INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
Toward Dynamic Spectral/hp Refinement: Algorithms and Applications to Flow-Structure Interactions

by

Robert M. Kirby II

B.S., Mathematics and Computer and Information Sciences. The Florida State University 1997

Thesis
Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

May 2003
This dissertation by Robert M. Kirby II is accepted in its present form by the Division of Applied Mathematics as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date 06/16/2002

George Em Karniadakis, Director

Recommended to the Graduate Council

Date 08/16/2002

David Gottlieb, Reader

Date 08/16/2002

Jan S. Hesthaven, Reader

Approved by the Graduate Council

Date 08/19/2002

Peder J. Estrup

Dean of the Graduate School and Research

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Vita of Robert M. Kirby II

Citizenship:
United States of America

Education:

  Project Title: “Visualizing Fluid Flow Data: From the Canvas to the Cave”
  Advisor: Professor Andries van Dam

  Advisor: Professor George Em Karniadakis

• The Florida State University. Bachelor of Science degree: Majors: Applied Mathematics and Computer and Information Sciences: Graduated Summa Cum Laude.

Publications

Book


Book Chapters:


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Peer-Reviewed Journal Publications:


Peer-Reviewed Conference Publications:


Other Conference Publications (Extended Abstracts Refereed):


Conference Abstracts:


Acknowledgments

Acknowledgments are those things which are often written last but are presented first in the document. I am truly blessed to have had the opportunity to interact with many people in my quest for advanced degrees in Applied Mathematics and Computer Science, and for this I am truly grateful.

First of all, I must express my gratitude to Professor George Em Karniadakis, my Ph.D. advisor in Applied Mathematics. Throughout my five years at Brown, he has been a faithful advisor and mentor, refining my skills as a simulation scientist. I am also grateful to him for affording me the opportunity to work on an advanced degree in Computer Science while also working on my Ph.D. in Applied Mathematics. He has both encouraged and supported my collaboration with scientists in Computer Science and in other fields. I look forward to continuing our professional relationship as collaborators and scholars, and continuing our personal relationship as friends.

Secondly, I would like to thank Professor Andries van Dam, my advisor in Computer Science. From the very beginning, Andy made me feel "part of the team": I was not considered an Applied Mathematician trying to do Computer Science, but rather I was respected and encouraged as a simulation scientist who could combine skills in both areas. Andy continually encouraged me to seek out those things which I do well, and do them better: to objectively see those areas which need improvement, and to refine them. His mentoring is very much appreciated.

Thirdly, I would like to thank Professor David Laidlaw of Computer Science. I have had the opportunity to interact with David on many different levels - as a student in the classroom, as co-author on several papers, as consultant on a variety of projects, and most importantly, as a friend. His scientific insight, his humor, his candor, and most of all his friendship are greatly appreciated.

Among the Applied Mathematics faculty, I would specifically like to thank several professors who have influenced me as a scientist. First, I would like to thank Professor Chi-Wang Shu for providing me with both guidance and mentoring as a student. I would like to thank Professor Jan Hesthaven for being a faithful sounding board for my ideas.
and providing corrections to my wayward thoughts as needed. Lastly, I would like to thank Professor David Gottlieb for allowing me to question him concerning the details of mathematical ideas, to which he always provided insight.

Graduate student experiences are created not only during research and in the classroom. Quite often a significant part of the graduate school experience consists of the people whom you meet along the way. I am thankful to have been part of the amazing CRUNCH group, directed by George Karniadakis. First, I would like to acknowledge the alumni members of the group from whom I have gained much advice and support: Dr. Ali Beskok, Dr. Spencer Sherwin, and Dr. Igor Lomtev. I thank Dr. Tim Warburton for being my graduate student mentor during my first years in the CRUNCH group and for preparing me to continue the development of *NekTAR*; I thank Dr. Haralambos Marmanis for our wonderful discussions about a multitude of things both scientific and otherwise; I thank Dr. Constantinos Evangelinos for always being willing to provide computing assistance; and I thank Dr. Ma Xia for always being willing to lend a helping hand. I would also like to acknowledge specifically those in the lab with whom I have interacted daily - Didier Lucor, Dongbin Xiu, and Igor Pivkin - each person contributing to making lab-life survivable. Lastly, I acknowledge all those group members with whom I have interacted - some only briefly and some for prolonged amounts of time - each person has influenced my experience here at Brown.

I would like to acknowledge and thank the following for their help in various capacities as I attempted to finish my degrees: Andy Forsberg, Sam Fulcomer, George Loriot, Loring Holden, Melih Bitim, and Joe LaViola.

I want to acknowledge and thank those funding agencies which have supported the research presented in this thesis:

- AFOSR Asert: F49620-96-1-0267
- AFOSR: F49620-98-1-0315
- AFOSR: F49620-01-1-0035
- DARPA/ILL: 98-240
Lastly, but certainly not least, I would like to thank my loving wife Alison. She has supported me constantly and waited patiently (well, most of the time) while I slowly moved toward my goal of a Ph.D. I am thankful to her for her persistence through this part of our journey together, and look forward to the many new beginnings that await us.
# Contents

**Acknowledgments**

<table>
<thead>
<tr>
<th>1 Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Objectives</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Outline</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 Discontinuous Galerkin Method Studies</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Examination of the Stencil</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Examination of the Eigenspectrum</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Convergence Studies</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1 A Study of h-Convergence</td>
<td>15</td>
</tr>
<tr>
<td>2.3.2 A Study of p-Convergence</td>
<td>16</td>
</tr>
<tr>
<td>2.4 Examination of Stabilization Factors</td>
<td>19</td>
</tr>
<tr>
<td>2.4.1 Test Case: Viscous Burgers Equation with $\nu = 10^{-2}/\pi$</td>
<td>20</td>
</tr>
<tr>
<td>2.4.2 Test Case: Elemental Interfaces with Variable p-Order</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Arbitrary Lagrangian-Eulerian Formulation for Compressible Flows</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Discontinuous Galerkin ALE for Advection</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Grid Velocity Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>3.3 Flow Simulations</td>
<td>30</td>
</tr>
<tr>
<td>3.3.1 2D Flow Around a Pitching Airfoil</td>
<td>30</td>
</tr>
<tr>
<td>3.3.2 3D Flow Around A Wing with Endplates</td>
<td>33</td>
</tr>
</tbody>
</table>
7 Non-Conforming Discretizations

7.1 Non-Conforming Discretizations Which Maintain p-Convergence ...................................................... 82
7.2 Non-Conforming Discretizations Using Finite Volume h-Refinement .............................................. 85
   7.2.1 Numerical Experiments Using Finite Volume h-Refinement .............................................. 88
   7.2.2 Flow Simulations .................................................................................................................. 95

8 Summary ........................................................................................................................................ 100

A Documentation of the Compressible $NeTa$ Code

A.1 Introduction ...................................................................................................................................... 103
A.2 Compilation Guide ........................................................................................................................ 104
   A.2.1 Veclib ..................................................................................................................................... 105
   A.2.2 Hlib ....................................................................................................................................... 105
   A.2.3 Source Directories ............................................................................................................... 105
A.3 Execution Guide .............................................................................................................................. 106
   A.3.1 The rea File ....................................................................................................................... 106
   A.3.2 Running the Code .............................................................................................................. 109
A.4 Programming Guide ....................................................................................................................... 109
   A.4.1 Miscellaneous Programming Notes .................................................................................... 110

xiii
## List of Figures

2.1 LDG and Baumann-Oden stencils (left) and Bassi-Rebay stencil (right). The element containing the black dot denotes the element on which the solution is being computed, and shaded areas denote the elements from which information is required for completing that computation. .............................. 9

2.2 Eigenvalues of the operator $A$ in equation (2.9) when using Bassi-Rebay fluxes to formulate $A$. Ten elements were used in all cases; each plot denotes a different value of the number of modes ($M$). The ordinate is the complex imaginary axis, and the abscissa is the complex real axis. .............................. 11

2.3 Eigenvalues of the operator $A$ in equation (2.9) when using LDG fluxes to formulate $A$. Ten elements were used in all cases; each plot denotes a different value of the number of modes ($M$). The ordinate is the complex imaginary axis, and the abscissa is the complex real axis. .............................. 12

2.4 Eigenvalues of the operator $A$ in equation (2.9) when using Baumann-Oden fluxes to formulate $A$. Ten elements were used in all cases; each plot denotes a different value of the number of modes ($M$). The ordinate is the complex imaginary axis, and the abscissa is the complex real axis. .............................. 13

2.5 Modulus of the maximum eigenvalues versus the number of modes per element for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right). The symbols denote the actual modulus of the eigenvalue, and the solid line denotes a least-squares $M^4$ fit where $M$ is the number of modes used per element. .............................. 14
2.6 Comparison of the rate of convergence for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right) fluxes for two modes (M=2) and three modes (M=3) per element. The $L_2$ error is given on the ordinate, and the number of elements is given on the abscissa. $k$ denotes the magnitude of the slope.

2.7 Comparison of the rate of convergence for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right) fluxes for viscous Burgers equation with $\nu = 10^{-2}/\pi$. Six uniformly spaced elements were used. The $L_2$ error is given on the ordinate, and the number of modes per element is given on the abscissa.

2.8 Exact solution $u(x, y) = \sin(2\pi x)\sin(2\pi y)$ of differing p-order test problem (left): One-dimensional slice of the exact solution showing the elemental decomposition in the x-direction (right).

2.9 The first column denotes element-wise values obtained for element two, and the second column denotes element-wise values obtained for element three. The $L_\infty$ error (top row) and $L_2$ error (bottom row) are given on the ordinate, and the value of the $\eta_k$ parameter in the stabilization factor is given on the abscissa (and is denoted as the $\alpha$ scaling). The symbols are explained in the text.

3.1 Notation for a triangular element.

3.2 Graph showing vertices with associated velocities and edges with associated weights.

3.3 Full domain (left) and local tessellation (right) for the simulation around the NACA 0015 pitching airfoil. All dimensions are in units of chord length.

3.4 Comparison of the lift coefficient $C_L$ versus angle of attack in degrees for the 9th order case (solid line) versus the computational and experimental results presented in [71] (symbols). The chord Reynolds number is $Re = 45.000$, and the Mach number is $Ma = 0.2$.

3.5 Density contours for angles of attack $\alpha = 10.4$ (left), $\alpha = 26.1$ (center), and $\alpha = 52.2$ (right).
3.6 Skeleton mesh for flow past a three-dimensional NACA 0012 airfoil with endplates (top and middle). Iso-contours for x-component of momentum for $M = 0.5$ flow past a three-dimensional NACA 0012 airfoil with endplates (bottom) ................................................................. 34

3.7 Initial configuration (left) and deflected (right) NACA 0012 wing section between two end-plates. Iso-contours of streamwise momentum are shown. .......................... 35

4.1 Diagram showing the one-way coupling of NekTAR with StressCheck. .......................... 39

4.2 Screen snap-shot showing the model created in StressCheck (left); Screen snap-shot showing the deformation of the structure based upon unit normal loading on one sub-panel of the structure (right) .............................................. 40

4.3 Time history of the displacement of a point on the airfoil. Displacement and time are both in non-dimensional units. ................................................................. 41

4.4 Time history of the coefficient of lift for a stationary airfoil configuration (solid) versus the fluid-structure model (dashed). ................................................................. 41

4.5 Diagram showing the two-way coupling of NekTAR with StressCheck. .......................... 42

5.1 Comparison of the difference in modal coefficients when different numbers of quadrature points are used. Quadratic nonlinearity is shown on the left and cubic nonlinearity is shown on the right. ................................................................. 48

5.2 Solution of the viscous Burgers equation with $\nu = 10^{-5}$ evaluated at $T = 0.5$. In $A$, $Q = 24$ quadrature points are used for integrating both the advection and diffusion terms; and in $B$, $Q = 24$ quadrature points are used for the advection term, and only $Q = 17$ points are used for the diffusion terms. ........................................................................ 49

5.3 Solution of the viscous Burgers equation with $\nu = 10^{-2}/\pi$ evaluated at $T = 0.5$. In $A$, $Q = 17$ quadrature points are used for integrating both the advection and diffusion terms; in $B$, $Q = 24$ quadrature points are used for integrating both the advection and diffusion terms; and in $C$, $Q = 24$ quadrature points are used for the advection term, and only $Q = 17$ points are used for the diffusion terms. ........................................................................ 50
5.4 Solution of the inviscid Burgers equation evaluated at $T = 0.5$. Five equally spaced elements were used with 16 modes in each element. On the ordinate we plot the $L_2$ norm of the solution, and on the abscissa we plot the number of quadrature points used for numerical integration. Unstable solutions are denoted by *. Observe that after $Q = 24$ points, the $L_2$ norm of the solution does not change.

5.5 Modal coefficients in the middle element of the inviscid Burgers solution at time $T = 0.35$. Over-integration using $Q = 3M/2$ quadrature points leads to a stable solution unlike the $Q = M + 1$ case.

5.6 Duct flow domain: The cross-section is an equilateral triangle and the streamwise length is three times the triangle edge. On the left we show a frame of the entire domain with flood contour cut-planes of the fluid velocity in the streamwise ($w$) direction. In the center and on the right we present flood contour cut-planes of the fluid velocity in the streamwise ($w$) direction with arrows denoting the velocity in the crossflow ($u,v$) directions at $z = 1$ and $z = 2$ respectively.

5.7 Wall shear forces as a function of time for (a) ($Q = M + 1$); (b) ($Q = 2M$); and (c) ($Q = 3M/2$).

5.8 Full domain (left) and local triangulation (right) for the simulation around the NACA 0015 pitching airfoil. All dimensions are in units of chord length.

5.9 Comparison of the lift coefficient $C_L$ versus angle of attack in degrees for Cases A-C (left) and Case A compared with the computational and experimental results presented in [71] (right).

6.1 Solution of inviscid Burgers equation at time $T = 0.5$ using continuous Galerkin without (top) and with (bottom) SVV. Five equally spaced elements spanning $[-1,1]$ were used, each of which contained 16 modes.

6.2 Comparison of eigenspectrum of the two operators $SVV_1$ and $SVV_2$. Details of this case are presented in the text.

6.3 Solution of inviscid Burgers equation at time $T = 0.5$ using the discontinuous Galerkin method without (top) and with (bottom) SVV.
6.4 Modal coefficients in the middle element of the inviscid Burgers solution at time $T = 0.35$. Both over-integration and SVV lead to a stable solution unlike the collocation approach.

6.5 Modal coefficients in the middle element of the inviscid Burgers solution at time $T = 0.35$. Results for $SVV_1$ and $SVV_2$ are presented.

6.6 Plot of the dynamic coefficient $c(x,t)$ at the final time $T = 0.5$. The three cases are explained in the text.

6.7 Segment of the unstructured mesh for flow past an airfoil at 10 degrees angle of attack and $Re = 10,000$.

6.8 Amplitude at one time instance of spectral vanishing viscosity in flow past an airfoil at 10 degrees angle of attack and $Re = 10,000$.

6.9 Mesh in the crossflow plane for turbulent channel flow at $Re = 180$.

6.10 Mean-velocity profile for the turbulent channel flow. The symbols correspond to the benchmark solutions of Kim, Moin & Moser [46]. The solid-line corresponds to the under-resolved DNS, the dotted-line to $(MSVV = 2, \epsilon = 1/8)$, the dot-dashed-line to $(MSVV = 5, \epsilon = 5/8)$, and the dashed-line to $(MSVV = 5, \epsilon = 9/8)$.

6.11 Turbulence intensities for the turbulent channel flow. The symbols correspond to the benchmark solutions of Kim, Moin & Moser [46]. The solid-line corresponds to the under-resolved DNS, the dotted-line to $(MSVV = 2, \epsilon = 1/8)$, the dot-dashed-line to $(MSVV = 5, \epsilon = 5/8)$, and the dashed-line to $(MSVV = 5, \epsilon = 9/8)$.

6.12 Full domain (left) and local tessellation (right) for the simulation around the NACA 0015 pitching airfoil. All dimensions are in units of chord length.

6.13 Comparison of the lift coefficient $C_L$ versus angle of attack in degrees for SVV (solid), over-integrated (dashed), and the computational (circle) and experimental (asterisk) results presented in [71].

6.14 Contours of instantaneous streamwise momentum for dynamic SVV simulation at time $T = 131.5$.
6.15 Contours of instantaneous streamwise momentum for simulation with over-
ingen at time $T = 131.5$. ................................................................. 78

6.16 Time history of lift coefficient ($C_L$): The solid line corresponds to over-
gen ation and the dotted line to dynamic SVV. ........................................ 78

6.17 Time history of drag coefficient ($C_D$): The solid line corresponds to over-
gen ation and the dotted line to dynamic SVV. ........................................ 79

7.1 Diagram showing how non-conforming discretizations are “mortared” to-
gether. Notation used in the diagram is explained in the text. ............ 83

7.2 Three different discretizations used for demonstrating the effectiveness of
‘h’ non-conforming and ‘p’ non-conforming discretizations. The three cases
A-C (left,center, and right respectively) are discussed in the text. ........ 84

7.3 Error in the broken $H_1$ norm versus the number of modes per element for
discretizations 7.2. The symbols from case A and B overlap. ............... 84

7.4 Diagram showing the splitting scheme from a triangle $\Omega$ having a $P_N$ order
polynomial to a collection of $M^2$ finite volumes $\Omega_j$. $\Pi^1$ and $\Pi^2$ represent the
forward and backward transformations respectively for this splitting scheme. 86

7.5 Diagram showing the splitting scheme from a quadrilateral $\Omega$ having a $P_N$
order polynomial to a collection of $M^2$ finite volumes $\Omega_j$. $\Pi^1$ and $\Pi^2$ repre-
sent the forward and backward transformations respectively for this split-
ting scheme. ................................................................. 86

7.6 Original four element mesh with $7^{th}$ order polynomials per element (left):
original mesh partitioned into eight finite volumes per element (center):
original mesh partitioned into 32 finite volumes per element (right). .... 89

7.7 $L_\infty$ error versus the number of finite volume elements per side ($M$) into
which each macro-element was partitioned. .......................................... 90

7.8 Macro-element discretization consisting of two triangles, each with $7^{th}$ order
polynomial expansions (left). Finite volume partitioning consisting of 49
finite volumes per macro-element (center). Reconstructed solution using
$\Pi^2$ operator (right). ................................................................. 91
7.9 Macro-element discretization consisting of four quadrilaterals, each with $7^{th}$ polynomial expansions (left). Finite volume partitioning consisting of 64 finite volumes per macro-element (center). Reconstructed solution using $\Pi^2$ operator (right).

