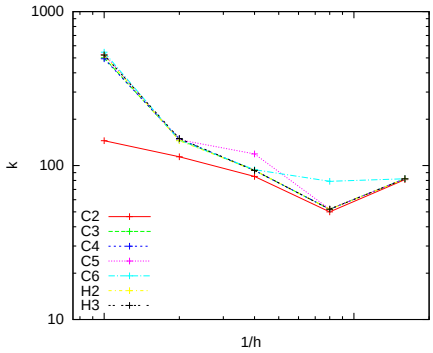

Figure 5.47

Condition numbers for 3^{rd}-order HRNTBL solutions at $Re = 5e5$

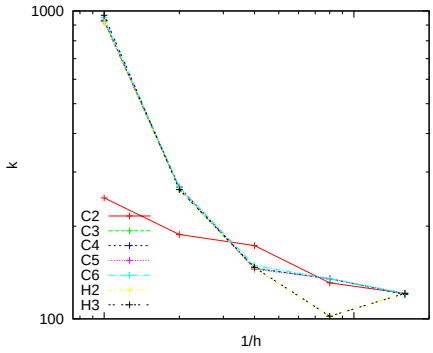

Figure 5.48

Condition numbers for 4^{th}-order HRNTBL solutions at $Re = 5e5$

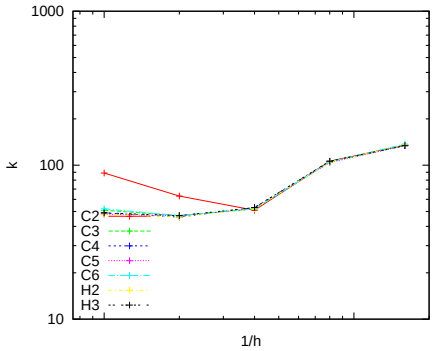

Figure 5.49

Condition numbers for 2^{nd}-order HRNTBL solutions at $Re = 1e6$

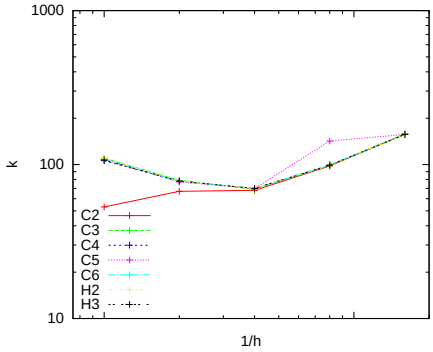

Figure 5.50

Condition numbers for 3^{rd}-order HRNTBL solutions at $Re = 1e6$

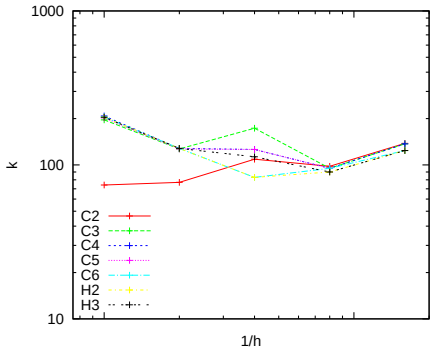

Figure 5.51

Condition numbers for 4^{th}-order HRNTBL solutions at $Re = 1e6$

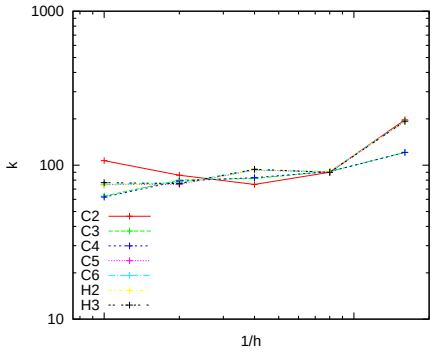

Figure 5.52

Condition numbers for 2^{nd}-order HRNTBL solutions at $Re = 2e6$

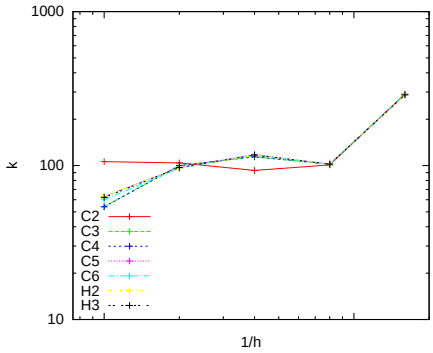

Figure 5.53

Condition numbers for 3^{rd}-order HRNTBL solutions at $Re = 2e6$

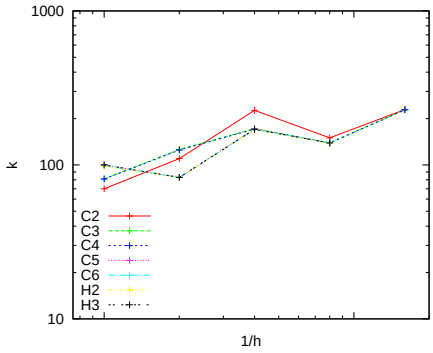

Figure 5.54

Condition numbers for 4^{th}-order HRNTBL solutions at $Re = 2e6$

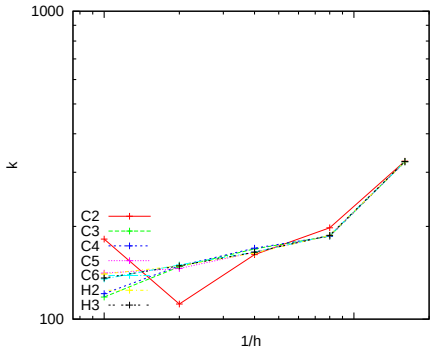

Figure 5.55

Condition numbers for 2^{nd}-order HRNTBL solutions at $Re = 4e6$

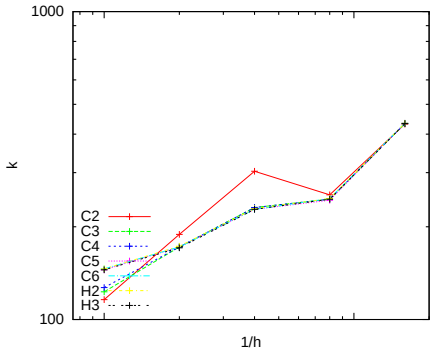

Figure 5.56

Condition numbers for 3^{rd}-order HRNTBL solutions at $Re = 4e6$

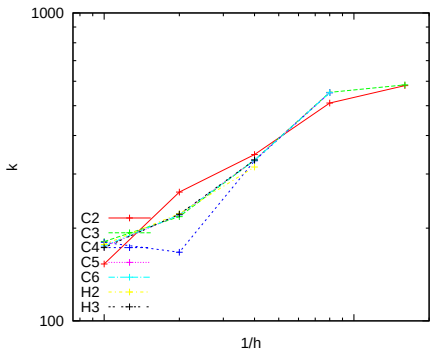

Figure 5.57

Condition numbers for 4^{th}-order HRNTBL solutions at $Re = 4e6$

5.7 Discussion of Numerical Results

Based on the results of these numerical experiments, there are a number of useful observations may be made with regard to mesh quality considerations for the discontinuous Galerkin method. With respect to the minimum criteria required to maintain the accuracy of the scheme:

- Linear elements should not be used to compute solutions when greater than second-order accuracy is desired,
- For inviscid simulations, quadratic and cubic elements may be used to obtain solutions that are up to fourth-order accurate,
- For viscous simulations, the mesh elements used near no-slip viscous boundary conditions must be iso-parametric or super-parametric,
- Boundary surfaces patches must be at least C^0-continuous at element interfaces, but C^1-continuity does not appear to be required,

The preponderance of numerical evidence presented both here and in prior published works, make it abundantly clear that use of linear mesh elements to resolve smoothly curving boundaries will result in severely degraded solution accuracy. The data from the supersonic vortex test and the turbulent boundary layer cases confirm that the DG solver is restricted to second-order accuracy, regardless of the order of the polynomial solution approximation. This trend appears to hold even when these boundary surfaces are covered with very fine meshes.

Using curved element shapes of quadratic or higher order appears to be sufficient to allow the solver to obtain first through fourth-order accurate solutions of inviscid flows. However, for fifth-order accurate solutions, both the quadratic and cubic elements were insufficient. This suggests that there may be some even-odd coupling between the orders

of the mesh and solution approximations when solving systems of hyperbolic conservation laws.

When computing convection-diffusion solutions, the use of iso-parametric elements is required to maintain the desired order of accuracy of the scheme. Slight improvements in solution errors have been observed when using mesh elements of at least one order higher than the desired solution accuracy. The improvement is most noticeable on coarser meshes. This suggests that it may be possible to trade mesh resolution for accuracy of surface approximation so long as the mesh remains fine enough to accurately resolve relevant flow features near the boundary.

From these numerical tests, it appears that C^1-continuity of the mesh elements near the boundary does not produce any significant benefits over C^0-continuity. In fact, the data provided in this report seems to indicate that the error norms are slightly worse for the C^1-continuous boundary representations. Thus, for a given polynomial order of mesh element, the data suggests that it is more appropriate to minimize surface displacement error than attempt to maintain the continuity of surface normal directions at element interfaces. This is an encouraging result that further demonstrates the compact nature of the DG scheme.

The stiffness of the solver, as indicated by the condition number estimates obtained for the scaled system matrix, does not appear to be related to the order of the mesh element shapes. There are some indications that the condition number may be more strongly influenced by the resolution and distribution of the mesh elements.

CHAPTER 6

CONCLUDING REMARKS

The current study has taken the position that mesh quality is as much dependent upon the numerical methods and algorithms employed by the solver as it is on the physical characteristics of the domain and the flow under consideration. In practice, it is often impractical to obtain a rigorous theoretical analysis for a particular solver implementation. Even when available, such analysis does not always give a complete picture of how the solver will perform on complex flow problems. In these situations, the only practical solution is to empirically evaluate the behavior of the solver under realistic usage conditions.

There are a number of existing verification methods which can be utilized to assess the numerical accuracy of a given solver implementation. Although these methods provide very little insight into the exact nature of the errors generated by the solver, they are invaluable for establishing confidence in the correct implementation of the solver algorithm. These verification methods may also be used to assess the numerical accuracy of a solver under conditions which approximate very specific flow regimes. This is a critical capability which allows for highly accurate characterization of the numerical errors produced by the solver in regions that have traditionally been very difficult to analyze with standard theoretical techniques.

In the present work, we have taken these new capabilities one step further by utilizing the increased error sensitivity to investigate of the issue of mesh quality as it pertains to a high-order accurate discontinuous Galerkin solver implementation. By systematically varying the mesh approximation space and then examining the numerical errors and condition number estimates, relationships between the accuracy and stiffness of the solver to various mesh properties have been deduced. The resulting data has provided some useful insights into the meshing strategies which best preserve the theoretical accuracy of the method and also yield the most accurate solutions.

These analysis techniques should be generally applicable to a wide variety of numerical solvers and mesh configurations. The only requisite condition is that there exist a well defined mathematical description of the flow field in the region under consideration. This description may be obtained from exact known solutions, nearby splined approximations to previously computed solutions, or even empirically and theoretically derived flow models.

6.1 Summary of Results

When considering problems involving smoothly curving boundaries, we have confirmed that the use of linear mesh elements near curved boundaries will prevent the solver from attaining any higher than second order error convergence, regardless of the polynomial order of the solution approximation. The use of curved elements permits a significant reduction in computed error norms, even in second-order accurate solutions.