7.10 Macro-element discretization consisting of four quadrilaterals, each with $7^{th}$ polynomial expansions (left). Finite volume partitioning consisting of 49 finite volumes per macro-element (center). Reconstructed solution using $\Pi^2$ operator (right).

7.11 Diagram showing which discretization methods were used in the solution of the problem discussed above.

7.12 Comparison of the exact solution to the three cases discussed in the text evaluated at $T = 0.3$.

7.13 Full domain (left) and local tessellation (right) for the supersonic simulation around the NACA 0015 airfoil. All dimensions are in units of chord length.

7.14 Full domain (left) and local tessellation (right) for the supersonic simulation around the NACA 0015 airfoil after non-conforming refinement has been accomplished. Non-conforming refinement for case one is shown. All dimensions are in units of chord length.

7.15 Diagram showing which discretization methods were used in the solution of the problem discussed below. All dimensions are in units of chord length.

7.16 Density contours for supersonic flow past a NACA 0015. The finite volume solution using 12,000 quadrilaterals (left) and the case two mixed non-conforming finite volume/high-order element computation (right) are presented.

7.17 Mach number contours for supersonic flow past a NACA 0015. The finite volume solution using 12,000 quadrilaterals (left) and the case two mixed non-conforming finite volume/high-order element computation (right) are presented.

7.18 Line slices of the density (left) and Mach number (right) taken at $y = 0$ (where the airfoil centerline is at $y = 0$. All four computations are presented.
7.19 Density contours for three dimensional supersonic flow past a NACA 0015. On the left we present line contours of the density at two positions on the wing (at 20% and 80% of the length). This is done so that the full length of the wing can be seen. On the right, we present a flood contour cut-plane of the density taken at the middle of the wing.

A.1 The current functionality tree for the compressible \texttt{NekTar} code.
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Proposed DG methodologies for elliptic problems and the flux choices they represent.</td>
</tr>
<tr>
<td>2.2</td>
<td>$L_2$ error and value of $</td>
</tr>
<tr>
<td>2.3</td>
<td>Parameters used for the results presented in table 2.4.</td>
</tr>
<tr>
<td>2.4</td>
<td>Results for Bassi-Rebay flux with $\alpha_j$ stabilization and $\alpha_7$ stabilization used.</td>
</tr>
<tr>
<td>2.5</td>
<td>$L_2$ and $L_\infty$ errors for differing values of $p_1$ and $p_2$. The definitions of $p_1$ and $p_2$ are discussed in the text.</td>
</tr>
<tr>
<td>3.1</td>
<td>Simulation parameters for compressible flow past a NACA 0012 airfoil with endplates.</td>
</tr>
<tr>
<td>5.1</td>
<td>Number of quadrature points necessary for Gauss-Lobatto-Legendre quadrature to be exact for given polynomial orders of the integrand.</td>
</tr>
<tr>
<td>5.2</td>
<td>Number of quadrature points necessary for Gauss-Lobatto-Legendre quadrature to be exact in terms of the number of modes for the original expansion for given polynomial orders of the integrand.</td>
</tr>
<tr>
<td>5.3</td>
<td>Parameters used for the results presented in figure 5.2.</td>
</tr>
<tr>
<td>5.4</td>
<td>Error in the discrete $L_\infty$ norm for the three cases presented above.</td>
</tr>
<tr>
<td>5.5</td>
<td>Mean shear forces on each wall versus the quadrature order employed.</td>
</tr>
<tr>
<td>5.6</td>
<td>Polynomial order per tensor product direction and number of quadrature points per direction used for cases A-C.</td>
</tr>
</tbody>
</table>
6.1 Comparison of $L_2$ and $L_\infty$ errors for the inviscid Burgers equation using dynamic SVV ($SVV_1$) with $M_{SVV} = 8$ and $\epsilon = 1/16$. ................................................. 70

6.2 Comparison of $L_2$ and $L_\infty$ errors for the inviscid Burgers equation using dynamic SVV ($SVV_2$) with $M_{SVV} = 8$ and $\epsilon = 1/16$. ................................................. 70
Chapter 1

Introduction

Most of the numerical analysis work for engineering applications over the last 40 years have been geared towards the development of methods which are accurate, flexible, and robust. Most methods have as their primary focus one of these criteria, and for some methods even two of these criteria are met. However, the computational method which encapsulates all three criteria is quite elusive, for satisfying all three goals quite often requires compromise. We will now present a brief (and certainly incomplete) overview of some of the attempts that have been made to meet the three goals mentioned above, and will show how, in this thesis, we have attempted to design a methodology for the solution of the fluid-structure interaction problems which contains the culmination of all three ideas.

In the search for accurate methods, the global spectral methods of Gottlieb and Orszag [34] are the ideal method for solving smooth problems in regular geometries. Given sufficient regularity of the solution, increasing the number of degrees of freedom in the approximation leads to exponential convergence to the exact solution. In addition, it has been argued by Kreiss et al. [49] that for the long-time integration of time dependent solutions, high-order methods are the most cost-effective in terms of amount of work performed to attain a given level of error. In terms of fluid simulations, spectral methods have been successfully used in the direct numerical simulations of turbulence [34, 16]. Although meeting one of the goals above, global spectral methods are rather restrictive in that only with great effort, if at all, can they be applied to complex geometries. Secondly,
the convergence properties of these methods require high regularity of the solution. When a reduction in the regularity of the solution occurs, such as with shocks in a compressible flow, wiggles may develop which render the method unstable.

In the search for flexible methods, finite element and finite volume methods provide a tremendous amount of flexibility in handling complex geometry domains. Traditionally, both of these methods have used low-order polynomial expansions for construction of their local bases, linear modes in the case of finite elements and constants in the case of finite volumes. In addition, the use of the low-order bases tends to make the methods somewhat robust. In terms of fluid simulations, these types of discretizations have been successfully applied to aerodynamic problems [41]. Although flexible, standard finite elements and finite volumes exhibit rather slow convergence as the number of elements increases compared to the previously mentioned spectral methods. This restriction manifests itself in problems like electromagnetics in which for long-time integration of high-frequency wave solutions a prohibitively large number of elements is needed to reduce the dissipation and dispersion effects of the low-order schemes.

The confluence of these two concepts, the \( hp \) version of the finite element method, was pioneered by B.A. Szabo. Seeking to merge the strengths of the spectral and finite element methods, Szabo formulated a methodology which allowed the flexibility of a standard finite element method to be combined with the convergence properties of a spectral method. Instead of restricting oneself to a linear basis as in the standard finite element method, Szabo formulated a new \( hp \) finite element method which accommodates higher order polynomial expansions on each element. This new \( hp \) methodology allowed for a dual path to convergence - allowing an increase in polynomial order (\( p \) resolution) when the regularity of the solution is high while still maintaining the flexibility of an \( h \)-discretization both in modeling complex geometries and in solution regions of low regularity. Several versions of this approach have been successfully applied to both solid mechanics and fluid dynamics [5, 4, 61, 44].

The discontinuous Galerkin methods provides a high-order extension of the finite volume method in much the same way as Szabo's work extended standard finite elements. In their overview of the development of the discontinuous Galerkin method (DGM), Cock-
burn et al. [19] trace the developments of DGM and provide a succinct discussion of the merits of this extension to finite volumes. The DG methodology also allows for the dual path to convergence, making it highly desirable for computational aerodynamics problems. It has been proven that DGM satisfies a cell entropy inequality (which is stronger than $L^2$ stability) for general scalar nonlinear conservation laws in multiple space dimensions [42], and DGM satisfies $L^2$ stability for general scalar convection-diffusion equations in multidimensions [20]. Though both accurate and flexible, this method is not always robust. The methodology is conservative, but not always monotonicity-preserving.

In this work, we focus on the use of the discontinuous Galerkin method for solving the compressible Navier-Stokes equations for moving geometries. Following the work of Lomtev et al. [52], we employ the arbitrary Lagrangian-Eulerian method for solving flow problems in moving geometries. We further extend Lomtev's work by accomplishing the fluid-structure coupling with the hp-FEM code StressCheck developed under the direction of B.A. Szabo. In addition to this extension, we attempt to address some of the robustness issues of the discontinuous Galerkin method in four ways: through an examination of the fluxes used in the discontinuous Galerkin formulation, through an examination of over-integration as a means of alleviating the effects of polynomial aliasing, through an examination of the use of spectral vanishing viscosity as a means of maintaining monotonicity, and through an examination of the use of non-conforming finite volume discretizations for solving problems which contain shock discontinuities. In the next section, we present the explicit goals of this thesis.

1.1 Objectives

The goals of this thesis are:

- To examine the ramifications of different flux choices when solving elliptic and parabolic problems using the discontinuous Galerkin method.

- To extend the arbitrary Lagrangian-Eulerian (ALE) work of Lomtev et al. [52] to encompass all element types by formulating a generalized graph theory algorithm for mesh movement.
• To couple the spectral/hp element fluid code (\textit{NekTAR}) with the hp-FEM structural code StressCheck.

• To understand the role of polynomial aliasing in the simulation of high Reynolds number fluid flow computations.

• To provide a formulation of spectral vanishing viscosity (SVV) for the solution of the incompressible Navier-Stokes equations in which the filtering allows for scale-separation.

• To formulated SVV for the discontinuous Galerkin method, and demonstrate its use in the solution of the compressible Navier-Stokes equations.

• To implement non-conforming discretizations within the discontinuous Galerkin framework for solving the compressible Navier-Stokes equations under supersonic conditions.

1.2 Outline

This work is organized as follows. In chapter two we present a collection of studies of different fluxes for solving elliptic and parabolic problems using the discontinuous Galerkin method. In chapter three we present the arbitrary Lagrangian-Eulerian formulation for the discontinuous Galerkin formulation of the compressible Navier-Stokes equations, and present a generalized graph theory algorithm for computing the mesh movement. In chapter four we present the coupling of the fluid solver \textit{NekTAR} using the ALE algorithm with the hp-FEM structural solver StressCheck. In chapter five we examine polynomial aliasing and its effect on both incompressible and compressible flow simulations. In chapter six we present a spectral vanishing viscosity formulation for both the continuous Galerkin and discontinuous Galerkin formulations, and provide both incompressible and compressible flow examples respectively which demonstrate SVV’s effectiveness. In chapter seven we present a non-conforming implementation of the discontinuous Galerkin formulation for the compressible Navier-Stokes, and provide computational examples demonstrating its use for the simulation of supersonic flows around airfoils. We conclude in chapter seven.
by summarizing what was accomplished in this work, pointing out the relevant individual contributions that were made.
Chapter 2

Discontinuous Galerkin Method Studies

Although the original thrust of most discontinuous Galerkin research was in solving hyperbolic problems, the general proliferation of the DG methodology has also spread to the study of parabolic and elliptic problems. For example, works such as [7], in which the viscous compressible Navier-Stokes equations were solved, required that a discontinuous Galerkin formulation be extended beyond the hyperbolic advection terms to the viscous terms of the Navier-Stokes equations. Concurrently, both in [20] and [10] other discontinuous Galerkin formulations for parabolic and elliptic problems were proposed. In an effort to classify all the efforts made toward the use of DG methods for elliptic problems, Arnold et al., first in [2] and then more fully in [3], published a unified analysis of discontinuous Galerkin methods for elliptic problems.

In [3] a mathematical framework is provided for studying a variety of the different discontinuous Galerkin approaches for elliptic problems. In an attempt to ascertain which formulation was appropriate for us to use in the solution of the viscous compressible Navier-Stokes equations, we set out to study several of the different formulations presented in [3]. To accomplish this goal, we first recognize from [3] that the problem of solving
\[- \Delta u = f \quad \text{in } \Omega \quad \text{(2.1)}
\]
\[u = 0 \quad \text{on } \partial \Omega \quad \text{(2.2)}\]

can be formulated in the discrete case as follows.

Assume we are given a tessellation \( T_h = \{ K \} \) of the domain \( \Omega \). We define the following two spaces

\[V_h := \{ v \in L^2(\Omega) : v|_K \in P(K), \forall K \in T_h \}\]
\[\Sigma_h := \{ \tau \in [L^2(\Omega)]^2 : \tau|_K \in \Sigma(K), \forall K \in T_h \}\]

where \( P(K) = P_p(K) \) is the space of polynomial functions of degree at most \( p \geq 1 \) on \( K \) and \( \Sigma(K) = [P_p(K)]^2 \). Following [3] we now define the discrete solution of equation (2.1) as the problem of finding \( u_h \in V_h \) and \( \sigma_h \in \Sigma_h \) such that for all \( K \in T_h \)

\[
\int_K \sigma_h \cdot \tau \, dx = - \int_K u_h \nabla_h \cdot \tau \, dx + \int_{\partial K} \hat{u}_K \cdot n_K \cdot \tau \, ds \quad \text{(2.3)}
\]
\[
\int_K \sigma_h \cdot \nabla v \, dx = \int_K f v \, dx + \int_{\partial K} \hat{\sigma}_K \cdot n_K \, ds \quad \text{(2.4)}
\]

where the numerical fluxes \( \hat{\sigma}_K \) and \( \hat{u}_K \) are approximations to \( \sigma = \nabla u \) and \( u \) respectively, on the boundary of \( K \). Given this general unified formulation of the discrete problem, the two remaining choices which determine exactly which DG methodology is used are the choice of the numerical fluxes \( \hat{\sigma}_K \) and \( \hat{u}_K \). In table 2.1 we present the methodologies and the corresponding numerical fluxes which we will study in this chapter. The operator \( \{ \cdot \} \) denotes averaging across the interface while \( [\cdot] \) denotes the jump difference across the interface as described in [3].

The three primary fluxes which we will study in this chapter are the Bassi-Rebay flux...
Table 2.1: Proposed DG methodologies for elliptic problems and the flux choices they represent.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\hat{u}_K$</th>
<th>$\tilde{\sigma}_K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bassi-Rebay [7]</td>
<td>${u_h}$</td>
<td>${\sigma_h}$</td>
</tr>
<tr>
<td>Brezzi et al. [15]</td>
<td>${u_h}$</td>
<td>${\sigma_h} - \alpha_f([u_h])$</td>
</tr>
<tr>
<td>LDG [20]</td>
<td>${u_h} - \beta \cdot [u_h]$</td>
<td>${\sigma_h} + \beta \cdot [\sigma_h] - \alpha_j([u_h])$</td>
</tr>
<tr>
<td>Baumann-Oden [10]</td>
<td>${u_h} + n_K \cdot [u_h]$</td>
<td>$\nabla_h u_h$</td>
</tr>
</tbody>
</table>

[7] which we will denote with the initials BR, the LDG flux [20] which we will denote with the initials LDG, and the Baumann-Oden flux [10] which we will denote with the initials BO. To accomplish our study, we will follow the work of Shu in [65], and by algebraic manipulation rewrite equation (2.4) to eliminate the auxiliary variable $\sigma$ from the formulation (taking into account the proper flux when manipulating the variable out of the expression). For the one-dimensional parabolic problem

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}$$

on $[a,b]$ with periodic boundary conditions, this manipulation leads to the following systems for Bassi-Rebay, LDG, and Baumann-Oden respectively:

$$\frac{d\hat{u}_j}{dt} = A^{BR}_{-2} \hat{u}_{j-2} + A^{BR}_{-1} \hat{u}_{j-1} + A^{BR}_0 \hat{u}_j + A^{BR}_1 \hat{u}_{j+1} + A^{BR}_2 \hat{u}_{j+2} \quad (2.5)$$

$$\frac{d\tilde{u}_j}{dt} = A^{LDG}_{-1} \tilde{u}_{j-1} + A^{LDG}_0 \tilde{u}_j + A^{LDG}_1 \tilde{u}_{j+1} \quad (2.6)$$

$$\frac{d\tilde{u}_j}{dt} = A^{BO}_{-1} \tilde{u}_{j-1} + A^{BO}_0 \tilde{u}_j + A^{BO}_1 \tilde{u}_{j+1} \quad (2.7)$$

where $u$ denotes a vector of the modal coefficients of the polynomial expansion on an element $j$, and the matrices $A_k$ are formulated based upon the choice of the numerical fluxes $\hat{\sigma}_K$ and $\hat{u}_K$ in equation (2.4). The subscript $k$ on each matrix $A_k$ denotes the offset from the current element $j$ for which the solution is being sought. The particular LDG stencil above corresponds to a choice of $\beta = 1/2$ as in the work of [65]. A different choice of the $\beta$ parameter may lead to a wider stencil for LDG.
In this chapter, we will examine a variety of factors such as the stencil width, eigenspectrum, h-convergence properties, and p-convergence properties of the different numerical fluxes to explore the differences between the difference choices presented.

2.1 Examination of the Stencil

The first observation that can be made immediately upon examination of equation (2.7) is that both LDG and Baumann-Oden have shorter stencils than Bassi-Rebay. LDG and Baumann-Oden require information only from nearest neighboring elements, hence producing a three element stencil, while Bassi-Rebay requires what equates to a five element stencil. In the two-dimensional case, as shown in figure 2.1, LDG and Baumann-Oden require only local elemental communication, which for triangular meshes require only a four element stencil. For Bassi-Rebay, however, information from as many as ten elements may be required for the computation of the solution on a single element.

![Figure 2.1: LDG and Baumann-Oden stencils (left) and Bassi-Rebay stencil (right). The element containing the black dot denotes the element on which the solution is being computed, and shaded areas denote the elements from which information is required for completing that computation.](image)

This fact concerning the different methods is important when considering parallel communication costs: depending on the way in which the communications are imple-
mented. the Bassi-Rebay method requires twice the communication as both the LDG and Baumann-Oden methods.

2.2 Examination of the Eigenspectrum

To understand the ramifications of choosing each type of flux, we begin by studying the one-dimensional parabolic equation

\[
\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = 0 \quad \nu \in \mathbb{R}, \nu > 0
\]  

(2.8)
on [0,1] with periodic boundary conditions. We write our numerical approximation of equation (2.8) in the following form:

\[
\frac{d\hat{u}_g}{dt} = A\hat{u}_g
\]  

(2.9)
where \( \hat{u}_g \) denotes the concatenation of modal coefficients of each element (hence if given \( N \) elements, each having \( M \) modal coefficients, the size of \( \hat{u}_g \) is \( N \times M \)), and \( A \) is a size(\( \hat{u}_g \)) \( \times \) size(\( \hat{u}_g \)) square matrix. We now examine the eigenvalues of the operator \( A \) for the three different fluxes for the case in which \( \nu = 1 \), and ten equally spaced elements are used. Eigenspectra of \( A \) for one to nine modes per element are presented for the Bassi-Rebay (BR), LDG (LDG), and Baumann-Oden (BO) fluxes in figures 2.2, 2.3, and 2.4 respectively.

Examination of the eigenspectrum leads to the following observations:

- Both the Bassi-Rebay and LDG fluxes are purely diffusive (all eigenvalues lie on the negative real axis) up to machine precision. This is consistent with the fact that for both Bassi-Rebay and LDG the matrix \( A \) is both real and symmetric. The use of Baumann-Oden fluxes forms a real, non-symmetric operator \( A \), which is evident by eigenvalues which have non-zero imaginary components.

- When using only one mode (finite volumes), Baumann-Oden reduces to an inconsistent scheme, which is denoted by all the eigenvalues lying at \((0.0)\). For modes greater than or equal to two, Baumann-Oden fluxes provide a consistent scheme.
Figure 2.2: Eigenvalues of the operator $A$ in equation (2.9) when using Bassi-Rebay fluxes to formulate $A$. Ten elements were used in all cases; each plot denotes a different value of the number of modes ($M$). The ordinate is the complex imaginary axis, and the abscissa is the complex real axis.
Figure 2.3: Eigenvalues of the operator $A$ in equation (2.9) when using LDG fluxes to formulate $A$. Ten elements were used in all cases; each plot denotes a different value of the number of modes ($M$). The ordinate is the complex imaginary axis, and the abscissa is the complex real axis.
Figure 2.4: Eigenvalues of the operator $A$ in equation (2.9) when using Baumann-Oden fluxes to formulate $A$. Ten elements were used in all cases: each plot denotes a different value of the number of modes ($M$). The ordinate is the complex imaginary axis, and the abscissa is the complex real axis.
• The Baumann-Oden flux, although dispersive, does not show significant dispersion for low number of modes (Modes = 2.3). It is conceivable that this fact can be exploited to create a symmetric preconditioner which would accelerate the convergence of implicit methods which use Baumann-Oden fluxes.

• Comparison of the eigenspectrum of the operator formed using LDG fluxes and the operator formed using Bassi-Rebay fluxes shows that LDG requires a more stringent time step if an explicit time stepping method is used for the advancement of the ODE system given by equation (2.9). In one-dimension, this fact can be rationalized by examining the width of the stencil that is created (as is done in [65]) by the two methods considered. As pointed out earlier, for LDG the width of the stencil is three, whereas for Bassi-Rebay the width of the stencil is five. Hence the effective $\Delta x$ for the LDG operator is smaller than that of the Bassi-Rebay operator. This fact requires that when using an explicit time stepping scheme, the time step of the LDG method will be smaller than that of the Bassi-Rebay method so that the diffusion number limit can be maintained.

• In figure 2.5 we plot the modulus of the maximum eigenvalue versus the number of modes per element for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right).

![Figure 2.5](image)

Figure 2.5: Modulus of the maximum eigenvalues versus the number of modes per element for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right). The symbols denote the actual modulus of the eigenvalue, and the solid line denotes a least-squares $M^4$ fit where $M$ is the number of modes used per element.

All three flux choices show a $M^4$ scaling where $M$ is the number of modes used per element, however the coefficient is different between the fluxes. For both Bassi-Rebay
and Baumann-Oden. the leading coefficient in the least-squares approximation is 1.35 whereas in the case of LDG. the leading coefficient is around 4.1.

2.3 Convergence Studies

To continue our study, we now examine the convergence properties both for h-convergence and p-convergence.

2.3.1 A Study of h-Convergence

To study the h-convergence of the Bassi-Rebay, LDG, Oden-Baumann fluxes, we consider the one-dimensional parabolic equation

\[ \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = 0 \quad \nu \in \mathbb{R}, \nu > 0 \]

on \([0, 1]\) with periodic boundary conditions. An initial condition of \(u(x) = \sin(2\pi x)\) was used, and the error was examined at \(T = 0.1\). To determine the convergence rate for each flux choice, we hold the number of modes per element fixed (both \(M = 2\) and \(M = 3\) modes) while successively doubling the number of elements used in the spatial discretization. A plot of the \(L_2\) error versus the number of elements is presented in figure 2.6.

Observe that both Bassi-Rebay and Oden-Baumann have different convergence rates depending on whether the number of modes is odd or even while LDG maintains a constant convergence rate. This behavior is consistent with the results shown in [65]. Bassi-Rebay achieves \(M^{th}\) order convergence when \(M\) is even and \((M - 1)^{th}\) order convergence when \(M\) is odd. Baumann-Oden achieves \((M - 1)^{th}\) order convergence when \(M\) is even and \(M^{th}\) order convergence when \(M\) is odd. LDG achieves \(M^{th}\) order convergence for both odd and even \(M\). In this example, for an odd number of modes, \(M = 3\). Bassi-Rebay and LDG have the same convergence rate and have similar absolute errors. The LDG flux, in general, shows superior performance to both the Bassi-Rebay and the Baumann-Oden fluxes in that it has a consistent convergence rate not dependent on the parity of the polynomial order employed.
2.3.2 A Study of p-Convergence

To further our examination of different flux terms used for diffusion in the discontinuous Galerkin method, we examined the solution of viscous Burgers equation:

\[
\frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial u^2}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}.
\]  

(2.10)

For this problem, the viscosity is taken to be \( \nu = 10^{-2}/\pi \). This problem was used as a test problem in [6] for evaluating the effectiveness of different spatial discretizations. As is pointed in that paper, for \( \nu = 10^{-2}/\pi \) the solution develops into a sawtooth wave centered at the origin around the time \( t = 1/\pi \) (an approximation which comes from the inviscid theory for Burgers equation), and the maximum gradient of the solution occurs at approximately 0.5. As in [6], the Cole transformation is used to obtain the exact solution to equation (2.10):

\[
u(x,t) = -\frac{\int_{-\infty}^{\infty} \sin(\pi(x-\eta))f(x-\eta)\exp(-\eta^2/4\nu t) \, d\eta}{\int_{-\infty}^{\infty} f(x-\eta)\exp(-\eta^2/4\nu t) \, d\eta}
\]  

(2.11)
where
\[ f(y) = \exp(-\cos(\pi y/2\pi \nu)). \] (2.12)

In the current study, the convolution operator given in equation (2.11) was solved numerically using Gauss-Hermite integration with up to thirty terms. Also, for the advection operator, the conservative form of the equations were solved, and upwinding was used for the advection fluxes [51].

To evaluate the three different fluxes, we examined two quantities, the \(L_2\) error and the magnitude of the slope of the approximate solution at the origin. As is pointed out in [6], the slope at the origin is a sensitive quantity and hence provides a good measure of the accuracy of the numerical methods. According to [6], the analytical solution admits a maximum magnitude of the slope that the origin at \(T = 0.51047356\), where the maximum value is given as 152.00516. We verified these numbers numerically to four significant digits. All three methods were evaluated at time \(T = 0.5105\), with both the \(L_2\) error and magnitude of the slope at the origin provided. To calculate the derivative of the approximate solution, the auxiliary variable \(\sigma_h\) which approximates the first derivative from each method was used. This was done so that not only the elemental derivative but also the jump terms would be taken into account when tabulating the slope at the origin.

In figure 2.7 we compare the rate of convergence of the \(L_2\) error for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right) fluxes evaluated at time \(T = 3/\pi\). Six uniformly spaced elements were used, and the modal order was varied from four to eight.

In table 2.2 we compare the \(L_2\) error and value of \(|\frac{\partial u}{\partial x}(0)|\) at time \(T = 0.5105\) for the three different fluxes. Four uniformly spaced elements were used, each element containing 16 modes.

Both Bassi-Rebay and LDG show similar p-convergence properties for this problem, and both perform better than the Baumann-Oden flux. When examining the quantity \(|\frac{\partial u}{\partial x}(0)|\), both Bassi-Rebay and Baumann-Oden are superior to LDG. This may be due, however, to the symmetries in the problem and the averaging nature of the Bassi-Rebay and Baumann-Oden fluxes. Recall that the LDG flux takes information from only one side during the intermediary stage (the computation of \(\sigma^{LDG}\)) and uses information from
Figure 2.7: Comparison of the rate of convergence for Bassi-Rebay (left), LDG (center), and Baumann-Oden (right) fluxes for viscous Burgers equation with $\nu = 10^{-2}/\pi$. Six uniformly spaced elements were used. The $L_2$ error is given on the ordinate, and the number of modes per element is given on the abscissa.

| Flux Term       | $L_2$ Error | $\frac{|u_x(0)|}{\sqrt{\pi}}$ |
|-----------------|-------------|-------------------------------|
| Bassi-Rebay     | 0.0135916   | $\sigma_{BR} = 150.171$      |
| LDG             | 0.0103299   | $\sigma_{LDG} = 167.946$     |
| Baumann-Oden    | 0.028627    | $\sigma_{BO} = 150.858$      |
| Exact Value     | -           | 152.00516                    |

Table 2.2: $L_2$ error and value of $\frac{|u_x(0)|}{\sqrt{\pi}}$ for the Bassi-Rebay, LDG, and Oden-Baumann fluxes. The test problem is explained in the text.
the other direction in the final computation. This alternating feature of LDG may act as a form of prediction and correction so that final computation of the second derivative yields an accurate solution.

2.4 Examination of Stabilization Factors

Recently there has been an increased interest in the addition of a penalty-like term to stabilize the numerical solution of elliptic problems using the discontinuous Galerkin method. For LDG as presented in [20] a stabilization term of the form

$$\alpha_f([u_h]) = \eta_e h_e^{-1}[[u_h]]$$

exists, with the free parameter $\eta_e$ allowing the user to specify how "weak" or "strong" in some sense the element interface condition is. In LDG, this term is subtracted from the numerical flux approximation $\sigma_K^h$ yielding a modified flux $\sigma_K = \{\sigma_h\} + \mathcal{I}_h([[\sigma_h]] - \alpha_f([u_h]))$.

In [15] an alternative jump term was presented using the lift operator $r_e$ given by

$$\int_{\Omega} r_e(\phi) \cdot \tau \, dx = -\int_{\Sigma} \phi \cdot \{\tau\} \, ds \quad \forall \tau \in \Sigma_h, \quad \phi \in [L^1(e)]^2$$

where $\Omega$ is the domain over which we have our tessellation $T_h$, $e$ denotes an edge within that tessellation (which may be owned by only one element within the tessellation or may be shared by two elements), and where

$$\alpha_e(\phi) = -\eta_e r_e(\phi). \quad (2.13)$$

Once again, a free parameter $\eta_e$ allows this stabilizing factor to be tuned for the problem being solved. As with LDG, this stabilizing factor is subtracted from the standard Bassi-Rebay numerical flux approximation $\tilde{\sigma}_K = \{\sigma_h\}$ yielding a modified numerical flux $\tilde{\sigma}_K = \{\sigma_h\} - \alpha_e([u_h])$.

We first examine these two stabilizing factors when used with the Bassi-Rebay flux, and we then examine the use of this type of stabilization when interfaces having different orders on either side are used for solving elliptic problems.
2.4.1 Test Case: Viscous Burgers Equation with $\nu = 10^{-2}/\pi$

In an attempt to understand the effects of the aforementioned stabilizers, we revisited the Burgers equation example studied in section 2.3.2. In table 2.3 we present the parameters used for this study.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Discontinuous Galerkin</td>
</tr>
<tr>
<td>Diffusion Term</td>
<td>Bassi-Rebay</td>
</tr>
<tr>
<td>Number of Elements</td>
<td>4</td>
</tr>
<tr>
<td>Modes</td>
<td>16</td>
</tr>
<tr>
<td>Time Integrator</td>
<td>2nd order Adams-Bashforth</td>
</tr>
<tr>
<td>Final Time $T$</td>
<td>0.5105</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>0.0001</td>
</tr>
<tr>
<td>$\nu$</td>
<td>$10^{-2}/\pi$</td>
</tr>
</tbody>
</table>

Table 2.3: Parameters used for the results presented in table 2.4.

In table 2.4 we present the $L_2$ error and value of $|\frac{\partial u}{\partial x}(0)|$ for standard Bassi-Rebay, Bassi-Rebay with an $\alpha_J$ stabilization, and Bassi-Rebay with an $\alpha_R$ stabilization.

| Penalty Term                     | $L_2$ Error | $|\frac{\partial u}{\partial x}(0)|$ |
|----------------------------------|-------------|----------------------------------|
| No Penalty Term                  | 0.0135916   | 150.166                          |
| $\alpha_J([u_h])$ with $\eta' = 1$ | 0.00933867  | 164.212                          |
| $\alpha_R([u_h])$ with $\eta' = 1$ | 0.0159191   | 154.55                           |
| Exact Value                      | -           | 152.00516                        |

Table 2.4: Results for Bassi-Rebay flux with $\alpha_J$ stabilization and $\alpha_R$ stabilization used.

We observed in both cases that the stabilization parameter does indeed dictate in some sense how “weak” or “strong” the interfacing condition is. As the stabilization parameter was increased, the jump between the solutions at the element interfaces was decreased; however, this change in the jump condition invariably increased the error in the middle of the solution, which was recorded by the change in the $L_2$ error. From the limited experiments performed we conclude that the stabilization terms can be used to help minimize the $L_\infty$ error (which is normally greatest at the element interfaces) at the
expense of the $L_2$ error. This point is explored further in the next section.

2.4.2 Test Case: Elemental Interfaces with Variable p-Order

To understand the ramifications of variable p-order when solving elliptic problems with the discontinuous Galerkin method, we accomplished a numerical study of the following equation:

$$-\nabla^2 u(x, y) = f(x, y) \text{ on } \Omega = [0, 4] \times [0, 1]$$

with periodic boundary conditions on $\partial \Omega$ and exact solution $u(x, y) = \sin(2\pi x)\sin(2\pi y)$ as shown in figure 2.8 (left). The two-dimensional computational domain was decomposed into eight quadrilateral elements: four elements covering $[0, 4] \times [0, 0.5]$, and four elements covering $[0, 4] \times [0.5, 1]$. A one-dimensional slice of the exact solution with the elemental decomposition labeled is given in figure 2.8 (right).

In this study, the elements were set to differing polynomial orders, and the numerical solution computed. The standard Bassi-Rebay flux was used in all cases. The polynomial order was set to $p_1$ on elements covering $[0, 2] \times [0, 1]$ (i.e., elements one and two in figure 2.8 (left)) and to $p_2$ on elements covering $[2, 4] \times [0, 1]$ (i.e., elements three and four in figure 2.8 (right)). Because this domain is periodic, polynomial orders differ at both the internal elemental interface at $x = 2$ and at the periodic boundary condition interface $x = 0$. In table 2.5 we present the $L_2$ and $L_\infty$ errors for a collection of $p_1$ and $p_2$ combinations.

Figure 2.8: Exact solution $u(x, y) = \sin(2\pi x)\sin(2\pi y)$ of differing p-order test problem (left); One-dimensional slice of the exact solution showing the elemental decomposition in the x-direction (right).
Increasing the polynomial order to \( p_2 = 7 \) while keeping \( p_1 = 5 \) decreases both the \( L_2 \) and \( L_\infty \) errors as expected. Increasing \( p_2 \) further, however, deteriorates the accuracy of the numerical solution in both norms. In an attempt to understand why this may be true, we examined the even simpler one-dimensional problem:

\[
-\frac{d^2}{dx^2} u(x) = f(x) \quad \text{on} \quad \Omega = [0, 1]
\]

with periodic boundary conditions on \( \partial \Omega \) and exact solution \( u(x) = \sin(2\pi x) \) as shown in figure 2.8 (right).

To further understand how increasing the polynomial order across an interface can cause a worsening of the solution, we experimented with the stabilization parameter given in equation (2.13). In this study, we examine the element-wise error in elements two and three (see figure 2.8 (right)): the polynomial order within element two is given by \( p_1 \) and the polynomial order within element three is given by \( p_2 \). Four cases with no stabilization were considered:

- **Case 1:** \( p_1 = 5 \) and \( p_2 = 5 \) (denoted by a triangle).
- **Case 2:** \( p_1 = 5 \) and \( p_2 = 7 \) (denoted by a square).
- **Case 3:** \( p_1 = 5 \) and \( p_2 = 9 \) (denoted by a circle).
- **Case 4:** \( p_1 = 9 \) and \( p_2 = 9 \) (denoted by a diamond).