When solving the inviscid equations, quadratic and cubic elements may be used to compute up to fourth-order accurate solutions. For viscous flows, however, it was observed that the error convergence rate was limited to the same order of accuracy as the mesh element approximations to the domain boundaries. This result confirms the standard practice in the finite element community of utilizing iso-parametric mesh elements to discretize the domain.

The iso-parametric condition is only a minimum criteria required to maintain the theoretical order of accuracy of the scheme. The data from our experiments indicate that increasing the polynomial order of the boundary elements to at least one order higher than the solution approximation typically results in a noticeable decrease in the computed error norms. This result was most dramatic for second order solutions where the computed error decreased by up to an order of magnitude when curved elements were employed. It is further noted that increasing the polynomial order of the mesh elements beyond one greater than the solution approximation did not produce any further reduction in the error norms.

With regards to the specific interpolation techniques used to generate the curved mesh element shapes, our results indicate that there is no requirement for the surface patches to maintain continuous normal directions between elements. In fact, the data from this investigation indicates that enforcing C^1-continuity at the element interfaces actually results in slightly higher computed error norms than that which is produced by the same order element generated with only C^0-continuity.

We therefore conclude that the degrees of freedom for the mesh element should be optimized for minimizing the surface displacement error within each element rather than

attempting to preserve continuity of the surface normal direction at the element interfaces. This is a significant result because it indicates that the only coupling between adjacent mesh elements is the C^0-continuity constraint which is required to form a valid partition of space. The implications of this finding is that it should be much easier to generate a satisfactory curved element mesh than is currently believed. In particular, the current practice of generating cubic surface patches from positions and normals at the mesh nodes may neither be required nor optimal.

6.2 Future Work

One of the conclusions of this work was that requiring C^1-continuity along the boundary at element interfaces resulted in slightly worse error norms than those which required only C^0-continuity. Although we concluded that the smoothness condition along the edge between two elements on the boundary is not a requirement for generating a quality mesh, there is one particular solver implementation detail which may influence this result. In the current implementation, the numerical quadrature integrations are computed using the Gauss-Legendre points. Since this set of points does not include the end-points of the parametric interval, $[-1, 1]$, the solver may not be actually sampling the geometry along the edges of the boundary surface patches. It is possible that if the solver were to use an alternative quadrature rule that does include the end-points (e.g. Gauss-Lobatto), then the smoothness condition may play a more significant role that we claim here.

In this work, the curved meshes were generated by fitting all of the mesh element shapes to sections of cylindrical coordinate system. In other words, all of the elements

throughout the domain were curved in such a way that their interfaces were interpolating surfaces of constant cylindrical coordinates (i.e. r, θ, z). This was done to avoid some of the mesh generation difficulties associated with grid line overlapping which would likely occur if only the boundary surfaces of the anisotropic mesh elements were curved. A more robust solver would provide facilities for propagating the surface curvature further into the domain if necessary. There are a number of ways in which this could be accomplished, several of which already appear in the literature [42, 43, 55, 59].

There are a number of ways in which the turbulent boundary layer profile (described in Section 5.4) may be improved to make it more nearby. Most notably, the thermodynamic state of the fluid, described by density, pressure, and temperature, were taken directly from the supersonic vortex case, which assumed isentropic conditions. Obviously, viscous flows are not isentropic. So, it is likely that a more realistic thermodynamic state for the fluid that can be derived. The velocity profile for the turbulent boundary layer is primarily derived from the law-of-the-wall relations for a flat plate. A better flow model would more accurately account for the curvature of the domain and the compressibility of the fluid, possibly even allowing for the growth of the thickness of the boundary layer. This latter enhancement would need to be properly correlated with the pressure and density profiles which would be derived from the thermodynamic equations of state.

The domains for the test cases given in this report are much less complex than practical engineering domains. This was done to reduce the number of complicating factors in the experimental analysis. However, the results presented in this report could benefit from additional test cases involving more complex geometric bodies. Surfaces with non-constant

curvature would be a logical next step as would the extension to three-dimensional flow fields.

The methodology developed for the current work should serve as a useful framework for those wishing to conduct similar or even more ambitions investigations. These techniques should allow the investigator to explore a variety of circumstances in which the ability of the solver to effectively compute an accurate solution may be dependent upon the manner in which the domain has been discretized. The turbulent boundary layer profile described in Section 5.4 is just one example of how these methods may be used.

REFERENCES

[1] M. Aftosmis, D. Gaitonde, and T. S. Tavares, *The behavior of linear reconstruction techniques on unstructured meshes*, Tech. Rep. WL-TR-94-3023, Wright Laboratory, Wright-Patterson AFB, February 1994.

[2] F. Bassi, A. Crivellini, D. A. Di Pietro, and S. Rebay, "An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows," *Computers and Fluids*, vol. 36, no. 10, December 2007, pp. 1529–1546.

[3] F. Bassi, A. Crivellini, A. Ghidoni, and S. Rebay, "High-order discontinuous Galerkin discretization of transonic turbulent flows," *47th AIAA Aerospace Sciences Meeting*, Orlando, FL, January 2009, AIAA, p. 15, AIAA, AIAA 2009-180.

[4] F. Bassi, A. Crivellini, L. Pelagalli, S. Rebay, and M. Savini, "A parallel high-order discontinuous Galerkin solver applied to complex three-dimensional turbulent flows," *European Conference on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2004*, Jyväskylä, July 2004, p. 14.

[5] F. Bassi, A. Crivellini, D. D. Pietro, and S. Rebay, "A high-order discontinuous Galerkin solver for 3D aerodynamic turbulent flows," *European Conference on Computational Fluid Dynamics, ECCOMAS CFD*, TU Delft, The Netherlands, 2006, p. 20.

[6] F. Bassi, A. Crivellini, S. Rebay, and M. Savini, "Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-ω turbulence model equations," *Computers and Fluids*, vol. 34, 2005, pp. 507–540.

[7] F. Bassi and S. Rebay, "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.

[8] F. Bassi and S. Rebay, "High-order accurate discontinuous finite element solution of the 2D Euler equations," *Journal of Computational Physics*, vol. 138, 1997, pp. 251–285.

[9] F. Bassi and S. Rebay, "A high order discontinuous Galerkin method for compressible turbulent flows," In Cockburn et al. [21], pp. 77–88, ISBN: 3540667873.

[10] F. Bassi and S. Rebay, "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 40, 2002, pp. 197–207, DOI: 10.1002/fld.338.

[11] C. E. Baumann and J. T. Oden, "A discontinuous hp finite element method for the Euler and Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 31, 1999, pp. 79–95.

[12] Y. Bazilevs, V. Calo, J. Cottrell, T. Hughes, A. Reali, and G. Scovazzi, "Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 1-4, 2007, pp. 173 – 201.

[13] R. M. Beam and R. Warming, "An implicit factored scheme for the compressible Navier-Stokes equations," *AIAA Journal*, vol. 16, no. 4, April 1978, pp. 393–402.

[14] R. B. Bond, P. M. Knupp, and C. C. Ober, "A manufactured solution for verifying CFD boundary conditions," *34th AIAA Fluid Dynamics Conference and Exhibit*, Portland, OR, June 2004, AIAA, p. 10, AIAA, AIAA 2004-2629.

[15] R. B. Bond, P. M. Knupp, and C. C. Ober, "A manufactured solution for verifying CFD boundary conditions, part II," *43rd AIAA Aerospace Sciences Meeting*, Reno, NV, January 10-13 2005, AIAA, p. 22, AIAA, AIAA 2005-0088.

[16] R. B. Bond, C. C. Ober, and P. M. Knupp, "A manufactured solution for verifying CFD boundary conditions, part III," *36th AIAA Fluid Dynamics Conference and Exhibit*, San Francisco, CA, June 5-8 2006, AIAA, p. 17, AIAA, AIAA 2006-3722.

[17] R. B. Bond, C. C. Ober, P. M. Knupp, and S. W. Bova, "Manufactured solution for computational fluid dynamics boundary condition verification," *AIAA Journal*, vol. 45, no. 9, September 2007.

[18] J. Boris, F. Grinstein, E. Oran, and R. Kolbe, "New insights into large eddy simulation," *Fluid Dynamics Research*, vol. 10, no. 4-6, 1992, pp. 199–228.

[19] B. Cockburn, S. Hou, and C.-W. Shu, "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case," *Math. Comp.*, vol. 54, 1990, pp. 545–581.

[20] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, "The development of discontinuous Galerkin methods," [21], pp. 3–50, ISBN: 3540667873.

[21] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds., *Discontinuous Galerkin methods: theory, computation and applications*, vol. 11 of *Lecture Notes in Computational Science and Engineering*, Springer-Verlag, Berlin, 2000, ISBN: 3540667873.

[22] B. Cockburn, S. Lin, and C.-W. Shu, "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems," *Journal of Computational Physics*, vol. 84, 1989, pp. 90–113.

[23] B. Cockburn and C.-W. Shu, "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework," *Math. Comp.*, vol. 52, 1989, pp. 411–435.

[24] B. Cockburn and C.-W. Shu, *TheP^1-RKDG method for two-dimensional Euler equations of gas dynamics*, Tech. Rep. 91-32, ICASE, 1991.

[25] B. Cockburn and C.-W. Shu, "The Local Discontinuous Galerkin Method for Time-Dependent convection-Diffusion Systems," *SIAM Journal of Numerical Analysis*, vol. 35, no. 6, 1998, pp. 2440–2463.

[26] B. Cockburn and C.-W. Shu, "The Runge-Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems," *Journal of Computational Physics*, vol. 141, no. 2, 1998, pp. 199–224.

[27] S. S. Collis, "Monitoring unresolved scales in multiscale turbulence modeling," *Physics of Fluids*, vol. 13, no. 6, 2001, pp. 1800–1806.

[28] S. S. Collis, "Discontinuous Galerkin methods for turbulence simulation," *Proc. Summer Program 2002*. Center for Turbulence Research, 2002, pp. 155–167.

[29] S. S. Collis and K. Ghayour, "Discontinuous Galerkin methods for compressible DNS," *2003 ASME/JSME Joint Fluids Engineering Conference*, Honolulu, HA, July 6-10 2003, ASME/JSME, p. 10, ASME, FEDSM2003-45632.

[30] R. Cools, "An encyclopedia of cubature formulas," *Journal of Complexity*, vol. 19, no. 3, 2003, pp. 445 – 453, Oberwolfach Special Issue.

[31] J. Deardorff, "A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers," *Journal of Fluid Mechanics*, vol. 41, 1970, pp. 453–480.

[32] W. J. Gordon and C. A. Hall, "Transfinite element methods: blending-function interpolation over arbitrary curved element domains," *Numerical Methods in Mathematics*, vol. 21, 1973, pp. 109–129.

[33] R. Hansen and J. Forsythe, "Large and Detached Eddy Simulations of Flow Over a Circular Cylinder Using Unstructured Grids," *41st Aerospace Sciences Meeting*, Reno, NV, January 2003, AIAA, AIAA, AIAA 2003-0775.

[34] C. Hirsch, *Numerical computation of internal and external flows: Computational methods for inviscid and viscous flows*, vol. 2, Wiley-Interscience, 1990.

[35] L. Krivodonova and M. Berger, "High-order accurate implementation of solid wall boundary conditions in curved geometries," *Journal of Computational Physics*, vol. 211, 2006, pp. 492–512.

[36] B. Landmann, *A parallel discontinuous Galerkin code for the Navier-Stokes and Reynolds-averaged Navier-Stokes equations*, doctoral dissertation, University of Stuttgart, Stuttgart, Germany, December 2007.

[37] B. Landmann, M. Kessler, S. Wagner, and E. Krämer, "A parallel discontinuous Galerkin code for the Navier-Stokes equations," *44th AIAA Aerospace Sciences Meeting*, Reno, NV, January 2006, AIAA, p. 17, AIAA 2006-111.

[38] B. Landmann, M. Kessler, S. Wagner, and E. Krämer, "A parallel, high-order discontinuous Galerkin code for laminar and turbulent flows," *Computers & Fluids*, vol. 37, 2008, pp. 427–438.

[39] M. Lenoir, "Optimal isoparametric finite elements and error estimates for domains involving curved boundaries," *SIAM Journal of Numerical Analysis*, vol. 23, no. 3, June 1986.

[40] P. LeSaint and P. Raviart, *On a finite element method for solving the neutron transport equation*, Academic Press, New York, 1974.

[41] I. Lomtev and G. E. Karniadakis, "A discontinuous Galerkin method for the Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 29, 1999, pp. 587–603.

[42] C. Lübon, M. Kessler, S. Wagner, and E. Krämer, "High-order boundary discretization for discontinuous Galerkin codes," *24th Applied Aerodynamics Conference*, San Francisco, CA, June 5-8 2006, AIAA, p. 13, AIAA, AIAA 2006-111.

[43] C. Lübon and S. Wagner, "Three-dimensional discontinuous Galerkin codes to simulate viscous flow by spatial discretization of high order and curved elements on unstructured grids," *New results in numerical and experimental fluid mechanics VI; contributions to the 15th STAB/DGLR Symposium*, Darmstadt, Germany, 2007, STAB/DGLR, pp. 145–161, STAB/DGLR.

[44] E. A. Luke, *A Rule-Based Specification for Computational Fluid Dynamics*, doctoral dissertation, Mississippi State University, 1999.

[45] D. Mahajan, *Boundary-conforming discontinuous Galerkin methods via extensions from subdomains*, doctoral dissertation, University of Minnesota, Minneapolis, MN, December 2007.

[46] N. C. Nguyen, P.-O. Persson, and J. Peraire, "RANS Solutions Using High Order Discontinuous Galerkin Methods," *45th AIAA Aerospace Sciences Meeting*, Reno, NV, 2007, AIAA, p. 16, AIAA, 2007-914.

[47] R. Nichols and C. Nelson, "Application of hybrid RANS/LES turbulence models," *41st Aerospace Sciences Meeting*, Reno, NV, January 2003, AIAA, p. 16, AIAA, AIAA 2003-0083.

[48] J. T. Oden and C. E. Baumann, "A conservative DGM for convection-diffusion and Navier-Stokes problems," In Cockburn et al. [21], pp. 179–196, ISBN: 3540667873.

[49] T. A. Oliver, *A high-order, adaptive, discontinuous Galerkin finite element method for the Reynolds-averaged Navier-Stokes equations*, doctoral dissertation, Massachusetts Institute of Technology, Boston, MA, September 2008.

[50] S. Orsag and G. Patterson, "Numerical Simulation of Three-Dimensional Homogeneous Isotropic Turbulence," *Physical Review Letters*, vol. 28, 1972, pp. 76–79.

[51] J. Peraire and P.-O. Persson, "The compact discontinuous Galerkin (CDG) method for elliptic problems," *SIAM Journal of Scientific Computing*, vol. 30, no. 4, 2008, pp. 1806–1824.

[52] P.-O. Persson and J. Peraire, "Sub-Cell Shock Capturing for Discontinuous Galerkin Methods," *44th AIAA Aerospace Sciences Meeting*, Reno, NV, January 2006, AIAA, AIAA 2006-0112.

[53] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge University Press, 1992, ISBN 0521431085.

[54] W. Reed and T. Hill, *Triangular Mesh Methods for the Neutron Transport Equation*, Tech. Rep. LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[55] J.-F. Remacle, X. Li, M. S. Shephard, and J. E. Flaherty, "Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods," *International Journal for Numerical Methods in Engineering*, vol. 62, 2005, pp. 899–923.

[56] C. J. Roy, A. Raju, and M. M. Hopkins, "Estimation of Discretization Errors Using the Method of Nearby Problems," *AIAA Journal*, vol. 45, no. 6, June 2007, pp. 1232–1243.

[57] D. Ruiz, *A scaling algorithm to equilibrate both rows and columns norms in matrices*, Tech. Rep. Technical Report RT/APO/01/4, Institut National Polytechnique de Toulouse, Toulouse, 2001.

[58] K. Salari and P. Knupp, *Code verification by the method of manufactured solutions*, Technical Report SAND 2000-1444, Sandia National Labs, Albuquerque, NM, June 2000.

[59] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, J.-F. Remacle, M. W. Beall, and R. M. O'Bara, "Adaptive mesh generation for curved

domains," *Applied Numerical Mathematics*, vol. 52, no. 2-3 SPEC ISS, 2005, pp. 251–271.

[60] T. M. Smith, M. F. Barone, R. B. Bond, A. A. Lorber, and D. G. Baur, "Comparison of reconstruction techniques for unstructured mesh vertex centered finite volume schemes," *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, June 2007, AIAA, p. 22, AIAA 2007-3958.

[61] V. Sobotikova and M. Feistauer, "Effect of numerical integration in the DGFEM for nonlinear convection-diffusion problems," *Numerical Methods for Partial Differential Equations*, vol. 23, no. 6, 2007, pp. 1368–1395.

[62] P. Spalart, "Strategies for turbulence modeling and simulations," *International Journal of Heat and Fluid Flow*, vol. 21, no. 3, 2000, pp. 252–263.

[63] G. Strang and G. J. Fix, *An analysis of the finite element method*, Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[64] Y. Sun and Z. Wang, "Evaluation of discontinuous Galerkin and spectral volume methods for conservation laws on unstructured grids," *41st Aerospace Sciences Meeting*, Reno, NV, January 2003, AIAA, p. 11, AIAA, AIAA 2003-0253.

[65] J. Thompson, B. Soni, and N. Weatherill, eds., *Handbook of grid generation*, chapter Part IV: Adaptation and Quality, CRC Press, 1998.

[66] J. Thompson, Z. Warsi, and C. Mastin, *Numerical grid generation: Foundations and applications*, North Holland, 1985.

[67] E. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*, 2nd edition, Springer, Berlin, 1999, ISBN: 3540659668.

[68] B. van Leer, M. Lo, and M. van Raalte, "A discontinuous Galerkin method for diffusion based on recovery," *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, June 2007, AIAA, p. 12, AIAA 2007-4083.

[69] B. van Leer and S. Nomura, "A discontinuous Galerkin method for diffusion base on recovery," *17th AIAA Computational Fluid Dynamics Conference*, Toronto, Ontario, Canada, June 2005, AIAA, p. 30, AIAA 2005-5108.

[70] Z. Warsi, *Fluid dynamics: theoretical and computational approaches*, 2nd edition, CRC Press, Boca Raton, 1999, ISBN: 0849324076.

[71] E. W. Weisstein, "Gram-Schmidt Orthonormalization," From *MathWorld* – A Wolfram Web Resource, http://mathworld.wolfram.com/Gram-SchmidtOrthonormalization.html.

[72] D. C. Wilcox, *Turbulence modeling for CFD*, DCW Industries, 1993, ISBN 0963605100.

[73] M. Zhang and C.-W. Shu, *An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations,* Tech. Rep., Brown University, 2002.

APPENDIX A

DATA TABLES FOR THE SSV TEST CASES

The following tables list the L_1-error norms for the Supersonic Vortex (SSV) test cases described in Section 5.2 and for which the results are discussed in Section 5.3.

On each page, there are four tables depicting the L_1-error norm data for density, x-momentum, y-momentum, and total energy, in that order. Each table is split into two sections: the first section lists the actual L_1-error norms computed for each mesh configuration (labeled columns) versus the level of mesh refinement (each successive row represents a factor of two reduction in mesh spacing). The second section of each table is the rate at which the errors are being reduced as computed from Equation (3.66).

Blank entries in the tables indicate runs which failed to produce a converged solution. Typically, when the solution could not be obtained, the solver failed to reach iterative convergence to within the specified tolerance.

The data for the SSV cases with the solution approximation set to first through fourth-order are provided in Table A.1, Table A.2, Table A.3, Table A.4, Table A.5.

Table A.1

L_1-errors in ρ, ρu, ρv, and ρE for 1^{st} order SSV solutions

C_2	C_3	C_4	C_5	C_6	H_2	H_3
3.132e+00	2.998e+00	2.994e+00	2.994e+00	2.994e+00	2.994e+00	2.994e+00
2.324e+00	2.291e+00	2.290e+00	2.290e+00	2.290e+00	2.290e+00	2.290e+00
3.477e-01	3.415e-01	3.414e-01	3.414e-01	3.414e-01	3.414e-01	3.414e-01
1.509e-01	1.494e-01	1.494e-01	1.494e-01	1.494e-01	1.494e-01	1.494e-01
7.336e-02	7.299e-02	7.298e-02	7.298e-02	7.298e-02	7.298e-02	7.298e-02
3.678e-02	3.670e-02	3.670e-02	3.670e-02	3.670e-02	3.670e-02	3.670e-02
0.43	0.3877	0.3866	0.3865	0.3865	0.3867	0.3865
2.741	2.746	2.746	2.746	2.746	2.746	2.746
1.204	1.193	1.192	1.192	1.192	1.192	1.192
1.041	1.034	1.034	1.034	1.034	1.034	1.034

C_2	C_3	C_4	C_5	C_6	H_2	H_3
6.118e+01	9.948e+01	9.909e+01	9.910e+01	9.910e+01	9.898e+01	9.910e+01
4.881e+01	4.463e+01	4.467e+01	4.467e+01	4.467e+01	4.467e+01	4.467e+01
4.793e+01	4.712e+01	4.717e+01	4.717e+01	4.717e+01	4.717e+01	4.717e+01
2.754e+01	2.736e+01	2.737e+01	2.737e+01	2.737e+01	2.737e+01	2.737e+01
1.611e+01	1.607e+01	1.607e+01	1.607e+01	1.607e+01	1.607e+01	1.607e+01
9.222e+00	9.212e+00	9.212e+00	9.212e+00	9.212e+00	9.212e+00	9.212e+00
0.3259	1.156	1.149	1.15	1.15	1.148	1.15
0.02624	-0.07822	-0.07844	-0.07845	-0.07845	-0.07835	-0.07845
0.7994	0.7845	0.7852	0.7852	0.7852	0.7852	0.7852
0.7734	0.7678	0.7682	0.7682	0.7682	0.7682	0.7682