The results of this study are presented in figure 2.9. Observe that when no stabilization is used (i.e. when the \( \alpha \) scaling is zero), the results presented in figure 2.9 mimic those
Figure 2.9: The first column denotes element-wise values obtained for element two, and the second column denotes element-wise values obtained for element three. The $L_\infty$ error (top row) and $L_2$ error (bottom row) are given on the ordinate, and the value of the $\eta_c$ parameter in the stabilization factor is given on the abscissa (and is denoted as the $\alpha$ scaling). The symbols are explained in the text.
presented in table 2.5. When $p_1 = 5$ and $p_2 = 9$ are used, both the element-wise $L_2$ and $L_\infty$ values increase versus the $p_1 = 5, p_2 = 7$ case. Based on the stabilization study presented earlier, we decided to test the stabilization parameter given in equation (2.13) by varying the $\eta_e$ parameter. For the $p_1 = 5$ and $p_2 = 9$ case only, we gradually increased the $\eta_e$ parameter in discrete increments from zero to 2.0 as shown in figure 2.9 (line). Observe that any addition of the stabilization term increases the $L_2$ error, however, there exists an optimal value of the $\eta_e$ parameter which minimizes the $L_\infty$ error. This result is consistent with the work of [31] in which the scaling of a penalty parameter yields an optimal value at which a minimum error occurs. This optimal $\eta_e$ parameter significantly decreases the $L_\infty$ but only marginally modifies the $L_2$ error. This study shows that the stabilization parameter can be used to minimize the effects of gross discrepancies in the polynomial orders used in two adjacent elements.
Chapter 3

Arbitrary Lagrangian-Eulerian Formulation for Compressible Flows

As in [52], we consider the non-dimensional compressible Navier-Stokes equations, which we can write in a compact form in the Eulerian reference frame as

\[
\hat{U}_t + \nabla \cdot \mathbf{F} = Re_{\infty}^{-1} \nabla \cdot \mathbf{F}^\nu \quad \text{in } \Omega
\]

(3.1)

where \( \mathbf{F} \) and \( \mathbf{F}^\nu \) correspond to inviscid and viscous flux contributions, respectively, and \( Re_{\infty} \) is the reference Reynolds number. In the expression above, the vector \( \hat{U} = [\rho, \rho u_1, \rho u_2, \rho u_3, \rho e]^T \) with \( \mathbf{u} = (u_1, u_2, u_3) \) the local fluid velocity, \( \rho \) the fluid density, and \( e \) the total energy.

We now will solve the Navier-Stokes equations in a time-dependent domain \( \Omega(t) \) by discretizing on a grid whose points may be moving with velocity \( \mathbf{U}^g \), which is, in general, different than the local fluid velocity. This is the so-called arbitrary Lagrangian Eulerian (or ALE) formulation which reduces to the familiar Eulerian and Lagrangian framework by setting \( \mathbf{U}^g = 0 \) and \( \mathbf{U}^g = \mathbf{u} \), respectively [40, 59, 37, 24, 38, 70].

In the following sections, we will first review the ALE formulation for the advection contribution as given in [52] (we need not give special consideration to the viscous terms

25

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
because the viscous contributions do not depend on the grid velocity $U^g$). We will then present a graph theory based algorithm which is independent of the element type for updating the grid velocity $U^g$.

### 3.1 Discontinuous Galerkin ALE for Advection

Using the Reynolds transport theorem we can write the Euler equations in the ALE framework following the formulation proposed in [70] as

$$
\vec{U}_t + G_{i,t} = -U^g_i \vec{U}.
$$

(3.2)

where the ALE flux term is defined as

$$
G_i = (u_i - U^g_i)\vec{U} + \rho[0, \delta_{i1}, \delta_{i2}, \delta_{i3}, \delta_{i4}]^T, \quad i = 1, 2, 3.
$$

We can recover the Euler flux $F$ (see equation 3.1) by setting $U^g = 0$, and in general we have that $G_i = F_i - U^g_i \vec{U}$. Now if we write the ALE Euler equations in terms of the Euler flux then the source term on the right-hand-side of equation (3.2) is eliminated, and we obtain:

$$
\vec{U}_t + F_{i,t} - U^g_i \vec{U} = 0.
$$

(3.3)

which can then be recast in the standard quasi-linear form

$$
\vec{U}_t + [A_i - U^g_i I] \vec{U} = 0.
$$

(3.4)

where $A_i = \partial F_i / \partial \vec{U}$ ($i = 1, 2, 3$) is the flux Jacobian and $I$ is the unit matrix. In this form it is straightforward to obtain the corresponding characteristic variables since the ALE Jacobian matrix can be written

$$
A_i^{ALE} \equiv [A_i - U^g_i I] = R_i \cdot [A_i - U^g_i I] \cdot L_i
$$

where brackets denote matrix. Here the matrix $\Lambda$ contains on the diagonal the eigenvalues of the original Euler Jacobian matrix $A$, and $R$ and $L$ are the right- and left-eigenvector
matrices, respectively, containing the corresponding eigenvectors of $A$. Note that the shifted eigenvalues of the ALE Jacobian matrix do not change the corresponding eigenvectors in the characteristic decomposition.

![Figure 3.1: Notation for a triangular element.](image)

To explain the discontinuous Galerkin ALE formulation we consider the two-dimensional equation for advection of a conserved scalar $q$ in a region $\Omega(t)$

$$\frac{\partial q}{\partial t} + \nabla \cdot F(q) - U^g \cdot \nabla q = 0.$$  

In the discontinuous Galerkin framework, we test the equation above with discontinuous test functions $v$ separately on each element $(e)$ (see also [53, 13]) to obtain

$$(v, \partial_t q)_e + (v, \nabla \cdot F(q))_e - (v, U^g \cdot \nabla q)_e$$

$$+ \int_{\partial \Omega_e} v^i \left[ \tilde{f}(q_i, q_e) - F(q) - (q_u - q_i) \cdot U^g \right] \cdot n \, ds = 0.$$  \hspace{1cm} (3.4)

Here $(\cdot, \cdot)_e$ denotes inner product evaluated over each element $e$, and $\tilde{f}$ is a numerical boundary flux [53]; the notation is explained in figure 3.1.

To compute the boundary terms, we follow an upwind treatment based on characteristic-
tics similar to the work in [53], including here the term representing the grid motion. We need to linearize the ALE Jacobian normal to the surface, i.e. \([A - U^g_n] = R[A - U^g_n]L\), where \(U^g_n\) is the velocity of the grid in the surface normal direction. The term \((q_{up} - q_1)\) expresses a jump in the variable at inflow edges of the element resulted from an upwind treatment. In the case of a system of conservation laws the numerical flux \(\hat{f}\) is computed from an approximate Riemann solver [53].

### 3.2 Grid Velocity Algorithm

The grid velocity is arbitrary in the ALE formulation, and therefore great variety exists in the choice of technique for updating it. Mesh constraints such as smoothness, consistency, and lack of edge crossover, combined with computational constraints such as memory use and efficiency dictate the update algorithm used. Two broad classifications of algorithms exist for updating the mesh: Velocity smoothing methods and coordinate smoothing methods.

Typically, in velocity smoothing methods the grid velocity \(U^g\) is updated by solving

\[
\nabla \cdot (k(x)\nabla U^g) = 0.
\]

with Dirichlet conditions for \(U^g\) on both the moving wall boundary and on the outer boundary of the computational domain. The choice \(k = 1\) leads to the classic elliptic velocity smoothing which produces the most uniform deformation of the elements. Since in most computational fluid applications Poisson solvers are necessary, the choice of a Laplacian velocity smoother is natural due to its straightforward implementation. Though this method produces the most uniform deformation of elements, even small body motions can lead to edge crossover as demonstrated in [52] which will render the computation unstable.

Modifications to this approach were presented in [50] where a variable diffusivity \((k(x)\) being a function of position within the mesh) was introduced to help avoid edge crossing. Contrastingly, other researchers have attempted to calculate the mesh deformation using coordinate smoothing methods [8], [62] and [14]. Mesh positions are obtained using
methods based on a graph theory analogy to the spring problem. Vertices are treated as nodes, while edges are treated as springs of varying length and tension. At each time step, the mesh coordinate positions are updated by equilibration of the spring network. Once the new vertex positions are calculated, the mesh velocity is obtained through differences between the original and equilibrated mesh vertex positions.

![Graph showing vertices with associated velocities and edges with associated weights](image)

Figure 3.2: Graph showing vertices with associated velocities and edges with associated weights

In the current work, for updating the grid velocity of the mesh we combined the two concepts mentioned above by formulating the problem of solving for the mesh velocity in terms of its graph theory equivalent problem. Specifically, we incorporate the idea of variable diffusivity as in [50] while maintaining the computational efficiency of the methods used in [8], [62] and [14]. The combination of these two methodologies provides a computationally efficient way of minimizing edge crossover in situations where Laplacian smoothing fails.

The method we use for updating the mesh velocity is a variation of the barycenter method [9] and relies on graph theory. Given the graph $G = (V,E)$ of element vertices $V$ and connecting edges $E$, we define a partition $V = V_0 \cup V_1 \cup V_2$ of $V$ such that $V_0$ contains all vertices affixed to the moving boundary, $V_1$ contains all vertices on the outer boundary of the computational domain, and $V_2$ contains all remaining interior vertices. To create the effect of variable diffusivity, we use the concept of layers. As is pointed in [50], it is desirable for the vertices very close to the moving boundary to have a grid velocity almost equivalent to that of the boundary. Hence, locally the mesh appears to move with solid
movement, whereas far away from the moving boundary the velocity must gradually go to zero. To accomplish this in our formulation, we use the concept of local tension within layers to allow us to prescribe the rigidity of our system. Each vertex is assigned to a layer value which heuristically denotes its distance from the moving boundary. Weights are chosen such that vertices closer to the moving boundary have a higher influence on the updated velocity value. To find the updated grid velocity \( u^g \), at a vertex \( v \in V_2 \), we use a force-directed method. Given a configuration as in figure 3.2, the grid velocity at the center vertex is given by:

\[
\begin{align*}
\sum_{i=1}^{\text{deg}(v)} \alpha_i^l u_i^l, & \quad \sum_{i=1}^{\text{deg}(v)} \alpha_i^l = 1,
\end{align*}
\]

where \( \text{deg}(v) \) is the number of edges meeting at the vertex \( v \) and \( \alpha_i^l \) is the \( l \)th layer weight associated with the \( i \)-th edge. This is subjected to the following constraints: \( u^g = 0 \) (\( \forall v \in V_1 \)), and \( u^g (\forall v \in V_0) \) is prescribed to be the wall velocity. This procedure is repeated for a few cycles following an incomplete iteration algorithm, over all \( v \in V_2 \). Once the grid velocity is known at every vertex, the updated vertex positions are determined using explicit time-integration of the newly found grid velocities.

### 3.3 Flow Simulations

#### 3.3.1 2D Flow Around a Pitching Airfoil

We consider flow around a pitching NACA 0015 airfoil and compare lift coefficients against the established computational results of Visbal et al. [71]. The NACA 0015 airfoil is pitching upwards about a fixed axis at a constant rate from zero incidence to a maximum angle of attack of approximately 60 degrees. The pivot axis location is at 1/4 of the chord measured from the leading edge. The temporal variation of the pitch given in [71] is

\[
\Omega(t) = \Omega_0 [1 - e^{-4.6t/b}], \quad t \geq 0
\]
where $t_0$ denotes the time elapsed for the airfoil to reach 99% of its final pitch rate $\Omega_0$. Here the non-dimensional values are $t_0^* = 0.5$ and $\Omega_0^* = 0.2$ based on the chord length and free stream velocity. As initial condition the computed field at zero degrees angle of attack is used. The Mach number is 0.2 and the chord Reynolds number is 45,000.

The computational mesh used for this simulation consisted of 912 quadrilateral elements: the full domain and local tessellation are shown in figure 3.3.

![Figure 3.3: Full domain (left) and local tessellation (right) for the simulation around the NACA 0015 pitching airfoil. All dimensions are in units of chord length.](image)

In [71] a similar simulation was performed using a grid fixed to the airfoil by employing an appropriate transformation and discretizing the modified compressible Navier-Stokes equations using the implicit approximate factorization of Beam and Warming [11]. A typical grid used in [71] involved $203 \times 101$ points. Although accurate, this approach is not general for moving domains and cannot be used, for example, in simulating multi-body dynamics.

In the present study, we employ the ALE formulation on the domain shown in figure 5.8. We plot the computed lift coefficient versus the angle of attack using 9th order polynomials with over-integration (see section 5). We also include the computational results (denoted with circles) and experimental results (denoted with asterisks) presented in [71].

Observe that the coefficient of lift as we have computed contains many more wiggles than that of the computation of Visbal. We attribute this to the fact that we are using a
Figure 3.4: Comparison of the lift coefficient $C_L$ versus angle of attack in degrees for the 9th order case (solid line) versus the computational and experimental results presented in [71] (symbols). The chord Reynolds number is $Re = 45,000$, and the Mach number is $Ma = 0.2$.

much higher resolution than the original computation of Visbal. Personal communications with Visbal have verified that new computations at higher-order show similar features as our computation. In figure 3.5 we present density contours for three different angles of attack, $\alpha = 10.4$ (left), $\alpha = 26.1$ (center), and $\alpha = 52.2$ (right).

The above results were obtained by prescribing the grid velocity $U^g$ so that the entire grid moves with the airfoil in a rigid body rotation. In this case, there is no grid distortion as in the method in [71]. In [52], a similar experiment as this was one was performed for a pitching NACA 0015 at Reynolds number $Re = 10,000$. In that study, comparisons of a no grid distortion case versus a grid distortion case in which the mesh does not move as a rigid body were presented. It was demonstrated that the two computations obtain similar results for lift and drag up to the point were the mesh becomes tangled (in the mesh distortion case).
3.3.2 3D Flow Around A Wing with Endplates

We consider flow past a NACA 0012 airfoil with plates attached to each end as a simple model of a wing between an engine and fuselage. We impose uniform upwind boundary conditions at the inflow and outflow, and the domain is periodic from one end of the airfoil to the other. A thin layer of hexahedra was used on the surface of the wing, and a combination of both hexahedra and prisms were used in the remainder of the computational domain. The simulation was run with up to $4^{th}$ order expansion at $Re = 2000$ (based on chord length). A summary of the simulation parameters is given in table 3.1, and the hybrid grid and representative results are shown in 3.6.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>3</td>
</tr>
<tr>
<td>$Re$</td>
<td>2000 based on chord length</td>
</tr>
<tr>
<td>Mach</td>
<td>0.5</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>$1e^{-4}$</td>
</tr>
<tr>
<td>P-Range</td>
<td>1 to 4</td>
</tr>
<tr>
<td>$K_{prisms}$</td>
<td>1960</td>
</tr>
<tr>
<td>$K_{hex}$</td>
<td>2095</td>
</tr>
<tr>
<td>Method</td>
<td>Discontinuous Galerkin</td>
</tr>
</tbody>
</table>

Table 3.1: Simulation parameters for compressible flow past a NACA 0012 airfoil with endplates.

At low-order the simulation ran to steady state. This is due to a reduced effective
Figure 3.6: Skeleton mesh for flow past a three-dimensional NACA 0012 airfoil with endplates (top and middle). Iso-contours for x-component of momentum for $M = 0.5$ flow past a three-dimensional NACA 0012 airfoil with endplates (bottom).
Reynolds number achieved because of numerical dissipation. As we increased the order we saw unsteadiness developing in the wake of the wing, and what appears to be oblique shedding. This is only a marginally three-dimensional domain, but it does demonstrate the ability of NekTAR to direct resolution into boundary layers and to fill out a domain with larger elements.

We now present in figure 3.7 a moving case of the wing above with a prescribed motion. For this case, we set the $x$ and $z$ direction motion to zero, and force the structure to move with a structural velocity given by:

$$\frac{d}{dt} y(x, y, z, t) = A_0 \cos(\Omega_0 t) \sin(K \pi z^*)$$

where $A_0 = 0.5$, $\Omega_0 = \pi/2$, $K = 1$, and $z^* \in [0, 1]$ is a normalized variable along the length of the wing.

![Figure 3.7: Initial configuration (left) and deflected (right) NACA 0012 wing section between two end-plates. Iso-contours of streamwise momentum are shown.](image)

This ALE case was used to verify the implementation of the graph algorithm, and as a verification of the sustainable mesh deformation prior to using this mesh for the fluid/structure problem. The stationary run described above was used as the initial condition for both the prescribed motion ALE run just presented and as the initial condition for the fluid/structure coupling presented in the next chapter.
Chapter 4

Fluid-Structure Interaction for Compressible Flows

Aircraft under realistic flight conditions experience tremendous variations in the aero-
dynamic loading applied to the structure of the aircraft. This buffeting of the aircraft
structure can significantly reduce the service life of the aircraft. In addition, resonance
conditions in the form of panel flutter may arise: such sustained oscillations within the
flight envelope may lead to catastrophic structural failure. Modeling of the buffeting
regime requires both an accurate approximation of the dynamic loading produced by the
fluid as well as a realistic model of the structure. Many efforts have been focussed on
the fluid-structure coupling problem for solving real-world aircraft configuration problems
such as the F-18 [64], the P-3 [63], and the F-16 [27]. Concurrent with these efforts have
been developmental work on computational systems for studying the aeroelastic response
of rapidly maneuvering aircraft [29]. Each of these efforts requires simplifying assumptions
to be made either in the fluid or the structural solver. On the part of the fluid solvers,
asumptions like using the Euler equations or different turbulence models are used to sim-
plify the fluid computation. On the part of the structure, thin solid structures are modeled
with combinations of beam, plate and shell elements, each of which are formulated upon
certain assumptions. As was pointed out in [25], simplifications in both the fluid and the
structure have led to advancement in the area of fluid-structure interaction, however the
assumptions under which these simplifications are made lead to an idealization error - error which needs to be quantified.

In an effort to minimize this idealization error, we have proposed the use of a parallel spectral/hp element fluid solver, which enables a good and fast resolution of the flow field, linked to a hp-FEM structural solver, which enables a realistic representation of thin solid structures, as a natural choice for simulating such situations as wing structures in flight. To this end, we coupled the discontinuous Galerkin viscous compressible fluid solver \textit{NekTAR} with the hp-version finite element structural solver StressCheck\footnote{StressCheck is a Trade Mark of Engineering Software Research & Development, Inc., 10845 Olive Blvd., St. Louis, MO, USA}. Several strategies for coupling the two hp-FEM codes exist, two of which will be discussed here. The first simplified approach which can simulate only linear elastic structures is based on a one-way coupling, for which results are reported. The more general and realistic approach, which may represent both geometrical as well as material nonlinearities in the structure, is based on a two-way coupling of the spectral/hp element fluid solver \textit{NekTAR} with the hp-version finite element structural solver StressCheck.