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.074e+02	1.513e+02	1.498e+02	1.498e+02	1.498e+02	1.497e+02	1.498e+02
5.040e+01	5.229e+01	5.218e+01	5.218e+01	5.218e+01	5.218e+01	5.218e+01
6.793e+01	6.846e+01	6.849e+01	6.849e+01	6.849e+01	6.849e+01	6.849e+01
4.105e+01	4.120e+01	4.121e+01	4.121e+01	4.121e+01	4.121e+01	4.121e+01
2.372e+01	2.376e+01	2.376e+01	2.376e+01	2.376e+01	2.376e+01	2.376e+01
1.334e+01	1.335e+01	1.335e+01	1.335e+01	1.335e+01	1.335e+01	1.335e+01
1.092	1.533	1.522	1.522	1.522	1.521	1.522
-0.4308	-0.3887	-0.3924	-0.3924	-0.3924	-0.3924	-0.3924
0.7268	0.7327	0.733	0.733	0.733	0.733	0.733
0.7913	0.7942	0.7944	0.7944	0.7944	0.7944	0.7944

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.528e+06	1.459e+06	1.457e+06	1.457e+06	1.457e+06	1.457e+06	1.457e+06
1.057e+06	1.040e+06	1.040e+06	1.040e+06	1.040e+06	1.040e+06	1.040e+06
1.511e+05	1.479e+05	1.479e+05	1.479e+05	1.479e+05	1.479e+05	1.479e+05
6.488e+04	6.411e+04	6.410e+04	6.410e+04	6.410e+04	6.410e+04	6.410e+04
3.176e+04	3.158e+04	3.158e+04	3.158e+04	3.158e+04	3.158e+04	3.158e+04
1.637e+04	1.633e+04	1.633e+04	1.633e+04	1.633e+04	1.633e+04	1.633e+04
0.5316	0.4882	0.4865	0.4864	0.4864	0.4866	0.4864
2.807	2.814	2.814	2.814	2.814	2.814	2.814
1.22	1.206	1.206	1.206	1.206	1.206	1.206
1.031	1.021	1.021	1.021	1.021	1.021	1.021

Table A.2

L_1-errors in ρ, ρu, ρv, and ρE for 2^{nd} order SSV solutions

C_2	C_3	C_4	C_5	C_6	H_2	H_3
4.030e-01	6.750e-02	6.752e-02	6.757e-02	6.757e-02	6.740e-02	6.757e-02
1.084e-01	2.124e-02	2.124e-02	2.124e-02	2.124e-02	2.124e-02	2.124e-02
3.125e-02	6.204e-03	6.204e-03	6.204e-03	6.204e-03	6.205e-03	6.204e-03
8.670e-03	1.699e-03	1.699e-03	1.699e-03	1.699e-03	1.699e-03	1.699e-03
2.373e-03	4.466e-04	4.466e-04	4.466e-04	4.466e-04	4.466e-04	4.466e-04
6.399e-04	1.147e-04	1.147e-04	1.147e-04	1.147e-04	1.147e-04	1.147e-04
1.895	1.668	1.668	1.669	1.669	1.666	1.669
1.794	1.776	1.776	1.776	1.776	1.776	1.776
1.85	1.868	1.868	1.868	1.868	1.869	1.868
1.87	1.928	1.928	1.928	1.928	1.928	1.928

C_2	C_3	C_4	C_5	C_6	H_2	H_3
9.253e+01	4.703e+01	4.700e+01	4.707e+01	4.707e+01	4.688e+01	4.707e+01
3.096e+01	1.380e+01	1.379e+01	1.380e+01	1.380e+01	1.379e+01	1.380e+01
9.507e+00	3.857e+00	3.857e+00	3.857e+00	3.857e+00	3.857e+00	3.857e+00
2.674e+00	1.025e+00	1.025e+00	1.025e+00	1.025e+00	1.025e+00	1.025e+00
7.508e-01	2.651e-01	2.651e-01	2.651e-01	2.651e-01	2.651e-01	2.651e-01
2.056e-01	6.743e-02	6.743e-02	6.743e-02	6.743e-02	6.743e-02	6.743e-02
1.579	1.769	1.769	1.77	1.77	1.765	1.77
1.703	1.839	1.839	1.839	1.839	1.838	1.839
1.83	1.912	1.912	1.912	1.912	1.912	1.912
1.832	1.952	1.952	1.952	1.952	1.952	1.952

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.426e+02	7.070e+01	7.072e+01	7.075e+01	7.075e+01	7.064e+01	7.075e+01
5.023e+01	2.250e+01	2.250e+01	2.250e+01	2.250e+01	2.251e+01	2.250e+01
1.537e+01	6.601e+00	6.602e+00	6.602e+00	6.602e+00	6.603e+00	6.602e+00
4.370e+00	1.813e+00	1.813e+00	1.813e+00	1.813e+00	1.813e+00	1.813e+00
1.209e+00	4.772e-01	4.772e-01	4.772e-01	4.772e-01	4.772e-01	4.772e-01
3.241e-01	1.226e-01	1.226e-01	1.226e-01	1.226e-01	1.226e-01	1.226e-01
1.505	1.652	1.652	1.653	1.653	1.65	1.653
1.708	1.769	1.769	1.769	1.769	1.769	1.769
1.815	1.865	1.865	1.865	1.865	1.865	1.865
1.854	1.926	1.926	1.926	1.926	1.926	1.926

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.925e+05	4.695e+04	4.695e+04	4.699e+04	4.699e+04	4.687e+04	4.699e+04
5.481e+04	1.465e+04	1.465e+04	1.465e+04	1.465e+04	1.465e+04	1.465e+04
1.609e+04	4.248e+03	4.249e+03	4.248e+03	4.248e+03	4.249e+03	4.248e+03
4.529e+03	1.159e+03	1.159e+03	1.159e+03	1.159e+03	1.159e+03	1.159e+03
1.249e+03	3.042e+02	3.042e+02	3.042e+02	3.042e+02	3.042e+02	3.042e+02
3.372e+02	7.803e+01	7.803e+01	7.803e+01	7.803e+01	7.803e+01	7.803e+01
1.812	1.68	1.68	1.682	1.682	1.678	1.682
1.768	1.786	1.786	1.786	1.786	1.786	1.786
1.829	1.874	1.874	1.874	1.874	1.874	1.874
1.859	1.93	1.93	1.93	1.93	1.93	1.93

Table A.3

L_1-errors in ρ, ρu, ρv, and ρE for 3^{rd} order SSV solutions

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	6.455e-03	6.404e-03	6.415e-03	6.415e-03	6.530e-03	6.415e-03
–	1.026e-03	1.018e-03	1.018e-03	1.018e-03	1.029e-03	1.018e-03
4.329e-02	1.476e-04	1.469e-04	1.470e-04	1.470e-04	1.478e-04	1.470e-04
1.187e-02	1.992e-05	1.987e-05	1.988e-05	1.988e-05	1.993e-05	1.988e-05
3.231e-03	2.591e-06	2.588e-06	2.588e-06	2.588e-06	2.592e-06	2.588e-06
8.677e-04	3.306e-07	3.304e-07	3.304e-07	3.304e-07	3.306e-07	3.304e-07
–	2.654	2.654	2.655	2.655	2.665	2.655
–	2.797	2.792	2.792	2.792	2.8	2.792
1.867	2.89	2.886	2.887	2.887	2.891	2.887
1.877	2.942	2.941	2.941	2.941	2.943	2.941

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	4.555e+00	4.479e+00	4.492e+00	4.492e+00	4.608e+00	4.492e+00
–	6.579e-01	6.519e-01	6.528e-01	6.528e-01	6.605e-01	6.528e-01
1.407e+01	8.992e-02	8.951e-02	8.957e-02	8.957e-02	9.007e-02	8.957e-02
3.963e+00	1.179e-02	1.177e-02	1.178e-02	1.178e-02	1.181e-02	1.178e-02
1.097e+00	1.513e-03	1.512e-03	1.512e-03	1.512e-03	1.514e-03	1.512e-03
2.949e-01	1.916e-04	1.915e-04	1.916e-04	1.916e-04	1.917e-04	1.916e-04
–	2.791	2.78	2.783	2.783	2.803	2.783
–	2.871	2.865	2.866	2.866	2.874	2.866
1.828	2.93	2.927	2.927	2.927	2.931	2.927
1.853	2.963	2.961	2.962	2.962	2.963	2.962

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	6.578e+00	6.517e+00	6.530e+00	6.530e+00	6.648e+00	6.530e+00
–	1.073e+00	1.066e+00	1.067e+00	1.067e+00	1.079e+00	1.067e+00
2.392e+01	1.564e-01	1.559e-01	1.560e-01	1.560e-01	1.568e-01	1.560e-01
6.495e+00	2.125e-02	2.122e-02	2.123e-02	2.123e-02	2.128e-02	2.123e-02
1.741e+00	2.773e-03	2.772e-03	2.773e-03	2.773e-03	2.777e-03	2.773e-03
4.594e-01	3.544e-04	3.545e-04	3.545e-04	3.545e-04	3.547e-04	3.545e-04
–	2.616	2.612	2.613	2.613	2.624	2.613
–	2.778	2.773	2.774	2.774	2.782	2.774
1.881	2.88	2.877	2.878	2.878	2.882	2.878
1.899	2.938	2.936	2.936	2.936	2.938	2.936

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	4.446e+03	4.402e+03	4.409e+03	4.409e+03	4.486e+03	4.409e+03
–	7.010e+02	6.961e+02	6.968e+02	6.968e+02	7.033e+02	6.968e+02
2.303e+04	1.005e+02	1.001e+02	1.002e+02	1.002e+02	1.007e+02	1.002e+02
6.325e+03	1.354e+01	1.351e+01	1.351e+01	1.351e+01	1.355e+01	1.351e+01
1.720e+03	1.760e+00	1.758e+00	1.758e+00	1.758e+00	1.760e+00	1.758e+00
4.614e+02	2.244e-01	2.243e-01	2.243e-01	2.243e-01	2.244e-01	2.243e-01
–	2.665	2.661	2.662	2.662	2.673	2.662
–	2.802	2.798	2.798	2.798	2.805	2.798
1.864	2.892	2.89	2.89	2.89	2.893	2.89
1.879	2.944	2.942	2.942	2.942	2.944	2.942

Table A.4

L_1-errors in ρ, ρu, ρv, and ρE for 4^{th} order SSV solutions

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	7.363e-04	3.027e-04	2.669e-04	2.669e-04	1.059e-03	2.669e-04
–	6.527e-05	3.007e-05	2.179e-05	2.179e-05	8.711e-05	2.179e-05
–	5.233e-06	2.449e-06	1.640e-06	1.640e-06	7.445e-06	1.640e-06
1.494e-02	3.814e-07	1.820e-07	1.141e-07	1.141e-07	5.846e-07	1.141e-07
4.152e-03	2.680e-08	1.253e-08	7.527e-09	7.527e-09	4.119e-08	7.527e-09
1.121e-03	1.837e-09	8.336e-10	4.830e-10	4.830e-10	2.748e-09	4.828e-10
–	3.496	3.331	3.614	3.615	3.603	3.615
–	3.641	3.618	3.732	3.732	3.549	3.732
–	3.778	3.75	3.845	3.845	3.671	3.845
1.847	3.831	3.86	3.922	3.922	3.827	3.922

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	2.667e-01	1.919e-01	1.770e-01	1.770e-01	3.994e-01	1.770e-01
–	2.154e-02	1.584e-02	1.472e-02	1.472e-02	2.868e-02	1.472e-02
–	1.758e-03	1.168e-03	1.058e-03	1.058e-03	2.255e-03	1.058e-03
4.793e+00	1.324e-04	8.120e-05	7.130e-05	7.130e-05	1.750e-04	7.130e-05
1.318e+00	9.475e-06	5.454e-06	4.624e-06	4.624e-06	1.268e-05	4.624e-06
3.506e-01	6.519e-07	3.603e-07	2.944e-07	2.943e-07	8.689e-07	2.942e-07
–	3.63	3.599	3.587	3.588	3.8	3.588
–	3.616	3.761	3.799	3.799	3.669	3.799
–	3.731	3.847	3.891	3.891	3.687	3.891
1.862	3.804	3.896	3.947	3.946	3.787	3.947

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	3.838e-01	2.347e-01	2.231e-01	2.231e-01	5.042e-01	2.231e-01
–	3.276e-02	2.387e-02	2.135e-02	2.135e-02	4.219e-02	2.135e-02
–	2.666e-03	1.896e-03	1.669e-03	1.669e-03	3.530e-03	1.669e-03
7.838e+00	1.977e-04	1.385e-04	1.176e-04	1.176e-04	2.791e-04	1.176e-04
2.122e+00	1.414e-05	9.531e-06	7.813e-06	7.813e-06	2.006e-05	7.813e-06
5.613e-01	9.859e-07	6.285e-07	5.029e-07	5.030e-07	1.367e-06	5.028e-07
–	3.55	3.298	3.386	3.386	3.579	3.386
–	3.619	3.654	3.677	3.677	3.579	3.677
–	3.753	3.775	3.827	3.827	3.661	3.827
1.885	3.805	3.862	3.912	3.912	3.799	3.912

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	3.758e+02	1.714e+02	1.582e+02	1.582e+02	5.465e+02	1.582e+02
–	3.386e+01	1.755e+01	1.438e+01	1.438e+01	4.459e+01	1.438e+01
–	2.728e+00	1.426e+00	1.091e+00	1.091e+00	3.797e+00	1.091e+00
7.819e+03	2.009e-01	1.063e-01	7.605e-02	7.605e-02	2.994e-01	7.605e-02
2.171e+03	1.423e-02	7.352e-03	5.022e-03	5.022e-03	2.130e-02	5.022e-03
5.856e+02	9.816e-04	4.888e-04	3.224e-04	3.224e-04	1.435e-03	3.223e-04
–	3.472	3.288	3.459	3.459	3.615	3.459
–	3.634	3.621	3.721	3.721	3.554	3.721
–	3.763	3.745	3.843	3.843	3.665	3.843
1.848	3.82	3.854	3.921	3.921	3.813	3.921

Table A.5

L_1-errors in ρ, ρu, ρv, and ρE for 5^{th} order SSV solutions

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.246e-03	4.953e-04	3.612e-05	3.594e-05	1.528e-03	3.599e-05
–	1.273e-04	5.377e-05	1.685e-06	1.675e-06	1.775e-04	1.677e-06
–	9.609e-06	4.128e-06	6.442e-08	6.436e-08	1.590e-05	6.438e-08
2.146e-02	9.388e-07	3.515e-07	2.271e-09	2.269e-09	1.075e-06	2.270e-09
7.778e-03	8.031e-08	3.266e-08	7.579e-11	7.589e-11	1.079e-07	7.579e-11
2.118e-03	–	–	–	–	–	–
–	3.291	3.203	4.422	4.423	3.105	4.424
–	3.727	3.703	4.709	4.702	3.481	4.703
–	3.356	3.554	4.826	4.826	3.887	4.826
1.464	3.547	3.428	4.905	4.902	3.316	4.904

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	3.894e-01	1.486e-01	2.400e-02	2.392e-02	4.562e-01	2.393e-02
–	3.237e-02	1.250e-02	1.054e-03	1.053e-03	4.302e-02	1.053e-03
–	2.400e-03	9.245e-04	3.886e-05	3.886e-05	3.728e-03	3.886e-05
6.023e+00	2.177e-04	7.607e-05	1.331e-06	1.331e-06	2.551e-04	1.331e-06
2.030e+00	1.773e-05	6.675e-06	4.374e-08	4.381e-08	2.370e-05	4.373e-08
5.503e-01	–	–	–	–	–	–
–	3.589	3.572	4.509	4.505	3.407	4.506
–	3.753	3.757	4.762	4.76	3.528	4.761
–	3.463	3.603	4.867	4.868	3.869	4.868
1.569	3.618	3.51	4.928	4.925	3.428	4.928

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	6.003e-01	2.533e-01	3.019e-02	3.013e-02	7.892e-01	3.015e-02
–	6.127e-02	2.674e-02	1.531e-03	1.527e-03	8.777e-02	1.527e-03
–	4.326e-03	1.921e-03	6.198e-05	6.195e-05	7.537e-03	6.195e-05
1.041e+01	4.280e-04	1.589e-04	2.227e-06	2.226e-06	4.698e-04	2.227e-06
3.758e+00	3.561e-05	1.453e-05	7.495e-08	7.508e-08	4.800e-05	7.498e-08
9.956e-01	–	–	–	–	–	–
–	3.292	3.244	4.301	4.303	3.169	4.303
–	3.824	3.799	4.627	4.623	3.542	4.624
–	3.337	3.596	4.799	4.798	4.004	4.798
1.47	3.587	3.451	4.893	4.89	3.291	4.892

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	6.383e+02	2.549e+02	2.351e+01	2.339e+01	7.842e+02	2.343e+01
–	6.469e+01	2.735e+01	1.114e+00	1.109e+00	9.007e+01	1.110e+00
–	4.832e+00	2.074e+00	4.313e-02	4.310e-02	8.014e+00	4.311e-02
1.093e+04	4.714e-01	1.758e-01	1.524e-03	1.523e-03	5.369e-01	1.523e-03
3.942e+03	4.008e-02	1.624e-02	5.088e-05	5.096e-05	5.379e-02	5.089e-05
1.070e+03	–	–	–	–	–	–
–	3.302	3.22	4.4	4.398	3.122	4.399
–	3.743	3.721	4.691	4.686	3.49	4.687
–	3.358	3.561	4.823	4.823	3.9	4.823
1.472	3.556	3.436	4.904	4.902	3.319	4.904

APPENDIX B

DATA TABLES FOR THE HRNTBL TEST CASES

The following tables list the L_1-error norms for the High-Reynolds Number Turbulent Boundary Layer (HRNTBL) test cases described in Section 5.4 and for which the results are discussed in Section **??**.

On each page, there are four tables depicting the L_1-error norm data for density, x-momentum, y-momentum, and total energy, in that order. Each table is split into two sections: the first section lists the actual L_1-error norms generated for each mesh configuration (labeled columns) versus the level of mesh refinement (each successive row represents a factor of two reduction in mesh spacing). The second section of each table is the rate at which the errors are being reduced as computed from Equation (3.66).

Blank entries in the tables indicate runs which failed to produce a converged solution. Typically, when the solution could not be obtained, the solver failed to reach iterative convergence to within the specified tolerance.

The data for case of $Re = 5e5$ are provided in Table B.1, Table B.2, and Table B.3. For the case of $Re = 1e6$, the data are presented in Table B.4, Table B.5, and Table B.6. For the case of $Re = 2e6$, the data is given in Table B.7, Table B.8, and Table B.9. For the case of $Re = 4e6$, the data is given in Table B.10, Table B.11, and Table B.12.

Table B.1

L_1-errors in ρ, ρu, ρv, and ρE for 2^{nd} order HRNTBL solutions at $Re = 5e5$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
2.053e-02*	1.089e-02	1.086e-02	1.088e-02	1.088e-02	1.083e-02	1.088e-02
4.759e-03*	2.699e-03	2.695e-03	2.696e-03	2.696e-03	2.693e-03	2.696e-03
1.041e-03	6.363e-04	6.360e-04	6.361e-04	6.361e-04	6.359e-04	6.361e-04
2.629e-04	1.520e-04	1.520e-04	1.520e-04	1.520e-04	1.520e-04	1.520e-04
7.129e-05*	–	3.713e-05	–	3.713e-05	3.713e-05	3.713e-05
2.109	2.013	2.011	2.013	2.013	2.008	2.013
2.193	2.085	2.083	2.084	2.084	2.082	2.084
1.985	2.066	2.065	2.065	2.065	2.065	2.065
1.883	–	2.033	–	2.033	2.033	2.033

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.433e+01*	2.911e+00	2.909e+00	2.915e+00	2.915e+00	2.899e+00	2.915e+00
3.751e+00*	6.982e-01	6.984e-01	6.986e-01	6.986e-01	6.982e-01	6.986e-01
9.742e-01	1.724e-01	1.724e-01	1.724e-01	1.724e-01	1.724e-01	1.724e-01
2.414e-01	4.337e-02	4.339e-02	4.339e-02	4.339e-02	4.339e-02	4.339e-02
6.018e-02*	–	1.101e-02	–	1.101e-02	1.101e-02	1.101e-02
1.934	2.06	2.058	2.061	2.061	2.054	2.061
1.945	2.018	2.018	2.018	2.018	2.018	2.018
2.013	1.991	1.991	1.991	1.991	1.99	1.991
2.004	–	1.978	–	1.978	1.978	1.978

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.056e+01*	3.167e+00	3.163e+00	3.170e+00	3.170e+00	3.154e+00	3.170e+00
2.908e+00*	7.393e-01	7.390e-01	7.394e-01	7.394e-01	7.384e-01	7.394e-01
7.718e-01	1.757e-01	1.756e-01	1.756e-01	1.756e-01	1.756e-01	1.756e-01
1.932e-01	4.453e-02	4.451e-02	4.451e-02	4.451e-02	4.451e-02	4.451e-02
4.901e-02*	–	1.142e-02	–	1.142e-02	1.142e-02	1.142e-02
1.86	2.099	2.098	2.1	2.1	2.095	2.1
1.914	2.073	2.073	2.074	2.074	2.072	2.074
1.998	1.98	1.98	1.98	1.98	1.98	1.98
1.979	–	1.962	–	1.962	1.962	1.962

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.024e+04*	3.757e+03	3.744e+03	3.753e+03	3.753e+03	3.729e+03	3.753e+03
2.526e+03*	9.474e+02	9.457e+02	9.463e+02	9.463e+02	9.448e+02	9.463e+02
5.812e+02	2.228e+02	2.227e+02	2.227e+02	2.227e+02	2.226e+02	2.227e+02
1.513e+02	5.276e+01	5.275e+01	5.275e+01	5.275e+01	5.275e+01	5.275e+01
4.193e+01*	–	1.268e+01	–	1.268e+01	1.268e+01	1.268e+01
2.019	1.988	1.985	1.988	1.988	1.981	1.988
2.12	2.088	2.087	2.087	2.087	2.086	2.087
1.941	2.078	2.078	2.078	2.078	2.077	2.078
1.852	–	2.056	–	2.056	2.056	2.056