A brief description of the structural solver StressCheck is provided first. We then present a discussion and preliminary results of the one-way coupling method, and present a discussion of the implementation of the two-way coupling method examined. We conclude with discussing some of the open issues which remain to be researched.

\renewcommand{\thefigure}{4.1}

\section{The Structural Solver - StressCheck}

StressCheck is a commercially available hp-version finite element solver (see \cite{67} for details of StressCheck underlying theory) for accomplishing linear and nonlinear structural analysis. It has been chosen because of its superiority in several aspects: a) it provides an error estimator to assure the accuracy of the computed data, b) it provides 3-D "thin solid" elements \cite{68} and represents well shell and plate like structures, without the need of modeling assumptions usually used in other codes (shell elements, beam element and alike that may introduce large modeling errors), c) it enables the use of elements with huge aspect ratios, a mandatory request in the simulation of thin wall structures, d) it is the
only p-FEM code enabling geometric non-linear capabilities [58]. (wings typically undergo large deformations during flight), e) it provides a Component Object Model (COM) interface (a functional interface for accessing StressCheck functionality) through which the coupling to the fluid solver is convenient.

4.2 Fluid-Structure Coupling

We examined the fluid-structure problem of flow past an airfoil between two fixed endplates as introduced in section 3.3.2. (see figure 3.7). The discontinuous Galerkin method is used for solving the fully three-dimensional viscous compressible Navier-Stokes equations. The fluid-structure coupling is achieved by an Arbitrary Eulerian Lagrangian (ALE) formulation as discussed in chapter 3, which allows "arbitrary" motion of the fluid-structure interface. As a post-solution operation at a time instant, the pressure and shear stress distributions on the structure are extracted. The interaction is then accomplished by incorporating these dynamic loads experienced by the structure with the resultant deformation obtained through the structural analysis by StressCheck. Two coupling situations, one-way coupling and two-way coupling, are considered. We now present a discussion of these two coupling situations.

4.2.1 One-way Coupling

When the solid is assumed to behave in a linear elastic manner, we can accomplish a one-way coupling of the fluid solver and the structural solver as illustrated in figure 4.1. In this circumstance, StressCheck is used to formulate an influence matrix based upon unit normal loading applied to each sub-panel of the structure. Once this influence matrix is formed, it is used by NekTar for updating the displacement of the structure based upon the dynamic loading with no additional communication with StressCheck.

Implementation Details

We now present a more detailed description of how this coupling occurs. In this first stage, the geometric model of the airfoil was created through the StressCheck GUI interface.
Figure 4.1: Diagram showing the one-way coupling of *NekTar* with StressCheck.

having a surface partitioning coinciding with the surface discretization used by *NekTar*. This allowed a one-to-one correspondence between the vertices of the macro-elements in both the fluid and the structural mesh. A screen snapshot of the model used is given in figure 4.2 (left). Zero displacement boundary conditions are applied to the ends of the airfoil at the airfoil/end-plate juncture. Future development of the coupled fluid-structure interface will enable a solid mesh completely independent of the mesh used in the fluid solver.

The following procedure was used to form the influence matrix. For each sub-panel on the airfoil surface, a unit normal load was applied and the corresponding deformation computed using StressCheck as shown in figure 4.2 (right). A total of 270 load cases were computed using a polynomial order of four (which ensures a small discretization error in energy norm). From the collected deformations induced by the 270 unit normal load cases we formed an influence matrix which when post-multiplied by a given pressure distribution, represented by a vector containing a collection of loads, yields the deformation of the structure due to that given pressure distribution.

Once the influence matrix is formed, it is read into *NekTar* at the beginning of the simulation and used each time the structural deformation requires updating. The
Figure 4.2: Screen snap-shot showing the model created in StressCheck (left); Screen snap-shot showing the deformation of the structure based upon unit normal loading on one sub-panel of the structure (right)

advantage of this coupling scheme is that all the structural analysis can be accomplished in a pre-processing stage, hence eliminating all communication between NekTar and StressCheck once the time integration of the fluid has begun. This advantage though is at the sacrifice of geometric non-linearity of the structure, which quite often is an important modeling component of the full fluid-structure problem. To this end, we will present a two-way coupling algorithm in section 4.2.2

Preliminary Results

We now present computational results from the one-way coupling of NekTar and StressCheck as described above. In figure 4.3 we represented the time history of the displacement of a point on the surface of the airfoil. Though small in magnitude, this plot demonstrates the small variations that occur in the position of the structure when the fluid-structure model is employed.

In figure 4.4 we present the time history of the coefficient of lift for the stationary airfoil configuration versus the fluid-structure configuration. The dynamic loading on the structure is modified when the fluid-structure coupling is allowed.
Figure 4.3: Time history of the displacement of a point on the airfoil. Displacement and time are both in non-dimensional units.

Figure 4.4: Time history of the coefficient of lift for a stationary airfoil configuration (solid) versus the fluid-structure model (dashed).
4.2.2 Two-way Coupling

When geometric non-linearity of the structure is assumed, we cannot accomplish a one-way coupling of the fluid solver and the structural solver through an influence matrix, but instead must solve the fluid-structure problem in a fully-coupled manner. A diagram outlining the two-way coupling is presented in figure 4.5. Different fluid-structure coupling algorithms such as staggering and subcycling were investigated following the work of [28].

We now present a discussion of the coupling issues arising in the two-way coupling.

![Diagram showing the two-way coupling of NekTar with StressCheck.](image)

Two-way coupling of NekTar and StressCheck required us to create a software solution for extracting loading information from the fluid solver and providing it to StressCheck. At this stage, a non-linear elastic analysis is performed, and the deformations on the structure faces are passed back to NekTar for updating the position of the structure within the fluid. This can be achieved by software integration in such a way that both codes can interact dynamically.

In order to maintain realistic computational running times for the fluid simulation, large-scale distributed-shared memory parallel machines were employed, most of which use a Unix/Linux operating system. StressCheck, however, runs only under the Microsoft Windows environment (the Microsoft .NET technology would allow in the future function-
ality to be accessed independent of the platform). However, at present, a software solution had to be devised to overcome this obstacle. The current software implementation of the coupling requires that the machine on which the fluid solver executes and the machine on which the structural solver executes share a common file system. Using StressCheck's Component Object Model (COM) interface, a Visual Basic server application was developed which is connected to by the fluid solver via sockets. The StressCheck server is initiated on its host PC while the fluid solver is executed on a parallel Unix machine.

At some specified interval, the fluid solver computes the loading on the structure and saves this information to a file. The fluid solver then signals the StressCheck server through the socket interface that it may now proceed to compute the displacement of the structure due to the newly updated loading. Once the structural solver has completed its calculation, it writes the updated displacement information to a file and notifies the fluid solver through the socket interface that the fluid simulation may proceed using the newly updated displacement information.

4.3 Open Issues

The work described above represents the initiation of an on-going effort to combine NekTar and StressCheck. Many areas of research still need to be explored. For instance, an understanding of modeling error should be accomplished by performing a geometrical non-linear analysis of a typical wing in flow, and comparing the results to a wing as if it behaves linearly elastic. This would provide a quantitative measure as to the role of large deformations on the validity of results obtained under the assumption of linear elasticity (which is usually used in such simulations). In the same spirit, the modeling assumptions of plate and rods elements instead of realistic 3-D elements should be quantified and addressed. From the algorithmic point of view, several staggered algorithms should be investigated, and the most efficient and accurate one adopted. Finally, this newly created fluid-structure hp-based code could be used in the future for addressing questions related to survivability of aircraft when parts of the structure are hit or torn.
Chapter 5

Polynomial De-aliasing

In spectral methods the quadratic nonlinearities in the incompressible Navier-Stokes equations or the nonlinearities in the compressible Navier-Stokes (which are either cubic if written in primitive variables or of a rational form if written in conservative variables) are computed in the physical space. Specifically, the fields (velocity, pressure, energy) are first transformed into physical space, and subsequently the products are obtained at all quadrature points in a collocation fashion. Another transform is then performed to bring the results back to modal space. More specifically, when the number of quadrature points $Q$ is the same as the number of modes in the spectral expansion $P$ we have a true collocation method, otherwise for $Q > P$ we have a super-collocation method.

Errors may be caused by insufficient quadrature used in the spectral/$hp$ element discretization of the nonlinear terms, especially in complex-geometry flows. Theoretical results in [18] provide a minimum requirement for the precision of numerical quadrature when computing volume and boundary integrals in the discontinuous Galerkin method. This result provides only a necessary condition on the number of quadrature points used. The errors caused by insufficient quadrature can be bounded by the theoretical results in [17], and quite often this result is used to support the idea that if the simulation is well-resolved, then the numerical crimes committed by insufficient quadrature are negligible. As was pointed out in [16], however, this does not address what happens in the marginally resolved simulation, which is far more often the case than most are willing to admit. Studies have also been performed to understand the effect of over-integration

44

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
when curvature is employed (see [56],[22]). however these studies do not address directly
the effect on the integration of the non-linear terms. We propose that these errors can be
eliminated effectively by employing over-integration, i.e., integrating the nonlinear terms
in the variational statement with higher order quadrature than the one employed for the
linear contributions, e.g., pressure and viscous terms in an incompressible flow simulation.
We will examine this issue in some detail next.

5.1 Accuracy, Stability and Over-Integration

From [45] we know the following numerical quadrature rule for Gauss-Lobatto-Legendre
quadrature: Given $Q$ quadrature points and corresponding weights (we will assume from
hence forth that whenever referring to the number of quadrature points we also are refer­
ring to the corresponding weights), we can integrate a polynomial $u(\xi) \in P_{2Q-3}$ exactly
(to within machine precision computationally). From this rule we can compute for any
given polynomial $u(\xi) \in P_N$ the number of quadrature points $Q$ as a function of $N$ nec­
essary for the quadrature to be exact. In table 5.1 we present the number of quadrature
points necessary given the polynomial order of the integrand for functions in $P_N$, $P_{2N}$,
$P_{3N}$, and $P_{4N}$.

<table>
<thead>
<tr>
<th>Polynomial Order $N$</th>
<th>Number of Quadrature Points $Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(\xi) \in P_N$</td>
<td>$Q = \lceil (N + 3)/2 \rceil$</td>
</tr>
<tr>
<td>$u(\xi) \in P_{2N}$</td>
<td>$Q = \lceil (2N + 3)/2 \rceil$</td>
</tr>
<tr>
<td>$u(\xi) \in P_{3N}$</td>
<td>$Q = \lceil (3N + 3)/2 \rceil$</td>
</tr>
<tr>
<td>$u(\xi) \in P_{4N}$</td>
<td>$Q = \lceil (4N + 3)/2 \rceil$</td>
</tr>
</tbody>
</table>

Table 5.1: Number of quadrature points necessary for Gauss-Lobatto-Legendre quadrature
to be exact for given polynomial orders of the integrand.

In Galerkin methods, it is often the case that we are interested in computing nu­
merically the inner product of two polynomials of the same (or lesser) degree, i.e., the
inner product $(\phi_i, \phi_j)$ where $\phi_i, \phi_j \in P_N$. A polynomial of degree at most $N$ has at
most $M = N + 1$ modal coefficients in its expansion. We can thus compute the num­
ber of quadrature points $Q$ as a function of $M$ necessary to integrate multiple powers of
\[ u(\xi) \in P_N \] exactly. In table 5.2 we present the number of quadrature points necessary as a function of \( M \) given the polynomial order of the integrand for polynomials in \( P_{2N} \), \( P_{3N} \), and \( P_{4N} \).

<table>
<thead>
<tr>
<th>Polynomial Order ( N )</th>
<th>Number of Quadrature Points ( Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>([u(\xi)]^2 \in P_{2N})</td>
<td>(Q = \lceil M + 1/2 \rceil)</td>
</tr>
<tr>
<td>([u(\xi)]^3 \in P_{3N})</td>
<td>(Q = 3M/2)</td>
</tr>
<tr>
<td>([u(\xi)]^4 \in P_{4N})</td>
<td>(Q = \lceil 2M - 1/2 \rceil)</td>
</tr>
</tbody>
</table>

Table 5.2: Number of quadrature points necessary for Gauss-Lobatto-Legendre quadrature to be exact in terms of the number of modes for the original expansion for given polynomial orders of the integrand.

From the table above we see that for linear problems which need only the inner product of two polynomials of the same or lesser degree (such as the viscous terms of the incompressible Navier-Stokes equations), only \( M + 1 \) quadrature points are needed to integrate exactly. In the case of quadratic non-linearities such as the non-linear terms in the incompressible Navier-Stokes equations, \( 3M/2 \) points are needed, and for cubic non-linearities such as those found in the compressible Navier-Stokes equations, \( 2M \) points are needed for the numerical integrations to be exact.

Most numerical solvers, however, only use the number of quadrature points necessary to integrate the linear terms exactly. To understand the ramifications of under-integration of nonlinear terms, we perform the following test:

1. Consider a single element in the space interval \([-1, 1]\) containing \( M = 16 \) Jacobi modes.
2. Initialize all the modal coefficients to one.
3. Evaluate the modal representation on a set of \( Q \) quadrature points.
4. Square (in a pointwise fashion) the values at the quadrature points.
5. Pre-multiply the set of points (as a vector) by the collocation derivative matrix of the appropriate size (rank \( Q \times Q \)).
6. Project back to modal coefficients by discrete inner products using Gaussian integration.

The procedure above mimics the “physical space” or pseudo-spectral evaluation of the term $\frac{\partial u^2}{\partial x}$ commonly used in spectral methods for evaluating nonlinear terms. This test was chosen because even in its simplicity it models the order of nonlinearity that occurs in the solution of the incompressible Navier-Stokes equations. All modes are set to one to mimic a case in which an element has under-resolved or marginally resolved the solution within the element. In the test above, the only unspecified parameter is the number of quadrature points $Q$ to be used. In using Gauss-Lobatto-Legendre points, the value of $Q$ is taken to be one more than the number of modes $M$ (in this case then $M = 16$ and $Q = 17$) ([52]), but this value is appropriate for the inner products corresponding to linear terms. For quadratic or cubic nonlinearities more quadrature points are required as discussed above. The ramifications of under-integration of this form are shown in figure 5.1. The figure on the left was obtained for quadratic nonlinearity ($\frac{\partial}{\partial x} u^2$) and the figure on the right was obtained for a cubic nonlinearity ($\frac{\partial}{\partial x} u^3$). The difference in the modal coefficients at the conclusion of the algorithm above for different values of $Q$ is provided. We observe that for the quadratic nonlinearity, once $\frac{3}{2}M$ quadrature points are used, the differences in the modal values do not change. Similarly for the cubic nonlinearity, once $2M$ quadrature points are used, the differences in the modal values do not change.

Note that the quadrature rules above for quadratic and cubic non-linearities mimic the de-aliasing rules used for the Fourier method [16]. Although the “rules” appear to give the same result, the rationale behind the rules is slightly different. The goal of polynomial de-aliasing is to find the necessary number of quadrature points so that the computation of the variation form of the non-linear terms is exact.

### 5.2 Over-integration Study using Burgers Equation

To further test the integration of the non-linear terms, we chose to solve viscous Burgers equation:
5.2.1 Continuous Galerkin Method

In this example we present the continuous Galerkin ($C^0$) solution of the viscous Burgers equation with $\nu = 10^{-5}$. In these examples, Gauss-Lobatto-Legendre points are used for each element. The simulation parameters are presented in table 5.3: we draw your attention to the fact that five equally spaced elements were used, each of which contained $M = 16$ modes in its polynomial expansion. The number of elements and the element spacing were chosen so that the steep gradient in the solution occurs in the middle of an element and not at an element interface. This choice forces the middle element to accommodate the steep gradient by involving most of the modes in its elemental polynomial expansion, hence making it a prime candidate for aliasing problems. The rationale for this example is to test whether the solution of a system with a little bit of viscosity is sufficient to overcome the "numerical crimes" committed by under-integrating the non-linear terms, and hence render the solution stable.

If $Q = M + 1 = 17$ quadrature points are used for integrating both the advection and diffusion terms, the solution is unstable. If, however, $Q = 3M/2 = 24$ quadrature
Table 5.3: Parameters used for the results presented in figure 5.2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Continuous Galerkin</td>
</tr>
<tr>
<td>Number of Elements</td>
<td>5</td>
</tr>
<tr>
<td>Modes</td>
<td>16</td>
</tr>
<tr>
<td>Time Integrator</td>
<td>2nd order Adams-Bashforth</td>
</tr>
<tr>
<td>Final Time T</td>
<td>0.5</td>
</tr>
<tr>
<td>Δt</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

points are used as in figure 5.2 (top), the solution is stable. To verify that indeed the over-integration is only necessary for the advection (non-linear) terms and not the diffusion terms (which because they are linear should only require \(M + 1\) points for exact integration), we also computed the solution when \(Q = 24\) points were used for integrating the advection term and \(Q = 17\) points were used for integrating the diffusion term. The result is presented in figure 5.2 (bottom). The difference in the \(L_\infty\) norm of the solution is machine zero.

![Figure 5.2](image)

Figure 5.2: Solution of the viscous Burgers equation with \(\nu = 10^{-5}\) evaluated at \(T = 0.5\). In A, \(Q = 24\) quadrature points are used for integrating both the advection and diffusion terms; and in B, \(Q = 24\) quadrature points are used for the advection term, and only \(Q = 17\) points are used for the diffusion terms.