Table B.2

L_1-errors in ρ, ρu, ρv, and ρE for 3^{rd} order HRNTBL solutions at $Re = 5e5$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.405e-04	9.883e-05	9.940e-05	9.943e-05	1.070e-04	9.943e-05
–	1.525e-05	1.199e-05	1.199e-05	1.200e-05	1.197e-05	1.200e-05
1.271e-03*	2.074e-06	1.902e-06	1.899e-06	1.899e-06	1.916e-06	1.899e-06
3.250e-04	2.327e-07	2.165e-07	2.166e-07	2.166e-07	2.174e-07	2.166e-07
–	–	–	–	–	–	–
–	3.204	3.043	3.051	3.051	3.16	3.051
–	2.878	2.657	2.659	2.659	2.644	2.659
1.968	3.156	3.135	3.132	3.132	3.14	3.132
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.034e-01	1.016e-01	1.018e-01	1.018e-01	1.056e-01	1.018e-01
–	1.278e-02	1.241e-02	1.243e-02	1.243e-02	1.262e-02	1.243e-02
5.911e-01*	1.766e-03	1.744e-03	1.743e-03	1.743e-03	1.742e-03	1.743e-03
2.366e-01	2.569e-04	2.436e-04	2.436e-04	2.436e-04	2.438e-04	2.436e-04
–	–	–	–	–	–	–
–	3.016	3.033	3.034	3.034	3.065	3.034
–	2.855	2.832	2.834	2.834	2.857	2.834
1.321	2.781	2.84	2.839	2.839	2.837	2.839
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.013e-01	9.638e-02	9.590e-02	9.592e-02	9.353e-02	9.592e-02
–	1.291e-02	1.206e-02	1.205e-02	1.205e-02	1.204e-02	1.205e-02
4.630e-01*	1.830e-03	1.730e-03	1.727e-03	1.727e-03	1.705e-03	1.727e-03
1.886e-01	2.541e-04	2.354e-04	2.353e-04	2.353e-04	2.341e-04	2.353e-04
–	–	–	–	–	–	–
–	2.972	2.999	2.993	2.993	2.958	2.993
–	2.819	2.801	2.802	2.802	2.82	2.802
1.296	2.848	2.877	2.876	2.876	2.864	2.876
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	4.798e+01	2.838e+01	2.809e+01	2.811e+01	3.014e+01	2.811e+01
–	4.690e+00	3.563e+00	3.538e+00	3.538e+00	3.467e+00	3.538e+00
7.012e+02*	5.854e-01	4.980e-01	4.989e-01	4.989e-01	5.108e-01	4.989e-01
1.673e+02	7.995e-02	7.480e-02	7.478e-02	7.478e-02	7.474e-02	7.478e-02
–	–	–	–	–	–	–
–	3.355	2.994	2.989	2.99	3.12	2.99
–	3.002	2.839	2.826	2.826	2.763	2.826
2.067	2.872	2.735	2.738	2.738	2.773	2.738
–	–	–	–	–	–	–

Table B.3

L_1-errors in ρ, ρu, ρv, and ρE for 4^{th} order HRNTBL solutions at $Re = 5e5$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.195e-04	1.378e-05	1.188e-05	1.186e-05	2.768e-05	1.186e-05
–	1.104e-05	7.809e-07	6.874e-07	6.870e-07	2.180e-06	6.869e-07
1.319e-03*	8.977e-07	4.375e-07	4.349e-07	4.349e-07	4.175e-07	4.349e-07
3.575e-04*	6.328e-08	4.501e-08	4.470e-08	4.470e-08	4.341e-08	4.605e-08
–	–	–	–	–	–	–
–	3.436	4.141	4.112	4.11	3.666	4.11
–	3.621	0.836	0.6605	0.6595	2.385	0.6595
1.884	3.827	3.281	3.282	3.282	3.266	3.24
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	6.273e-02	5.395e-03	3.732e-03	3.732e-03	2.295e-02	3.731e-03
–	7.779e-03	3.513e-04	2.274e-04	2.274e-04	1.330e-03	2.274e-04
9.589e-01*	1.004e-03	7.732e-05	7.437e-05	7.437e-05	1.324e-04	7.437e-05
2.430e-01*	1.268e-04	6.784e-06	6.631e-06	6.631e-06	9.873e-06	6.722e-06
–	–	–	–	–	–	–
–	3.012	3.941	4.037	4.036	4.109	4.036
–	2.954	2.184	1.613	1.613	3.328	1.613
1.98	2.985	3.511	3.487	3.487	3.746	3.468
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	6.740e-02	5.655e-03	4.598e-03	4.599e-03	1.940e-02	4.599e-03
–	8.341e-03	3.688e-04	2.425e-04	2.425e-04	1.085e-03	2.425e-04
7.730e-01*	1.072e-03	1.187e-04	1.217e-04	1.217e-04	1.757e-04	1.217e-04
2.040e-01*	1.312e-04	1.022e-05	1.038e-05	1.038e-05	1.344e-05	1.056e-05
–	–	–	–	–	–	–
–	3.015	3.938	4.245	4.245	4.16	4.245
–	2.96	1.636	0.9945	0.9948	2.626	0.9948
1.922	3.03	3.538	3.552	3.552	3.709	3.527
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	3.981e+01	4.872e+00	3.320e+00	3.313e+00	1.671e+01	3.312e+00
–	3.769e+00	2.874e-01	2.004e-01	2.003e-01	1.203e+00	2.003e-01
7.628e+02*	2.850e-01	1.108e-01	1.078e-01	1.078e-01	9.567e-02	1.078e-01
2.062e+02*	1.851e-02	1.247e-02	1.222e-02	1.222e-02	1.073e-02	1.269e-02
–	–	–	–	–	–	–
–	3.401	4.083	4.05	4.048	3.796	4.048
–	3.725	1.375	0.8947	0.8936	3.652	0.8935
1.888	3.944	3.151	3.141	3.141	3.156	3.086
–	–	–	–	–	–	–

Table B.4

L_1-errors in ρ, ρu, ρv, and ρE for 2^{nd} order HRNTBL solutions at $Re = 1e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
2.610e-02*	1.297e-02	1.293e-02	1.295e-02	1.295e-02	1.290e-02	1.295e-02
7.117e-03*	2.428e-03	2.422e-03	2.423e-03	2.423e-03	2.420e-03	2.423e-03
1.815e-03	4.767e-04	4.760e-04	4.761e-04	4.761e-04	4.759e-04	4.761e-04
4.379e-04*	1.188e-04*	1.188e-04*	1.188e-04*	1.188e-04*	1.188e-04*	1.188e-04*
1.075e-04*	3.155e-05*	3.155e-05*	3.155e-05*	3.155e-05*	3.155e-05*	3.155e-05*
1.875	2.417	2.417	2.418	2.418	2.415	2.418
1.972	2.349	2.347	2.347	2.347	2.346	2.347
2.051	2.004	2.003	2.003	2.003	2.002	2.003
2.026	1.913	1.913	1.913	1.913	1.913	1.913

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.513e+01*	4.133e+00	4.130e+00	4.137e+00	4.137e+00	4.118e+00	4.137e+00
3.988e+00*	1.026e+00	1.027e+00	1.027e+00	1.027e+00	1.026e+00	1.027e+00
1.008e+00	2.373e-01	2.373e-01	2.373e-01	2.373e-01	2.373e-01	2.373e-01
2.547e-01*	5.846e-02*	5.848e-02*	5.848e-02*	5.848e-02*	5.847e-02*	5.848e-02*
6.379e-02*	1.450e-02*	1.450e-02*	1.450e-02*	1.450e-02*	1.450e-02*	1.450e-02*
1.924	2.01	2.008	2.011	2.011	2.005	2.011
1.984	2.113	2.113	2.113	2.113	2.113	2.113
1.984	2.021	2.021	2.021	2.021	2.021	2.021
1.998	2.012	2.012	2.012	2.012	2.012	2.012

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.210e+01*	4.798e+00	4.793e+00	4.801e+00	4.801e+00	4.782e+00	4.801e+00
3.481e+00*	1.146e+00	1.146e+00	1.146e+00	1.146e+00	1.145e+00	1.146e+00
8.797e-01	2.392e-01	2.391e-01	2.391e-01	2.391e-01	2.391e-01	2.391e-01
2.149e-01*	5.829e-02*	5.828e-02*	5.829e-02*	5.829e-02*	5.828e-02*	5.829e-02*
4.374e-02*	1.450e-02*	1.450e-02*	1.450e-02*	1.450e-02*	1.450e-02*	1.450e-02*
1.798	2.066	2.065	2.066	2.066	2.062	2.066
1.985	2.261	2.261	2.261	2.261	2.26	2.261
2.033	2.037	2.037	2.037	2.037	2.036	2.037
2.297	2.007	2.007	2.007	2.007	2.007	2.007

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.254e+04*	4.568e+03	4.555e+03	4.564e+03	4.564e+03	4.541e+03	4.564e+03
3.372e+03*	8.916e+02	8.895e+02	8.901e+02	8.901e+02	8.885e+02	8.901e+02
8.818e+02	1.717e+02	1.715e+02	1.715e+02	1.715e+02	1.714e+02	1.715e+02
2.129e+02*	4.125e+01*	4.123e+01*	4.124e+01*	4.124e+01*	4.123e+01*	4.124e+01*
5.605e+01*	1.087e+01*	1.086e+01*	1.086e+01*	1.086e+01*	1.086e+01*	1.086e+01*
1.895	2.357	2.356	2.358	2.358	2.353	2.358
1.935	2.377	2.375	2.376	2.376	2.374	2.376
2.05	2.057	2.056	2.056	2.056	2.056	2.056
1.926	1.925	1.924	1.924	1.924	1.924	1.924

Table B.5

L_1-errors in ρ, ρu, ρv, and ρE for 3^{rd} order HRNTBL solutions at $Re = 1e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
2.452e-02*	2.822e-04	1.297e-04	1.312e-04	1.312e-04	1.425e-04	1.312e-04
5.140e-03*	2.337e-05	1.330e-05	1.330e-05	1.330e-05	1.344e-05	1.330e-05
1.434e-03*	2.306e-06	2.058e-06	2.054e-06	2.054e-06	2.051e-06	2.054e-06
3.710e-04	3.198e-07	3.051e-07	3.055e-07	3.046e-07	3.042e-07	3.046e-07
9.319e-05*	5.476e-08*	5.205e-08*	5.208e-08*	5.207e-08*	5.215e-08*	5.209e-08*
2.254	3.594	3.286	3.302	3.302	3.407	3.302
1.842	3.341	2.692	2.695	2.695	2.712	2.695
1.951	2.85	2.754	2.749	2.753	2.753	2.753
1.993	2.546	2.551	2.553	2.548	2.544	2.548

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.484e+01*	1.539e-01	1.499e-01	1.502e-01	1.502e-01	1.547e-01	1.502e-01
3.932e+00*	1.764e-02	1.736e-02	1.738e-02	1.738e-02	1.756e-02	1.738e-02
9.602e-01*	2.243e-03	2.208e-03	2.208e-03	2.208e-03	2.208e-03	2.208e-03
2.433e-01	3.067e-04	2.961e-04	2.961e-04	2.961e-04	2.960e-04	2.961e-04
6.230e-02*	4.736e-05*	4.378e-05*	4.378e-05*	4.378e-05*	4.379e-05*	4.378e-05*
1.916	3.124	3.11	3.111	3.112	3.138	3.112
2.034	2.976	2.975	2.977	2.977	2.992	2.977
1.981	2.87	2.899	2.898	2.898	2.899	2.898
1.965	2.695	2.758	2.758	2.758	2.757	2.758

C_2	C_3	C_4	C_5	C_6	H_2	H_3
8.266e+00*	1.555e-01	1.500e-01	1.496e-01	1.496e-01	1.469e-01	1.496e-01
3.219e+00*	1.772e-02	1.663e-02	1.661e-02	1.661e-02	1.650e-02	1.661e-02
7.712e-01*	2.270e-03	2.127e-03	2.124e-03	2.124e-03	2.103e-03	2.124e-03
2.011e-01	3.092e-04	2.834e-04	2.833e-04	2.832e-04	2.820e-04	2.832e-04
5.211e-02*	4.803e-05*	4.315e-05*	4.314e-05*	4.314e-05*	4.304e-05*	4.314e-05*
1.361	3.134	3.174	3.171	3.171	3.155	3.171
2.062	2.965	2.967	2.967	2.967	2.972	2.967
1.939	2.876	2.908	2.906	2.907	2.899	2.907
1.948	2.687	2.715	2.715	2.715	2.712	2.715

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.311e+04*	9.813e+01	4.326e+01	4.310e+01	4.311e+01	4.407e+01	4.311e+01
2.549e+03*	7.756e+00	4.284e+00	4.273e+00	4.273e+00	4.369e+00	4.273e+00
6.947e+02*	7.190e-01	6.047e-01	6.056e-01	6.056e-01	6.144e-01	6.056e-01
1.823e+02	9.441e-02	8.989e-02	9.004e-02	8.996e-02	9.055e-02	8.996e-02
4.844e+01*	1.549e-02*	1.484e-02*	1.486e-02*	1.486e-02*	1.492e-02*	1.486e-02*
2.362	3.661	3.336	3.334	3.335	3.334	3.335
1.875	3.431	2.825	2.819	2.819	2.83	2.819
1.93	2.929	2.75	2.75	2.751	2.762	2.751
1.912	2.608	2.599	2.599	2.598	2.601	2.598

Table B.6

L_1-errors in ρ, ρu, ρv, and ρE for 4^{th} order HRNTBL solutions at $Re = 1e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
3.109e-02*	2.075e-04	1.875e-05	1.459e-05	1.455e-05	4.063e-05	1.454e-05
–	1.831e-05	2.719e-06	2.650e-06	2.650e-06	2.987e-06	2.650e-06
1.440e-03*	9.964e-07*	2.386e-07	2.350e-07	2.350e-07*	2.381e-07*	2.349e-07*
–	3.923e-08*	2.848e-09*	1.766e-09*	1.766e-09*	8.958e-09*	1.766e-09*
–	–	–	–	–	–	–
–	3.502	2.786	2.461	2.457	3.766	2.456
–	4.2	3.51	3.495	3.495	3.649	3.496
–	4.667	6.388	7.056	7.056	4.732	7.056
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.560e+01*	7.116e-02	6.623e-03	4.869e-03	4.866e-03	2.329e-02	4.866e-03
–	9.075e-03	6.744e-04	5.970e-04	5.970e-04	1.667e-03	5.970e-04
9.911e-01*	1.153e-03*	4.988e-05	4.510e-05	4.510e-05*	1.082e-04*	4.509e-05*
–	1.480e-04*	1.456e-06*	1.063e-06*	1.063e-06*	4.243e-06*	1.063e-06*
–	–	–	–	–	–	–
–	2.971	3.296	3.028	3.027	3.804	3.027
–	2.977	3.757	3.727	3.726	3.945	3.727
–	2.961	5.098	5.407	5.407	4.673	5.406
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
9.424e+00*	7.579e-02	6.548e-03	5.225e-03	5.226e-03	1.916e-02	5.226e-03
–	9.934e-03	9.154e-04	9.621e-04	9.622e-04	1.960e-03	9.622e-04
8.287e-01*	1.206e-03*	6.908e-05	7.226e-05	7.226e-05*	1.352e-04*	7.224e-05*
–	1.497e-04*	1.471e-06*	1.061e-06*	1.062e-06*	3.960e-06*	1.062e-06*
–	–	–	–	–	–	–
–	2.932	2.839	2.441	2.441	3.289	2.441
–	3.042	3.728	3.735	3.735	3.858	3.735
–	3.01	5.553	6.089	6.089	5.093	6.089
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.526e+04*	7.026e+01	6.911e+00	5.184e+00	5.170e+00	1.828e+01	5.169e+00
–	6.343e+00	6.888e-01	5.938e-01	5.938e-01	1.196e+00	5.938e-01
7.860e+02*	3.597e-01*	5.961e-02	5.474e-02	5.473e-02*	7.512e-02*	5.472e-02*
–	1.622e-02*	1.456e-03*	6.202e-04*	6.201e-04*	4.884e-03*	6.201e-04*
–	–	–	–	–	–	–
–	3.469	3.327	3.126	3.122	3.934	3.122
–	4.141	3.531	3.439	3.439	3.993	3.44
–	4.471	5.356	6.464	6.464	3.943	6.463
–	–	–	–	–	–	–

Table B.7

L_1-errors in ρ, ρu, ρv, and ρE for 2^{nd} order HRNTBL solutions at $Re = 2e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
4.053e-02*	8.892e-03	8.870e-03	8.882e-03	8.883e-03	8.851e-03	8.883e-03
1.081e-02*	1.411e-03	1.410e-03	1.410e-03	1.410e-03	1.411e-03	1.410e-03
2.948e-03	2.319e-04	2.317e-04	2.317e-04	2.317e-04	2.316e-04	2.317e-04
6.688e-04*	6.499e-05*	6.495e-05*	5.935e-05*	6.496e-05*	6.495e-05*	6.496e-05*
1.495e-04*	1.787e-05*	1.420e-05*	1.828e-05*	1.828e-05*	1.828e-05*	1.828e-05*
1.907	2.656	2.653	2.655	2.655	2.65	2.655
1.874	2.605	2.606	2.605	2.605	2.606	2.605
2.14	1.835	1.835	1.965	1.835	1.835	1.835
2.161	1.863	2.193	1.699	1.829	1.829	1.829

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.664e+01*	4.598e+00	4.595e+00	4.602e+00	4.602e+00	4.583e+00	4.602e+00
4.152e+00*	1.198e+00	1.198e+00	1.198e+00	1.198e+00	1.197e+00	1.198e+00
9.976e-01	2.654e-01	2.654e-01	2.655e-01	2.655e-01	2.654e-01	2.655e-01
2.623e-01*	6.479e-02*	6.480e-02*	6.462e-02*	6.481e-02*	6.480e-02*	6.481e-02*
7.016e-02*	1.617e-02*	1.556e-02*	1.616e-02*	1.616e-02*	1.616e-02*	1.616e-02*
2.003	1.941	1.94	1.942	1.942	1.937	1.942
2.057	2.174	2.174	2.174	2.174	2.173	2.174
1.927	2.034	2.034	2.039	2.034	2.034	2.034
1.902	2.002	2.058	2	2.004	2.004	2.004

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.315e+01*	4.868e+00	4.863e+00	4.871e+00	4.871e+00	4.853e+00	4.871e+00
3.980e+00*	1.261e+00	1.261e+00	1.261e+00	1.261e+00	1.260e+00	1.261e+00
9.226e-01	2.600e-01	2.600e-01	2.600e-01	2.600e-01	2.600e-01	2.600e-01
2.069e-01*	6.334e-02*	6.333e-02*	6.291e-02*	6.333e-02*	6.333e-02*	6.333e-02*
4.552e-02*	1.580e-02*	1.547e-02*	1.580e-02*	1.580e-02*	1.580e-02*	1.580e-02*
1.724	1.949	1.948	1.95	1.95	1.946	1.95
2.109	2.278	2.277	2.278	2.278	2.277	2.278
2.157	2.037	2.037	2.047	2.038	2.037	2.038
2.184	2.004	2.034	1.993	2.003	2.003	2.003

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.874e+04*	3.428e+03	3.418e+03	3.426e+03	3.426e+03	3.406e+03	3.426e+03
4.959e+03*	6.484e+02	6.483e+02	6.484e+02	6.484e+02	6.481e+02	6.484e+02
1.339e+03	9.817e+01	9.811e+01	9.814e+01	9.814e+01	9.807e+01	9.814e+01
3.186e+02*	2.144e+01*	2.143e+01*	2.146e+01*	2.143e+01*	2.143e+01*	2.143e+01*
7.568e+01*	6.486e+00*	5.377e+00*	6.392e+00*	6.392e+00*	6.392e+00*	6.392e+00*
1.918	2.402	2.398	2.402	2.402	2.394	2.402
1.889	2.724	2.724	2.724	2.724	2.724	2.724
2.071	2.195	2.195	2.193	2.195	2.194	2.195
2.074	1.725	1.995	1.747	1.745	1.745	1.745

Table B.8

L_1-errors in ρ, ρu, ρv, and ρE for 3^{rd} order HRNTBL solutions at $Re = 2e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	2.260e-04	1.334e-04	1.337e-04	1.338e-04	1.445e-04	1.338e-04
–	1.595e-05	1.342e-05	1.340e-05	1.340e-05	1.339e-05	1.340e-05
–	1.888e-06	1.779e-06	1.776e-06	1.776e-06	1.754e-06	1.776e-06
4.345e-04	2.696e-07	2.562e-07	2.559e-07	2.559e-07	2.540e-07	2.559e-07
–	–	–	–	–	–	–
–	3.824	3.313	3.319	3.32	3.432	3.32
–	3.079	2.915	2.915	2.915	2.932	2.915
–	2.808	2.796	2.795	2.795	2.788	2.795
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.623e-01	1.583e-01	1.590e-01	1.590e-01	1.664e-01	1.590e-01
–	1.867e-02	1.835e-02	1.837e-02	1.837e-02	1.867e-02	1.837e-02
–	2.284e-03	2.235e-03	2.236e-03	2.236e-03	2.246e-03	2.236e-03
2.490e-01	3.005e-04	2.904e-04	2.904e-04	2.904e-04	2.909e-04	2.904e-04
–	–	–	–	–	–	–
–	3.12	3.109	3.114	3.114	3.156	3.114
–	3.031	3.038	3.039	3.039	3.055	3.039
–	2.926	2.944	2.944	2.944	2.949	2.944
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.710e-01	1.595e-01	1.593e-01	1.593e-01	1.589e-01	1.593e-01
–	1.897e-02	1.781e-02	1.778e-02	1.778e-02	1.760e-02	1.778e-02
–	2.293e-03	2.168e-03	2.166e-03	2.166e-03	2.147e-03	2.166e-03
1.783e-01	2.985e-04	2.826e-04	2.825e-04	2.825e-04	2.815e-04	2.825e-04
–	–	–	–	–	–	–
–	3.173	3.163	3.163	3.163	3.174	3.163
–	3.048	3.038	3.038	3.038	3.035	3.038
–	2.941	2.94	2.939	2.939	2.931	2.939
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	9.025e+01	5.430e+01	5.382e+01	5.383e+01	5.780e+01	5.383e+01
–	6.262e+00	4.991e+00	4.971e+00	4.972e+00	4.967e+00	4.972e+00
–	6.802e-01	6.286e-01	6.274e-01	6.274e-01	6.215e-01	6.274e-01
2.324e+02	9.512e-02	9.240e-02	9.228e-02	9.228e-02	9.150e-02	9.228e-02
–	–	–	–	–	–	–
–	3.849	3.444	3.436	3.437	3.541	3.437
–	3.203	2.989	2.986	2.986	2.999	2.986
–	2.838	2.766	2.765	2.765	2.764	2.765
–	–	–	–	–	–	–