To further extend our understanding of the problems which may arise due to under-integration, we also solved the viscous Burgers equation with \(\nu = 10^{-2}/\pi\). Our rationale in this example is to see how much viscosity is necessary to stabilize the solution when
under-integration is involved. Three cases were examined (and are plotted in figure 5.3):

1. Case 1: \( Q = 17 \) quadrature points used for integrating both the advection and diffusion terms (top figure)

2. Case 2: \( Q = 24 \) quadrature points used for integrating both the advection and diffusion terms (middle figure)

3. Case 3: \( Q = 24 \) points used for integrating the advection term and \( Q = 17 \) points used for integrating the diffusion term. (bottom figure)

![Figure 5.3: Solution of the viscous Burgers equation with \( \nu = 10^{-2}/\pi \) evaluated at \( T = 0.5 \). In A, \( Q = 17 \) quadrature points are used for integrating both the advection and diffusion terms; in B, \( Q = 24 \) quadrature points are used for integrating both the advection and diffusion terms; and in C, \( Q = 24 \) quadrature points are used for the advection term, and only \( Q = 17 \) points are used for the diffusion terms.](image)

In table 5.4 we present the error in the discrete \( L_\infty \) norm for the three cases presented above. Observe that both case two and case three show marked improvement over case one. The small discrepancy between case two and case three can be explained by the fact that the discrete (pointwise) \( L_\infty \) norm is taken over the number of points used for the diffusion operator. Hence in case two more points were used for the diffusion operator than in both case one and case three. The overall trend, and hence the point of the test, was not affected however.

As is evident by these examples, given sufficient viscosity, the solution can be rendered stable. However, since our interest is in the numerical simulation of high Reynolds number
Table 5.4: Error in the discrete $L_\infty$ norm for the three cases presented above.

<table>
<thead>
<tr>
<th>Case</th>
<th>Error in the discrete $L_\infty$ norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>0.5090</td>
</tr>
<tr>
<td>Case 2</td>
<td>0.2729</td>
</tr>
<tr>
<td>Case 3</td>
<td>0.2038</td>
</tr>
</tbody>
</table>

flows, special care must be taken when integrating the non-linear terms, as the viscosity may be insufficient in rendering the solution stable.

5.2.2 Discontinuous Galerkin Method

In order to appreciate the effect of under-integration in the context of a numerical solution obtained using the discontinuous Galerkin method, we consider the inviscid Burgers equation, which we discretize using the discontinuous Galerkin method. The initial condition is $u(x) = -\sin(\pi x)$, and five equally spaced elements spanning $[-1, 1]$ were used, each one having $M = 16$ modes. In figure 5.1, we plot the $L_2$ norm of the solution versus the number of quadrature points used for numerical integration. When using $Q = 17, 19$ and $Q = 21$ points, the solution is unstable (denoted by the *). Once the number of quadrature points reaches $Q = 24 (\frac{3}{2}M$ where $M$ is the number of modes), the $L_2$ norm of the solution does not change. The alternating unstable/stable pattern exhibited for $Q = 17$ through $Q = 22$ may be due to the fact that for some elements only the even modes are excited (this issue will be demonstrated below). As the quadrature order is increased, different mode combinations are integrated properly; only after a sufficient number of quadrature points is used ($Q > 21$) does the solution remain stable for all $Q$.

We can analyze this behavior by examining the energy in the modes (denoted by the square of the modal values) within the element that contains the jump in the inviscid Burgers solution. The modes were extracted at time $T = 0.35$, after the shock has formed (at time $\frac{1}{2}$) and prior to the solution becoming unstable. In figure 5.5, we plot the square of the modal coefficients versus the mode number. Due to the symmetry of the element placement, only even number modes were excited.

The case corresponding to using $Q = 17$ quadrature points will become unstable by
Figure 5.4: Solution of the inviscid Burgers equation evaluated at $T = 0.5$. Five equal spaced elements were used with 16 modes in each element. On the ordinate we plot the $L_2$ norm of the solution, and on the abscissa we plot the number of quadrature points used for numerical integration. Unstable solutions are denoted by *. Observe that after $Q = 24$ points, the $L_2$ norm of the solution does not change.

time $T = 0.5$. If a $\frac{3}{2}M$ rule is used, yielding $Q = 24$ points, the solution is stable, and the energy is much less than when the non-linear terms are under-integrated. Hence when no over-integration is used, the energy in the highest modes grows, indicating an aliasing error. When over-integration is applied, however, the modal coefficients of the inviscid Burgers solution converge monotonically to zero leading to a stable simulation unlike the untreated simulation. This plot shows vividly the effects of aliasing when under-integration of the non-linear terms is performed.

5.3 Polynomial De-Aliasing in Flow Simulations

5.3.1 Transition and Turbulence in a Triangular Duct

We demonstrate next the effect of under-integration and associated aliasing errors by simulating transition to turbulence of incompressible flow in a duct with its cross-section being an equilateral triangle. The laminar fully-developed solution is known analytically. We introduce some random disturbances in the flow, and we integrate in time until these disturbances start decaying or growing in time. All simulations were performed in the domain shown in figure 5.6 with the cross-section discretized using one triangular element.
Figure 5.5: Modal coefficients in the middle element of the inviscid Burgers solution at time $T = 0.35$. Over-integration using $Q = 3M/2$ quadrature points leads to a stable solution unlike the $Q = M + 1$ case.

only and 16 Fourier modes (32 collocation points) in the streamwise (homogeneous) direction. The Reynolds number is defined as $Re = U D_e / \nu$ where $U$ is the average velocity and $D_e$ is the equivalent (hydraulic) diameter. For $Re < 500$ all disturbances decay but for $Re = 1250$ the flow goes through transition, and a turbulent state is sustained.

We have performed three simulations at $Re = 1250$ corresponding to three different combinations of polynomial and quadrature order. In the first one, shown in figure 5.3.1(a), we consider the case where $Q = M + 1$, where $M = 16$. The forces on the three walls of the duct are plotted as a function of time. From symmetry considerations, we expect that the statistical averages of the three forces are identical, but obviously the symmetry in the mean is not preserved here. In figure 5.3.1(b) we plot the forces for the case with $Q = 2M$, and in figure 5.3.1(c) the case with $Q = 3M/2$. We have verified that in both cases the same statistical force average is obtained, consistent with the analysis presented above for handling under-integration induced errors.

Based on the above analysis and result as well as other similar results, we can state the following semi-empirical rule:

**De-alising Rule:** For quadratic nonlinearities employing super-collocation with $3/2M$ grid (quadrature) points per direction, where $M$ is the number of modes in one direction.
of the tensor product expansion, followed by a Galerkin projection leads to a de-aliased
turbulence simulation on non-uniform meshes.

![Figure 5.6: Duct flow domain: The cross-section is an equalateral triangle and the stream-
wise length is three times the triangle edge. On the left we show a frame of the entire
domain with flood contour cut-planes of the fluid velocity in the streamwise (w) direction. In
the center and on the right we present flood contour cut-planes of the fluid velocity in the streamwise (w) direction with arrows denoting the velocity in the crossflow (u,v) directions at \( z = 1 \) and \( z = 2 \) respectively.](image)

In table 5.5, we present the mean shear force calculated for each of the walls. Statistical
averaging was started 40 convective units (over 13 duct lengths) after the initial pertur-
bation period of 10 convective units; statistical averages were taken over 150 convective
units (50 duct lengths).

<table>
<thead>
<tr>
<th>Quadrature Order</th>
<th>Wall 1</th>
<th>Wall 2</th>
<th>Wall 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M + 1 )</td>
<td>0.0048</td>
<td>0.0054</td>
<td>0.0054</td>
</tr>
<tr>
<td>( \frac{3}{2} M )</td>
<td>0.0053</td>
<td>0.0053</td>
<td>0.0053</td>
</tr>
<tr>
<td>( 2M )</td>
<td>0.0053</td>
<td>0.0053</td>
<td>0.0053</td>
</tr>
</tbody>
</table>

Table 5.5: Mean shear forces on each wall versus the quadrature order employed.

### 5.3.2 2D Flow Around a Pitching Airfoil

To demonstrate the effect of over-integration when using the discontinuous Galerkin
method, we examine the laminar compressible flow around a rapidly pitching NACA
0015 airfoil. The details of this computation were presented previously in section 3.3.1: we
present here only those components of the simulation necessary to highlight the effect
Figure 5.7: Wall shear forces as a function of time for (a) \((Q = M + 1)\); (b) \((Q = 2M)\); and (c) \((Q = 3M/2)\).
of over-integration. The computational mesh used for this simulation consisted of 3.838 triangular elements; the full domain and local triangulation are shown in figure 5.8.

![Figure 5.8: Full domain (left) and local triangulation (right) for the simulation around the NACA 0015 pitching airfoil. All dimensions are in units of chord length.](image)

For this study three cases (labeled A-C) were computed. All three cases used the same triangulation, however different polynomial orders and number of quadrature points were used. The parameters for the three cases are presented in table 5.6.

<table>
<thead>
<tr>
<th>Case</th>
<th>Polynomial Order</th>
<th>Number of Quadrature Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9\textsuperscript{th} order</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>2\textsuperscript{nd} order</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>3\textsuperscript{rd} order</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.6: Polynomial order per tensor product direction and number of quadrature points per direction used for cases A-C.

In figure 5.9 we present the coefficient of lift $C_L$ for cases A-C with comparison to the experimental and computational results presented in [71]. Case A is our control case with high polynomial order and is stable both with and without over-integration. The coefficient of lift differs negligibly between the over-integrated and non-over-integrated cases for this high-order simulation. Cases B and C are both unstable if over-integration is not employed. Appropriate over-integration leads to a stable computation until an angle

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
of attack of approximately 37 degrees where both computations finally go unstable. Even with the extremely low resolution of these two cases, the computational results capture the general trend of the control case, only starting to deviate at high angles of attack prior to going unstable. Also observe that increasing the polynomial order draws the computational result closer to the control case indicating a p-type convergence to the control solution.

![Figure 5.9: Comparison of the lift coefficient $C_L$ versus angle of attack in degrees for Cases A-C (left) and Case A compared with the computational and experimental results presented in [71] (right).](image)

Based on the above result as well as other similar results, we can state the following semi-empirical rule:

**De-aliasing Rule:** For cubic nonlinearities employing super-collocation with $2M$ grid (quadrature) points per direction, where $M$ is the number of modes in one direction of the tensor product expansion, followed by a Galerkin projection leads to a de-aliased turbulence simulation on non-uniform meshes.
Chapter 6

Spectral Vanishing Viscosity

Tadmor [69] first introduced the concept of spectral vanishing viscosity (SVV) using the inviscid Burgers equation

$$\frac{\partial}{\partial t} u(x, t) + \frac{\partial}{\partial x} \left( \frac{u^2(x, t)}{2} \right) = 0. \quad (6.1)$$

subject to given initial and boundary conditions. The distinct feature of solutions to this problem is that spontaneous jump discontinuities (shock waves) may develop, and hence a class of weak solutions can be admitted. Within this class, there are many possible solutions, and in order to single out the physically relevant one an additional entropy condition is applied, of the form

$$\frac{\partial}{\partial t} \left( \frac{u^2(x, t)}{2} \right) + \frac{\partial}{\partial x} \left( \frac{u^3(x, t)}{3} \right) \leq 0. \quad (6.2)$$

In practical applications, spectral methods are often augmented with smoothing procedures in order to reduce the Gibbs oscillations [23] associated with discontinuities arising at the domain boundaries or due to under-resolution. However, with nonlinear problems, convergence of the Fourier method, for example, may fail despite additional smoothing of the solution. Tadmor [69] introduced the spectral vanishing viscosity method, which adds a small amount of controlled dissipation that satisfies the entropy condition, yet retains spectral accuracy. It is based on viscosity solutions of nonlinear Hamilton-Jacobi equa-
tions. which have been studied systematically in [21]. Specifically, the viscosity solution for the Burgers equation has the form

$$\frac{\partial}{\partial t} u(x,t) + \frac{\partial}{\partial x} \left( \frac{u^2(x,t)}{2} \right) = \epsilon \frac{\partial}{\partial x} \left[ Q \frac{\partial u}{\partial x} \right]. \quad (6.3)$$

where $\epsilon(\to 0)$ is a viscosity amplitude and $Q_k$ is a viscosity kernel, which may be nonlinear and, in general, a function of $x$. Convergence may then be established by compensated compactness estimates combined with entropy dissipation arguments [69]. To respect spectral accuracy, the SVV method makes use of viscous regularization and equation (6.3) may be rewritten in discrete form (retaining $N$ modes) as

$$\frac{\partial}{\partial t} u_N(x,t) + \frac{\partial}{\partial x} \left[ P_N \left( \frac{u^2(x,t)}{2} \right) \right] = \epsilon \frac{\partial}{\partial x} \left[ Q_N \frac{\partial u_N}{\partial x} \right]. \quad (6.4)$$

where the star (*) denotes convolution and $P_N$ is a projection operator. $Q_N$ is a (possibly nonlinear) viscosity kernel, which is only activated for high wave numbers. In Fourier space, this kind of spectral viscosity can be efficiently implemented as multiplication of the Fourier coefficients of $u_N$ with the Fourier coefficients of the kernel $Q_N$, i.e.,

$$\epsilon \frac{\partial}{\partial x} \left[ Q_N \frac{\partial u_N}{\partial x} \right] = -\epsilon \sum_{M_{SVV} \leq k \leq N} k^2 \hat{Q}_k(t) \hat{u_k}(t) e^{ikx}.$$

where $k$ is the wave number, $N$ the number of Fourier modes, and $M_{SVV}$ the wavenumber above which the spectral vanishing viscosity is activated.

Originally, Tadmor [69] used

$$\hat{Q}_k = \begin{cases} 0, & |k| \leq M_{SVV}, \\ 1, & |k| > M_{SVV}. \end{cases} \quad (6.5)$$

with $\epsilon M_{SVV} \sim 0.25$ based on the consideration of minimizing the total-variation of the numerical solution. In subsequent work, however, a smooth kernel was used, since it was found that the $C^\infty$ smoothness of $\hat{Q}_k$ improves the resolution of the SVV method. For Legendre pseudo-spectral methods, Maday, Kaber & Tadmor [54] used $\epsilon \approx N^{-1}$. activated
for modes \( k > M_{SVV} \approx 5\sqrt{N} \), with

\[
\hat{Q}_k = e^{-\frac{(k-N)^2}{(k-M_{SVV})^2}}, \quad k > M_{SVV}.
\]  

(6.6)

Karamanos et al. [43] made the first extension of the spectral vanishing viscosity concept to spectral/hp element methods. In [43], the general form of the SVV operation as presented by Tadmor is maintained; however, polynomial filtering is used to mimic the convolution operator in Tadmor’s formulation. In this chapter, we will present the following extensions to the concept of spectral vanishing viscosity for spectral/hp elements:

1. A new modification to the SVV operator for the continuous Galerkin method which filters on an orthogonal basis.

2. The extension of SVV to the discontinuous Galerkin method.

3. Introduce the concept of dynamic SVV as an alternative LES formulation.

We will present the basic mathematical concepts for each of the points above, and for each point provide computational examples by solving the inviscid Burgers equation. We will then conclude by demonstrating the use of SVV for the simulation of both incompressible and compressible flows.

### 6.1 SVV for the Continuous Galerkin Method

In [43] the SVV concept was implemented in the context of a modal spectral/hp discretization in which a formulation was used for the simulation of incompressible flows [45]. The SVV filtering was accomplished within the context of the \( C^0 \) basis. Although the basis in hierarchical, the continuity involved in the Galerkin projection destroys partially the orthogonality of the basis. A new modification compared to previous work is now the filtering operation is applied to an orthogonal basis which results from a “rotation” of the semi-orthogonal basis. We first present the new SVV formulation for the continuous Galerkin method, and then we demonstrate its use in solving the inviscid Burgers equation.
6.1.1 New SVV Formulation for CGM

Burgers equation written in a strong form with the SVV term added to the right hand side is given by:

\[ \frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial u^2}{\partial x} = \epsilon \frac{\partial}{\partial x} (Q \frac{\partial u}{\partial x}) \]  

(6.7)

If we examine the weak form of the SVV term only, and ignore boundary terms and the leading coefficient we have the following basic form of the SVV operator:

\[ \left( \frac{\partial v}{\partial x}, Q \frac{\partial u}{\partial x} \right) \]  

(6.8)

where \( v \) is a test function taken from \( \{\phi_k\} \), and \( u = \sum_k u_k \phi_k \). In the derivation below, we will assume that all discrete summations are from \( 1, \ldots, N \). In the notation above and henceforth, \( (\cdot, \cdot) \) denotes the \( L_2 \) inner product, and it is assumed that the continuous and discrete inner products are interchangeable given sufficient quadrature order.

Let \( B \) be a matrix which transforms the modal coefficients \( \hat{u} \) for the basis functions \( \{\phi_k\} \) to \( \hat{u} \) in \( \psi_k \) space where \( \{\phi_k\} \) is our \( C^0 \) basis used for the continuous Galerkin formulation and where \( \{\psi_k\} \) is an orthonormal basis which spans the same space as \( \{\phi_k\} \). Let \( F \) be a diagonal matrix which acts as a filtering function (the diagonal entries of which are given by equation (6.6)).

In the notation above, we have that \( \hat{u} = Bu \). Our goal is to filter the coefficients \( \hat{u} \) instead of filtering the coefficients \( \hat{u} \) as done in [43]. Hence we want to transform (by the rotation matrix \( B \)) to the orthogonal space, filter, and then transform back. This operator is accomplished as follows:

\[ \hat{u} = B^{-1}FB\hat{u}. \]  

(6.9)

We can rewrite this as expression as \( \hat{u} = \Theta\hat{u} \) where \( \Theta = B^{-1}FB \). We can now write equation (6.8) in the discrete form using matrix notation as follows:

\[ S^T B^{-1} F B M^{-1} \hat{u} \]
where \( S_{ij} = (\phi_i, \frac{\partial \phi_j}{\partial x}) \) and \( M_{ij} = (\phi_i, \phi_j) \). In the above equation, it can be shown that \( B^{-1} = M^{-1} B^T \), and hence we have that the discrete form of the SVV operator for the continuous Galerkin method is given by the following expression:

\[
S^T M^{-1} B^T F B M^{-1} S \hat{u}.
\]

**Proof of Symmetry of the New Operator**

We now must verify that the newly derived discrete operator is a symmetric operator.