Table B.9

L_1-errors in ρ, ρu, ρv, and ρE for 4^{th} order HRNTBL solutions at $Re = 2e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	2.606e-04	3.264e-05	1.172e-05	1.168e-05	9.161e-05	1.169e-05
–	1.689e-05	1.619e-06	5.828e-07	5.824e-07	4.259e-06	5.824e-07
–	9.509e-07*	9.339e-08*	6.175e-08*	6.175e-08*	2.046e-07*	6.175e-08*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.947	4.333	4.33	4.326	4.427	4.326
–	4.151	4.116	3.238	3.238	4.38	3.237
–	–	–	–	–	–	–
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.018e-01	8.293e-03	4.028e-03	4.021e-03	2.979e-02	4.021e-03
–	1.111e-02	4.699e-04	2.482e-04	2.480e-04	1.658e-03	2.480e-04
–	1.390e-03*	2.956e-05*	2.112e-05*	2.111e-05*	9.939e-05*	2.111e-05*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.196	4.142	4.021	4.019	4.167	4.019
–	2.999	3.991	3.555	3.554	4.061	3.554
–	–	–	–	–	–	–
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.022e-01	6.217e-03	4.392e-03	4.390e-03	2.060e-02	4.390e-03
–	1.161e-02	4.419e-04	3.025e-04	3.025e-04	1.273e-03	3.025e-04
–	1.434e-03*	2.993e-05*	2.474e-05*	2.474e-05*	8.782e-05*	2.474e-05*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.138	3.814	3.86	3.859	4.016	3.859
–	3.016	3.884	3.612	3.612	3.858	3.612
–	–	–	–	–	–	–
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	9.619e+01	1.166e+01	4.329e+00	4.313e+00	3.592e+01	4.314e+00
–	6.520e+00	6.281e-01	1.807e-01	1.805e-01	1.956e+00	1.805e-01
–	3.946e-01*	3.914e-02*	1.636e-02*	1.636e-02*	1.080e-01*	1.636e-02*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.883	4.215	4.583	4.579	4.199	4.579
–	4.046	4.004	3.465	3.464	4.179	3.464
–	–	–	–	–	–	–
–	–	–	–	–	–	–

Table B.10

L_1-errors in ρ, ρu, ρv, and ρE for 2^{nd} order HRNTBL solutions at $Re = 4e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
4.261e-02*	4.783e-03*	4.756e-03*	4.771e-03*	4.770e-03*	4.744e-03*	4.770e-03*
7.657e-03*	7.987e-04*	7.974e-04*	7.977e-04*	7.977e-04*	7.977e-04*	7.977e-04*
3.066e-03*	1.415e-04	1.414e-04	1.414e-04	1.414e-04	1.413e-04	1.414e-04
–	4.214e-05*	3.892e-05*	4.211e-05*	4.206e-05*	4.110e-05*	4.114e-05*
1.658e-04*	6.257e-06*	6.261e-06*	6.260e-06*	5.478e-06*	6.236e-06*	6.260e-06*
2.477	2.582	2.576	2.58	2.58	2.572	2.58
1.32	2.497	2.496	2.496	2.496	2.497	2.496
–	1.748	1.861	1.748	1.75	1.782	1.781
–	2.752	2.636	2.75	2.941	2.72	2.716

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.861e+01*	4.030e+00*	4.023e+00*	4.036e+00*	4.036e+00*	4.004e+00*	4.036e+00*
3.658e+00*	9.852e-01*	9.849e-01*	9.856e-01*	9.856e-01*	9.838e-01*	9.856e-01*
1.239e+00*	2.344e-01	2.344e-01	2.345e-01	2.345e-01	2.344e-01	2.345e-01
–	5.797e-02*	5.795e-02*	5.798e-02*	5.798e-02*	5.796e-02*	5.797e-02*
6.501e-02*	1.426e-02*	1.427e-02*	1.427e-02*	1.437e-02*	1.427e-02*	1.427e-02*
2.347	2.032	2.03	2.034	2.034	2.025	2.034
1.561	2.071	2.071	2.072	2.072	2.07	2.072
–	2.016	2.016	2.016	2.016	2.016	2.016
–	2.023	2.022	2.023	2.012	2.022	2.023

C_2	C_3	C_4	C_5	C_6	H_2	H_3
1.196e+01*	4.444e+00*	4.436e+00*	4.446e+00*	4.446e+00*	4.421e+00*	4.446e+00*
2.795e+00*	1.019e+00*	1.019e+00*	1.019e+00*	1.019e+00*	1.018e+00*	1.019e+00*
7.014e-01*	2.398e-01	2.397e-01	2.398e-01	2.398e-01	2.397e-01	2.398e-01
–	5.887e-02*	5.836e-02*	5.886e-02*	5.885e-02*	5.876e-02*	5.877e-02*
4.033e-02*	1.427e-02*	1.426e-02*	1.426e-02*	1.428e-02*	1.426e-02*	1.426e-02*
2.097	2.124	2.122	2.125	2.125	2.119	2.125
1.995	2.088	2.087	2.088	2.088	2.086	2.088
–	2.026	2.038	2.026	2.027	2.028	2.029
–	2.045	2.033	2.045	2.043	2.043	2.043

C_2	C_3	C_4	C_5	C_6	H_2	H_3
2.260e+04*	2.249e+03*	2.238e+03*	2.246e+03*	2.246e+03*	2.229e+03*	2.246e+03*
4.625e+03*	3.805e+02*	3.801e+02*	3.803e+02*	3.803e+02*	3.800e+02*	3.803e+02*
1.530e+03*	4.558e+01	4.553e+01	4.557e+01	4.557e+01	4.548e+01	4.557e+01
–	9.478e+00*	1.007e+01*	9.475e+00*	9.484e+00*	9.559e+00*	9.564e+00*
8.564e+01*	3.420e+00*	3.424e+00*	3.423e+00*	2.534e+00*	3.405e+00*	3.423e+00*
2.289	2.564	2.558	2.562	2.562	2.553	2.562
1.596	3.061	3.061	3.061	3.061	3.063	3.061
–	2.266	2.177	2.266	2.264	2.25	2.252
–	1.47	1.556	1.469	1.904	1.489	1.482

Table B.11

L_1-errors in ρ, ρu, ρv, and ρE for 3^{rd} order HRNTBL solutions at $Re = 4e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.596e-04*	1.347e-04*	1.319e-04*	1.319e-04*	1.288e-04*	1.319e-04*
–	1.287e-05	1.241e-05	1.230e-05	1.230e-05	1.200e-05	1.230e-05
–	1.535e-06	1.517e-06	1.513e-06	1.513e-06	1.478e-06	1.513e-06
–	2.422e-07*	2.316e-07*	2.111e-07*	2.291e-07*	2.296e-07*	2.312e-07*
–	–	–	–	–	–	–
–	3.632	3.44	3.423	3.423	3.424	3.423
–	3.068	3.032	3.023	3.023	3.021	3.023
–	2.664	2.712	2.841	2.723	2.686	2.71
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.263e-01*	1.169e-01*	1.174e-01*	1.174e-01*	1.276e-01*	1.174e-01*
–	1.354e-02	1.297e-02	1.301e-02	1.301e-02	1.352e-02	1.301e-02
–	1.704e-03	1.676e-03	1.678e-03	1.678e-03	1.700e-03	1.678e-03
–	2.374e-04*	2.386e-04*	2.363e-04*	2.386e-04*	2.394e-04*	2.386e-04*
–	–	–	–	–	–	–
–	3.222	3.172	3.174	3.174	3.239	3.174
–	2.99	2.952	2.955	2.955	2.992	2.955
–	2.843	2.813	2.828	2.814	2.828	2.814
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.275e-01*	1.210e-01*	1.203e-01*	1.203e-01*	1.152e-01*	1.203e-01*
–	1.338e-02	1.264e-02	1.259e-02	1.259e-02	1.220e-02	1.259e-02
–	1.667e-03	1.613e-03	1.610e-03	1.610e-03	1.583e-03	1.610e-03
–	2.255e-04*	2.215e-04*	2.210e-04*	2.211e-04*	2.195e-04*	2.213e-04*
–	–	–	–	–	–	–
–	3.252	3.259	3.256	3.257	3.239	3.257
–	3.004	2.97	2.967	2.967	2.946	2.967
–	2.886	2.864	2.865	2.864	2.85	2.863
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	8.541e+01*	7.212e+01*	7.081e+01*	7.082e+01*	7.112e+01*	7.082e+01*
–	6.503e+00	5.997e+00	5.932e+00	5.932e+00	5.843e+00	5.932e+00
–	7.592e-01	7.366e-01	7.334e-01	7.334e-01	7.206e-01	7.334e-01
–	1.150e-01*	1.105e-01*	1.059e-01*	1.097e-01*	1.095e-01*	1.103e-01*
–	–	–	–	–	–	–
–	3.715	3.588	3.577	3.578	3.605	3.578
–	3.098	3.025	3.016	3.016	3.019	3.016
–	2.723	2.737	2.792	2.741	2.718	2.733
–	–	–	–	–	–	–

Table B.12

L_1-errors in ρ, ρu, ρv, and ρE for 4^{th} order HRNTBL solutions at $Re = 4e6$

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	2.156e-04	2.809e-05*	8.309e-06	8.298e-06	1.191e-04	8.299e-06
–	1.785e-05*	2.000e-06*	5.561e-07*	5.560e-07*	7.307e-06*	5.560e-07*
–	1.254e-06*	–	2.786e-08*	2.786e-08*	–	2.786e-08*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.594	3.812	3.901	3.9	4.027	3.9
–	3.831	–	4.319	4.319	–	4.319
–	–	–	–	–	–	–
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.196e-01	1.407e-02*	6.965e-03	6.961e-03	5.158e-02	6.961e-03
–	1.446e-02*	9.218e-04*	4.202e-04*	4.201e-04*	3.078e-03*	4.201e-04*
–	1.773e-03*	–	2.391e-05*	2.391e-05*	–	2.391e-05*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.048	3.932	4.051	4.051	4.067	4.051
–	3.028	–	4.135	4.135	–	4.135
–	–	–	–	–	–	–
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	1.305e-01	1.108e-02*	7.468e-03	7.467e-03	2.557e-02	7.467e-03
–	1.518e-02*	6.398e-04*	4.354e-04*	4.355e-04*	1.600e-03*	4.355e-04*
–	1.824e-03*	–	2.493e-05*	2.493e-05*	–	2.493e-05*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.104	4.115	4.1	4.1	3.998	4.1
–	3.056	–	4.126	4.126	–	4.126
–	–	–	–	–	–	–
–	–	–	–	–	–	–

C_2	C_3	C_4	C_5	C_6	H_2	H_3
–	9.883e+01	1.436e+01*	6.138e+00	6.134e+00	5.278e+01	6.134e+00
–	8.242e+00*	9.714e-01*	3.757e-01*	3.757e-01*	3.152e+00*	3.757e-01*
–	5.901e-01*	–	2.172e-02*	2.172e-02*	–	2.172e-02*
–	–	–	–	–	–	–
–	–	–	–	–	–	–
–	3.584	3.886	4.03	4.029	4.066	4.029
–	3.804	–	4.112	4.112	–	4.112
–	–	–	–	–	–	–
–	–	–	–	–	–	–