**Claim:** \( S^T M^{-1} B^T F B M^{-1} S \) is a symmetric matrix.

**Proof:** Observe that

\[
(S^T M^{-1} B^T F B M^{-1} S)^T = S^T M^{-T} B^T F^T B M^{-1} S \tag{6.10}
\]

\[
= S^T M^{-1} B^T F B M^{-1} S \tag{6.11}
\]

using the fact that \( F \) is diagonal and hence symmetric, and using the fact that \( M \) is symmetric and hence \( M^{-T} = M^{-1} \).

**Proof of Semi-Positive Definiteness of the New Operator**

To show that a matrix is positive semi-definite, we must show that

\( \hat{u}^T S^T M^{-1} B^T F B M^{-1} S \hat{u} \geq 0 \) for all non-zero real vectors \( \hat{u} \).

Observe that the above expression can be rewritten as:

\[
(B M^{-1} S \hat{u})^T F (B M^{-1} S \hat{u})
\]

Let \( \xi = B M^{-1} S \hat{u} \), yielding the expression \( \xi^T F \xi \).

Because \( F \) is a diagonal matrix in which all the diagonal entries are greater than or equal to zero (this fact is due to the choice of the filtering function given by equation (6.6)), we know that it is true that \( x^T F x \geq 0 \) for all \( x \). Hence, in particular, \( \xi^T F \xi \geq 0 \) for all \( \xi \) of the form \( \xi = B M^{-1} S \hat{u} \). Therefore we have that \( S^T M^{-1} B^T F B M^{-1} S \) is positive semi-definite.
6.1.2 Numerical Example Demonstrating CGM-SVV

To test our implementation of SVV, we first returned to the continuous Galerkin implementation of the inviscid Burgers equation. In figure 6.1 we plot the solution of the inviscid Burgers equation at time $T = 0.5$ without and with SVV. Five equally spaced element spanning $[-1, 1]$ were used, each element containing 16 modes. In this example, a wave cutoff of $M_{SVV} = 8$ and amplitude of $\epsilon = 1/16$ were used.

![Figure 6.1: Solution of inviscid Burgers equation at time $T = 0.5$ using continuous Galerkin without (top) and with (bottom) SVV. Five equally spaced elements spanning $[-1, 1]$ were used, each of which contained 16 modes.](image)

Observe that the SVV has two positive effects: in the region of the discontinuity (the discontinuity is centered at the origin), SVV greatly minimizes the variation of the solution; away from the discontinuity, the "wiggles" in the solution have been removed. This feature of SVV is consistent with the results shown for Fourier methods in [69] and for continuous Galerkin spectral element methods in [43]. The SVV result away from the discontinuity looks amazingly smooth. This can be attributed to the fact that for the one dimensional CG method, only the linear modes directly transmit information between elements (all higher order modes are bubble modes interior to the element and zero on the elemental boundary). Hence information is only propagated through the linear modes to elements away from the sharp gradient, and SVV immediately dampens the energy introduced to the higher modes through the non-linear terms.
6.2 SVV for the Discontinuous Galerkin Method

We extended the idea of SVV to the discontinuous Galerkin formulation. One natural advantage of the discontinuous Galerkin formulation is that the basis functions used for the modal expansion are orthonormal, and hence no additional transformation prior to filtering is necessary (unlike the continuous Galerkin case presented earlier).

6.2.1 SVV Formulation for DGM

Initially, following the work of Tadmor [69], we formulated SVV for the discontinuous Galerkin formulation as the addition of the term (in one-dimension):

$$SVV_1 = \epsilon \frac{\partial}{\partial x} \left( Q \frac{\partial u}{\partial x} \right)$$

where $Q$ is once again a polynomial filtering operation and where all derivatives in the above expression are taken to be the numerical DG derivatives (in their appropriately defined weak forms). In [35], however, a slightly modified form of the SVV operator was proposed. We adapted the form of the operator presented in [35] to yield the following form for the discontinuous Galerkin SVV operator:

$$SVV_2 = \epsilon Q \frac{\partial^2}{\partial x^2} Q u$$

where $Q$ is a polynomial filter, and the second derivative operations are given by the appropriate numerical DG derivatives.

As seen above, we will designate these two different implementations of the SVV operator for the discontinuous Galerkin formulation as $SVV_1$ and $SVV_2$. We will now present some remarks concerning these two operators:

- Observe that the new $SVV_2$ form exercises two uses of the polynomial filter instead of just one as in $SVV_1$, and that instead of filtering only the intermediary stage first derivative, the new SVV form filters both the initial solution and the resulting second derivative. The complete ramifications of this are not understood at this time; however, when the SVV forms are discretized using five equally spaced elements and
65

16 modes per element, and SVV is used with wave-cutoff $M_{SVV} = 8$ and amplitude $\epsilon = 16$. The eigenspectrum of the two operators as given in figure 6.2 shows that the second SVV form is less dissipative than the first form.

![Figure 6.2: Comparison of eigenspectrum of the two operators $SVV_1$ and $SVV_2$. Details of this case are presented in the text.](image)

The ramifications of this will be discussed further when numerical examples are provided later.

- It should be noted that the second SVV form is not written in conservative form, whereas the first form is. No immediate consequence of this has been seen in the numerical studies performed thus far.

- The operation count of the two operators is not identical, and should be taken into account when determining which SVV form would be used.

Because no conclusion has been drawn as to which form of the operator is preferable, we will use both operators in the studies below, and attempt to highlight the differences found.

### 6.2.2 Numerical Example Demonstrating DGM-SVV

To test our implementation of SVV, we first returned to the discontinuous Galerkin implementation of the inviscid Burgers equation. In figure 6.3, we present the solution of
the inviscid Burgers equation at time $T = 0.5$ using the discontinuous Galerkin method both with and without SVV. The $SVV_1$ form was used in this numerical example. Five equally spaced elements spanning $[-1, 1]$ were used, each element containing 16 modes. Twenty-four quadrature points were used for the numerical integration of both solutions. For the SVV solution, a mode number cut-off of $M_{SVV} = 8$ was used, with $\epsilon = \frac{1}{M_{SVV}} = \frac{1}{8}$.

![Graph](image)

Figure 6.3: Solution of inviscid Burgers equation at time $T = 0.5$ using the discontinuous Galerkin method without (top) and with (bottom) SVV.

Due to the addition of viscosity (in the form of SVV) to the inviscid solution, variation in the solution can be observed in elements away from the discontinuity, unlike the non-SVV case. Although there is a slight increase in the $L_2$ error of the SVV solution versus the non-SVV solution (0.162305 versus 0.133395), there is a marked difference in the discrete $L_\infty$ error (0.652071 versus 1.17006 for SVV versus non-SVV respectively).

We now return to test presented in section 5.2.2 in which we examined the energy in the modes (denoted by the square of the modal values) within the element that contains the jump in the inviscid Burgers solution (the middle element). The modes were extracted at time $T = 0.35$. after the shock has formed (at time $\frac{1}{7}$) and prior to the solution becoming unstable. In figure 6.4 we plot the square of the modal coefficients versus the mode number. As noted earlier, we observe that for the case $Q = M + 1$, no-SVV the energy increases at the highest modal number, yet in the case where over integration is used for the non-linear terms, the square of the modes at the highest mode numbers is greatly reduced. We now draw your attention to the two SVV cases, one in which only

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\( Q = M + 1 \) quadrature points were used, and the other in which \( Q = \lceil \frac{3}{2} M \rceil \) points were used. When SVV is used with the under-integrated solution, the energy in the highest modes remains at a level comparable to the non-SVV, properly integrated case. For long-time integration, the SVV solution remains stable since it draws energy out of the high modes (energy which in the under-integration case is due to both under-resolution and aliasing error). When SVV is added to the properly integrated case, it merely reduces further the small amount of energy contained in the highest mode numbers. Observe also that in the properly integrated case with SVV, the low mode numbers are not influenced.

![Figure 6.4: Modal coefficients in the middle element of the inviscid Burgers solution at time \( T = 0.35 \). Both over-integration and SVV lead to a stable solution unlike the collocation approach.](image)

In figure 6.5 we present a comparison of the square of the modal coefficients in the middle element at \( T = 0.35 \) for \( SVV_1 \) and \( SVV_2 \). In both cases \( Q = M + 1 = 17 \) quadrature points were used. Observe that the trend is similar for both types of SVV. Upon examining \( SVV_2 \), we see that the square of the modal coefficients for high mode numbers do not die off as fast as for \( SVV_1 \). This fact is consistent with the observation made above in section 6.2.1 that \( SVV_1 \) is more dissipative than \( SVV_2 \).
6.3 Dynamic SVV

In the dynamic approach the viscosity amplitude in the SVV kernel varies as a function of space and time. We first apply this idea to the inviscid Burgers equation, and subsequently we implement it in the context of Navier-Stokes equations.

6.3.1 Burgers Equation

\[
\frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial u^2}{\partial x} = c(x,t)SVV(Q,u) \tag{6.12}
\]

where

\[
c(x,t) = \epsilon = \frac{\kappa |u_x(x,t)|}{M \|u_x(x,t)\|_{\infty}} \tag{6.13}
\]

Here \( \kappa \) is a scalar that can be determined by optimizing the quality of the solution as we shall see below. In other words, we have incorporated the solution into determining the magnitude of the viscosity through the coefficient \( c(x,t) \). In this one-dimensional case, we have employed a normalized gradient to accomplish this. This form of \( c(x,t) \) is meant to be analogous to the adaptive coefficient for the Navier-Stokes equations \( C(x,t) = \nu_e(x,t)/\nu \) (where \( \nu \) is the physical viscosity and \( \nu_e \) is the eddy viscosity) proposed in [43]. The use of the rate of strain tensor in the computation of \( \nu_e \) is mimicked by the use of the magnitude of the first derivative for this one-dimensional example.
We first examine how the dynamic coefficient affects the quality of the solution. First, we evaluate the dynamic coefficient $c(x, t)$ a posteriori from the numerical solution of inviscid Burgers without any viscosity treatment (figure 6.6: case A). We then compute using the global, i.e. max norm obtained across all elements, involving the inviscid Burgers equation with dynamic SVV applied to it (figure 6.6: case B). Finally, we compute using local i.e. max norm obtained in each element, involving the inviscid Burgers equation with dynamic SVV applied to it (figure 6.6: case C). We observe from figure 6.6 that if the global $\|u_t(x, t)\|_\infty$ is used in the definition of $c(x, t)$, the form of the $c(x, t)$ when dynamic SVV is acting on the system is the same in shape as when $c(x, t)$ is obtained a posteriori. When a local definition of $c(x, t)$ is used, however, the form of $c(x, t)$ changes greatly. From this we conclude that

- To effectively utilize dynamic SVV, a global scaling quantity such as taken over the entire domain must be used.

![Figure 6.6: Plot of the dynamic coefficient $c(x, t)$ at the final time $T = 0.5$. The three cases are explained in the text.](image)

To understand the effect of the scaling parameter $\kappa/M$ (where $M$ is the number of modes on an element), we performed a comparison of static SVV ($c(x, t) = \kappa/M$) versus dynamic SVV ($c(x, t)$ as given in equation (6.13)). Inviscid Burgers equation was solved with the added SVV term. Five equally spaced elements spanning $[-1, 1]$ were used, each
element containing 16 modes. Comparisons of the $L_2$ error versus $\kappa/M$ and the $L_\infty$ error versus $\kappa/M$ are presented for $SV'V_1$ in table 6.1 and for $SV'V_2$ in table 6.2.

<table>
<thead>
<tr>
<th>Form</th>
<th>$L_2$ Error</th>
<th>$L_\infty$ Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inviscid</td>
<td>0.133395</td>
<td>1.17006</td>
</tr>
<tr>
<td>Global $SV'V_1$, $\epsilon = \frac{1}{16}$</td>
<td>0.162305</td>
<td>0.652071</td>
</tr>
<tr>
<td>Dynamic $SV'V_1$, $\kappa = 1$, local $|u_f(x,t)|_\infty$</td>
<td>1.5039</td>
<td>4.7258</td>
</tr>
<tr>
<td>Dynamic $SV'V_1$, $\kappa = 1$, global $|u_f(x,t)|_\infty$</td>
<td>0.131261</td>
<td>1.05779</td>
</tr>
<tr>
<td>Dynamic $SV'V_1$, $\kappa = 2$, global $|u_f(x,t)|_\infty$</td>
<td>0.139909</td>
<td>0.579867</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of $L_2$ and $L_\infty$ errors for the inviscid Burgers equation using dynamic $SVV$ ($SV'V_1$) with $M_{SVV} = 8$ and $\epsilon = 1/16$.

<table>
<thead>
<tr>
<th>Form</th>
<th>$L_2$ Error</th>
<th>$L_\infty$ Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inviscid</td>
<td>0.133395</td>
<td>1.17006</td>
</tr>
<tr>
<td>Global $SV'V_2$, $\epsilon = \frac{1}{16}$</td>
<td>0.131833</td>
<td>0.44753</td>
</tr>
<tr>
<td>Dynamic $SV'V_2$, $\kappa = 1$, local $|u_f(x,t)|_\infty$</td>
<td>0.155219</td>
<td>0.699105</td>
</tr>
<tr>
<td>Dynamic $SV'V_2$, $\kappa = 1$, global $|u_f(x,t)|_\infty$</td>
<td>0.13296</td>
<td>1.15936</td>
</tr>
<tr>
<td>Dynamic $SV'V_2$, $\kappa = 5$, global $|u_f(x,t)|_\infty$</td>
<td>0.131130</td>
<td>1.04961</td>
</tr>
<tr>
<td>Dynamic $SV'V_2$, $\kappa = 10$, global $|u_f(x,t)|_\infty$</td>
<td>0.140145</td>
<td>0.775494</td>
</tr>
</tbody>
</table>

Table 6.2: Comparison of $L_2$ and $L_\infty$ errors for the inviscid Burgers equation using dynamic $SVV$ ($SV'V_2$) with $M_{SVV} = 8$ and $\epsilon = 1/16$.

Several observations can be made based on our studies:

- At the default value of $\frac{1}{37}$, the dynamic $SVV$ does slightly better than the static $SVV$ in the $L_2$; however, the dynamic $SVV$ does noticeably better in the $L_\infty$ than the static $SVV$ for the same value.

- For all values of $\kappa/M$ less than $\frac{1}{37}$, the static and dynamic $SVV$ perform identically.

- For both static and dynamic $SVV$, there exists an optimal value of $\kappa/M$ which is less than $\frac{1}{37}$. As viscosity in the form of $SVV$ is added to the system, the numerical solution moves toward becoming monotonic; however, at some point, too much viscosity is added and a degradation in the solution occurs as monotonicity is traded.
for accuracy.

### 6.3.2 Alternative LES Implementation

We apply this idea of dynamic SVV to compressible Navier-Stokes equations. Specifically, the SVV kernel is applied to the density, momentum, and energy equations with a variable viscosity amplitude given by

\[
c(x, t) = \frac{\rho |S|}{\|\rho|S|\|_{\infty}}
\]

where \( |S| \) is defined as \( \sqrt{\text{Tr}(S_{ij}S_{ij})} \), \( S_{ij} = \frac{1}{2} \left( \frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) \) and \( \rho \) is the local fluid density. As not to affect the flow at the wall we incorporate the Panton function [60] given by

\[
g(y^+) = \frac{2}{\pi} \tan^{-1} \left( \frac{2ky^+}{\pi} \right) \left[ 1 - \exp \left( -\frac{y^+}{C^+} \right) \right]^2
\]

where all quantities are expressed in viscous wall units denoted by \( + \). This function is multiplied pointwise by the coefficient \( c(x, t) \).

![Image](image.png)

Figure 6.7: Segment of the unstructured mesh for flow past an airfoil at 10 degrees angle of attack and \( Re = 10,000 \).

In figures 6.7 and 6.8 we plot, respectively, a segment of the mesh around an airfoil and contours of the SVV amplitude computed using the aforementioned procedure. The
Figure 6.8: Amplitude at one time instance of spectral vanishing viscosity in flow past an airfoil at 10 degrees angle of attack and $Re = 10,000$.

spectral/hp element simulation is for a two-dimensional flow past an airfoil at 10 degrees angle of attack. The mesh involved 912 quadrilateral elements with sixth-order polynomial interpolation. It is clear that the SVV is non-zero in regions of high vorticity which are the most probable candidates for under-resolution in subsonic flows.

6.4 Use of SVV in Flow Simulations

6.4.1 SVV-LES Coarse Resolution Simulations

The effectiveness of SVV in simulations of turbulent flows using low resolution has been demonstrated in [43]. Here, we revisit this problem using the aforementioned modification of the continuous Galerkin SVV operator to study the effect of the viscosity amplitude and the wave number cut-off on the solution quality. Specifically, we apply the new formulation for the continuous Galerkin method presented in section 6.1.1 which filters on an orthogonal trial basis instead of the semi-orthogonal basis employed previously in [43].

Channel flow at $Re_T = 180$ is simulated, with periodic boundary conditions in the streamwise and spanwise directions following the benchmark solutions of Kim, Moin & Moser [46]. The mesh used here is the same as in [43], but the resolution was doubled in the streamwise direction and was substantially reduced in the crossflow plane. Specifically, the size of computational domain was $L_x = 5$, $L_y = 2$, and $L_z = 2$. In contrast to the previous simulations in [43], we increased the streamwise resolution to 32 Fourier modes (64 points) to test more carefully the effect of SVV acting only in the crossflow planes and not in the streamwise direction. In the Fourier direction a $3/2$ de-aliasing rule was applied for all simulations. The spectral element mesh has 25 elements (see figure 6.9) in the crossflow plane, with a polynomial order of $N = 7$ compared to previous simulations.
in [43] where \( N = 21 \). We note that the resolution in the wall-normal direction in the current simulations involves only 35 points!

![Figure 6.9: Mesh in the crossflow plane for turbulent channel flow at \( Re_z = 180 \)](image)

In figure 6.10 we plot the mean-velocity profiles versus the distance from the wall, and in figure 6.11 we plot the turbulence intensities versus the distance from the wall for four different cases. The symbols denote the DNS of Kim, Moin & Moser [46]. First, we obtained converged (in-time) statistics without SVV using the aforementioned low-resolution. The corresponding results under-estimate the mean-velocity at the centerline \( (U_r/\bar{u}_r = 17.67 \text{ versus } U_c/\bar{u}_r = 18.2 \text{ in [46]}) \) as shown in figure 6.10 (solid line). Examining the turbulent intensities, the corresponding results under-estimate the streamwise velocity component and over-estimate the crossflow, as shown in figure 6.11 (solid line). In our initial runs, we tried two modifications. First, we applied polynomial over-integration which effectively removes any aliasing in the crossflow direction, but the results remained effectively the same. Secondly, we applied the new SVV operator with the default parameters \( (\tilde{M}_{SVV} = 5, \epsilon = 1/8) \), which also gave results similar to the untreated case.

We then experimented with several combinations of the SVV parameters \( (\tilde{M}_{SVV}, \epsilon) \). The SVV kernel is scaled with the given physical viscosity, so the actual term included in the Navier-Stokes equation is proportional to \( Re^{-1}\epsilon \).

The best results for the turbulent intensities are shown in figure 6.11 (dash-dot line)
correspond to \( M_{SVV} = 5 \) and \( \epsilon = 5/8 \). This set of parameters yields a mean-velocity at the centerline \( U_c/u_r = 17.9 \) as shown in figure 6.10 (dash-dot line). The other curves in the figures 6.10 and 6.11 correspond to \( (M_{SVV} = 2, \epsilon = 1/8) \) (dot) and \( (M_{SVV} = 5, \epsilon = 9/8) \) (dashed). The \( (M_{SVV} = 2, \epsilon = 1/8) \) case shows improvement in both the turbulent intensities and mean-velocity profile compared to the untreated case (solid-line); however, dissipation is being added over a larger number of modes compared to the \( (M_{SVV} = 5, \epsilon = 5/8) \) case. Observe that for \( (M_{SVV} = 5, \epsilon = 9/8) \) too much dissipation has been added to the system, and hence the solution over-estimates the streamwise velocity component as shown in 6.11. This case, however, yields the best mean-velocity profile with \( U_c/u_r = 18.15 \) at the centerline. Although case-specific, these results confirm the theoretical results that only the upper one-third of the spectrum should be treated with SVV, and that there is an optimum (but unknown) viscosity amplitude level. We propose computing dynamically the viscosity amplitude level using the methodology outlined earlier. Future work will consist of comparisons of static and dynamic SVV for high Reynolds number flow.

![Figure 6.10: Mean-velocity profile for the turbulent channel flow. The symbols correspond to the benchmark solutions of Kim, Moin & Moser [46]. The solid-line corresponds to the under-resolved DNS, the dotted-line to \( (M_{SVV} = 2, \epsilon = 1/8) \), the dot-dashed-line to \( (M_{SVV} = 5, \epsilon = 5/8) \), and the dashed-line to \( (M_{SVV} = 5, \epsilon = 9/8) \).]
6.4.2 2D Flow Around a Pitching Airfoil

To demonstrate the effect of SVV when using the discontinuous Galerkin method, we revisit the laminar compressible flow around a rapidly pitching NACA 0015 airfoil. For this simulation, the chord Reynolds number is $Re = 45,000$, and freestream Mach number is $Ma = 0.2$. The details of this computation were presented previously in section 3.3.1; we present here only those components of the simulation necessary to highlight the effects of SVV. In the previously discussed computation, over-integration as presented in 5 was employed. Without over-integration, this computation becomes unstable. The goal of this experiment is to see if SVV can replace over-integration as a means of stabilizing the computation, and if the computational results obtained using SVV satisfactorily replicate the over-integrated results.

The computational mesh used for this simulation consisted of 912 quadrilateral elements: the full domain and local tessellation are shown in figure 6.12.

We examined both the $SVV_1$ and the $SVV_2$ implementations starting from the same initial condition. Although from our previous analysis $SVV_2$ is less dissipative than $SVV_1$, $SVV_2$ remained stable for longer (as a function of pitching angle) than when $SVV_1$ was applied with identical wave cut-off and amplitude parameters. A multitude of experiments

Figure 6.11: Turbulence intensities for the turbulent channel flow. The symbols correspond to the benchmark solutions of Kim, Moin & Moser [46]. The solid-line corresponds to the under-resolved DNS, the dotted-line to ($M_{SVV} = 2, \epsilon = 1/8$), the dot-dashed-line to ($M_{SVV} = 5, \epsilon = 5/8$), and the dashed-line to ($M_{SVV} = 5, \epsilon = 9/8$).
Figure 6.12: Full domain (left) and local tessellation (right) for the simulation around the NACA 0015 pitching airfoil. All dimensions are in units of chord length.

were tried with variations on the amplitude of the SVV used; in all cases the wave cut-off was set to $M_{SVV} = 4$.

In figure 6.13 we present the coefficient of lift $C_L$ for the $SVV_2$ case with the most SVV added ($\epsilon = 20/9$). We present a comparison to the over-integrated result presented in 3.3.1 and to the experimental and computational results presented in [71].

Figure 6.13: Comparison of the lift coefficient $C_L$ versus angle of attack in degrees for SVV (solid), over-integrated (dashed), and the computational (circle) and experimental (asterisk) results presented in [71].

Even with $\epsilon = 20/9$, the SVV solution eventually goes unstable at an angle of attack
equal to $\alpha = 21.7816$. Numerical experiments show that increasing the amplitude further forced a decrease in the time step, as the SVV term diffusion number limit was not satisfied by the time step restriction set by the advection and diffusion terms of the original Navier-Stokes system. One possible area of study which may illuminate what is happening in this example is to investigate the energy in the modes of those elements that are causing the blow-up, and to isolate what rate of SVV would be necessary to balance the growth of energy in those elements. Another possible study is to examine how much of the spectrum should be affected by SVV (using the $M_{SVV}$ parameter).

6.4.3 Dynamic SVV Used for 2D Flow Around a NACA 0012 Airfoil

In this section we apply dynamic SVV to the two-dimensional unsteady flow past a NACA 0012 airfoil at 10 degrees angle of attack. The Reynolds number is $Re = 10,000$ based on the chord length and the incoming Mach number is $Ma = 0.2$. An unstructured mesh was employed: a segment around the airfoil is shown in figure 6.7. A fourth-order (Jacobi) polynomial interpolation was used corresponding to ten independent modes in each triangle.

Without any treatment the discontinuous Galerkin method for this resolution is unstable at this Reynolds number although it is stable at $Re = 1,000$. However, employing over-integration leads to a stable steady periodic state. We have also run this case with SVV following both a static and a dynamic approach. In this case we chose $\epsilon = 1/5$ and $M_{SVV} = 1$, i.e. all modes but the mean are treated with SVV. We also note that, as in the incompressible case, here too the SVV kernel is scaled with the given viscosity, so the actual viscosity amplitude is $Re^{-1}\epsilon$.

Figure 6.14: Contours of instantaneous streamwise momentum for dynamic SVV simulation at time $T = 131.5$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Both the over-integrated and SVV simulations are stable. A stationary periodic state is reached asymptotically but there are differences in the flow structure as shown in figures 6.14 and 6.15. These flow differences result in quantitative differences in the lift and drag coefficients as shown in figures 6.16 and 6.17. These differences may be attributed to dissipation being added over a large number of the modes (as specified by the SVV parameter $M_{SVV}$).

Figure 6.15: Contours of instantaneous streamwise momentum for simulation with over-integration at time $T = 131.5$.

Figure 6.16: Time history of lift coefficient ($C_L$): The solid line corresponds to over-integration and the dotted line to dynamic SVV.
Figure 6.17: Time history of drag coefficient ($C_D$): The solid line corresponds to over-integration and the dotted line to dynamic SVV.
Chapter 7

Non-Conforming Discretizations

One of the primary advantages of the spectral/hp element method is the ability to combine refinement of the tessellation (h-refinement) with variable polynomial order on each element (p-refinement) for solving problem in complex geometries [45]. In many cases, localized refinement is necessary to effectively resolve features with rapid variation. Although it is possible to refine locally with h-refinement using a conforming mesh, the conformity constraint often leads to a general propagation of the refinement away from the area of interest, and hence additional computation that is not necessary to resolve the features of interest in the localized area. A solution to this problem is the concept of non-conforming discretizations, which are discretizations in which the edge or face conformity constraint (in two-dimensions and three-dimensions, respectively) is relaxed to allow hanging nodes. By relaxing the conformity constraint, non-conforming discretizations allow for highly surgical h-refinement and p-refinement, and in theory provide a discretization paradigm in which the minimum degrees of freedom necessary to solve a given problem can be used.

Non-conforming discretizations have been used successfully in the continuous Galerkin hp-FEM formulation by introducing the concept of mortar elements [57]. The concept of a mortar element is to create a “mortaring” interface between a conforming and non-conforming interface in which the solution is mortared together under certain constraints. Both the mathematical constraints on the mortaring element and implementation details have been well studied for the continuous Galerkin hp-FEM method and the mortaring
concept has been successfully used in the solution of the incompressible Navier-Stokes equations (see [30], [55], [1], and [12]).

Following the continuous Galerkin mortar element idea, Kopriva in [47] and [48] extended the idea to the staggered-grid scheme used for the solution of the Euler and Navier-Stokes equations. The staggered-grid scheme of Kopriva does not require $C^0$ continuity like the continuous Galerkin formulation, and hence different mortaring constraints had to be devised. Kopriva formulated two mortar constraints which he called a conservation constraint and an upwind constraint, both of which can be met on a mortaring interface by choosing the higher functional space between two interfaces. For a ‘p’ non-conforming interface, in which two elements share a geometrically conforming interface but have different polynomial orders, this mortaring constraint requires that the higher polynomial order be used on the mortaring interface. In the case of a geometrically non-conforming interface, this mortaring constraint requires that the mortar be partitioned in the same way as the non-conforming side of the interface, and that the polynomial order on each mortar segment be the highest polynomial order between the edges. In Kopriva’s work, $L^2$ projections were used between element and mortar interfaces.

In [39], Houston et al. demonstrated ‘p’ and ‘h’ non-conforming interfaces using the discontinuous Galerkin method. In their work, similar constraints to those of Kopriva were formulated for the discontinuous Galerkin method. For non-conforming DG, the constraints of Kopriva reduce to guaranteeing that the surface integrals required for the computation of the flux terms remain exact. Insufficient quadrature order, i.e. insufficient number of points in the mortar, leads to losses of both convergence and conservation. Though the mathematical proofs in [39] are focused on the DG methodology, the general principle is the same: to maintain p-convergence it is sufficient to chose the higher functional space between two interfaces as the mortar on which to do the flux term surface integration.

Non-conforming discretizations allow us to accomplish two things: place increasingly higher p-resolution in a region in which the regularity of the solution is high, or accomplish h-refinement with a low order discretization where the solution regularity is low. In this chapter, we will first briefly discuss the first of these two concepts, following the work in
[39] for the computation of the mortars. We will then examine the second of these two concepts, presenting a geometrically non-conforming discretization scheme in which a high-order element is subdivided into a collection of finite volumes in which the element interface between the collection of finite volumes and neighboring elements is non-conforming. The utility of this type of non-conforming discretization is that in regions where the regularity of the solution is low (such as in a shock region in a supersonic flow), a large number (h-refinement) of finite volumes can be used to capture the solution, while away from the low regularity region higher order elements can be used to capture the solution features. We will discuss the geometric splitting scheme and projection schemes used to accomplish this type of non-conforming discretization. We will then present both two-dimensional and three-dimensional supersonic compressible flow calculations in which non-conforming finite volume refinement was used in the shock region while higher order elements were used away from the shock.

7.1 Non-Conforming Discretizations Which Maintain p-Convergence

In this section we will demonstrate, following the mathematical results of [39], how both ‘h’ (tessellation) and ‘p’ (polynomial order) refinement can be used to accurately solve a problem while minimizing the computation work (i.e., the degrees of freedom needed). We begin by explaining how non-conforming interfaces are handled, and then we present an example problem from [57].

In figure 7.1 we provide a diagram describing how three domains $\Omega_1, \Omega_2$ and $\Omega_3$ are interfaced. On the edge containing the hanging node, $\Omega_1, \Omega_2$ and $\Omega_3$ have $q_1, q_2$ and $q_3$ quadrature points respectively. Based upon the arguments of [47] and [39], in order to maintain p-convergence we must create a mortaring interface consisting of two parts as shown in figure 7.1. The number of quadrature points $\hat{q}_1$ used in the upper mortaring interface must be taken to equal the maximum of $q_1$ and $q_2$, and the number of quadrature points $\hat{q}_2$ used in the lower mortaring interface must be taken to equal the maximum of $q_1$ and $q_3$. As shown in [47], projections, not interpolations, must be used for transforming
from one set of points to another. This projection process is inherently built into the
discontinuous Galerkin formulation as presented in [39].

![Diagram showing how non-conforming discretizations are "mortared" together. Notation used in the diagram is explained in the text.](image)

To demonstrate the effectiveness of both 'h' non-conforming and 'p' non-conforming discretizations, we examine the test problem presented in [57] of solving:

\[-\Delta u(x, y) + \lambda^2 u(x, y) = f(x, y) \quad \Omega = [0, 1] \times [0, 1]\]

with exact solution \( u(x, y) = e^{\lambda(x-1)+(y-1)/\sqrt{2}} \) and \( \lambda = 50 \). This particular solution has nice regularity throughout the computational domain. With this particular choice of \( \lambda \) there exists a steep gradient in the upper right hand corner of the solution domain.

Three different discretizations were used as presented figure 7.2. The first discretization, case A. (figure 7.2 (left)) consists of 16 spatially conforming elements. The second discretization, case B. (figure 7.2 (center)) consists of 10 spatially non-conforming elements. The third discretization, case C. (figure 7.2 (right)) consists of four spatially conforming elements.

In figure 7.3 we present a convergence plot for cases A and B. From the figure we see that the convergence rate for the two discretizations is identical even though the non-conforming discretization contains far fewer degrees of freedom. For the final case of nine modes per element, the fully conforming case (case A) contains 1296 degrees of freedom compared to the non-conforming case (case B) which only has 810 degrees of freedom. This example demonstrates how a non-conforming discretization can be used to obtain
Figure 7.2: Three different discretizations used for demonstrating the effectiveness of 'h' non-conforming and 'p' non-conforming discretizations. The three cases A-C (left, center, and right respectively) are discussed in the text.

the same error using far fewer degrees of freedom.

Figure 7.3: Error in the broken $H_1$ norm versus the number of modes per element for discretizations 7.2. The symbols from case A and B overlap.

Since the solution has sufficient regularity, we demonstrate the power of 'p' non-conforming discretizations by solving for same problem on figure 7.2 (right) in which only four modes per element are used in all elements except for the element in the upper right hand corner (the one containing the steep gradient), in which 16 modes are used. With this choice of discretization, an error in the broken $H_1$ norm of $1.72 \times 10^{-6}$ can be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
obtained using only 304 degrees of freedom!

This example demonstrates that exponential convergence can be obtained for the non-conforming discretization when the discontinuous Galerkin mortaring rules of [39] are employed. This example also demonstrates succinctly one advantage of the spectral/hp element method - resolution can be placed as needed to resolve the solution at minimal cost (degrees of freedom).

7.2 Non-Conforming Discretizations Using Finite Volume h-Refinement

When the regularity of the solution is low in a particular region, we seek to use non-conforming h-refinement consisting of a collection of finite volumes to allow us to capture the solution region efficiently and robustly. To accomplish this non-conforming finite volume refinement, we must first define how to partition a high-order element into a collection of volumes, and we must define how the collection of finite volumes communicates (through fluxes) with adjoining high-order elements. In this section, we will present a way to partition the standard element types, triangles and quadrilaterals in two-dimensions, and tetrahedra, prisms, pyramids, and hexahedra in three dimensions into a collection of finite volumes. We will define the projection algorithm used, and will comment on the interfacing algorithm used when a collection of finite volumes in a non-conforming manner with a high-order element.

In figures 7.4 and 7.5 we demonstrate diagrammatically how to geometrically split triangles and quadrilaterals, respectively. First, a variable \( M \) is chosen which designates how many partitioned segments per edge should be used in the finite volume partitioning. Uniform partitioning on each edge in the pattern given in figures 7.4 and 7.5 produces \( M^2 \) self-similar elements. For three-dimensional elements, a similar methodology is used yielding \( M^3 \) finite volumes per element partitioning.

As shown in figures 7.4 and 7.5, we must now define a projection from the \( N^{th} \) order polynomial to a collection of finite volumes (for refinement) and a reverse transform from the collection of finite volumes back to a \( N^{th} \) order polynomial (for de-refinement). The
Figure 7.4: Diagram showing the splitting scheme from a triangle $\Omega$ having a $P_N$ order polynomial to a collection of $M^2$ finite volumes $\Omega_j$. $\Pi^1$ and $\Pi^2$ represent the forward and backward transformations respectively for this splitting scheme.

Figure 7.5: Diagram showing the splitting scheme from a quadrilateral $\Omega$ having a $P_N$ order polynomial to a collection of $M^2$ finite volumes $\Omega_j$. $\Pi^1$ and $\Pi^2$ represent the forward and backward transformations respectively for this splitting scheme.
projections $\Pi^1$ and $\Pi^2$ are formulated the same way for all element types. Any distinctions which arise for a particular element type will be discussed explicitly. We now define the two projections $\Pi^1$ and $\Pi^2$.

The projection $\Pi^1$ from a function $F(\vec{x}) \in P_N$ defined on $\Omega$ to a collection of finite volumes $\Omega_j$, $j = 1, \ldots, M^d$ where $d$ is the dimension of the element is defined as follows. The function $F(\vec{x}) \in P_N$ can be written as the expansion $F(\vec{x}) = \sum_{i=1}^{R} \hat{F}_i \phi_i(\vec{x})$ where $\phi_i$ is an orthonormal basis defined on $\Omega$ and where the rules for computing $R$ based upon $N$ and the element type are given in [45]. Let $|\Omega| = \int_\Omega 1 \, d\vec{x}$. The definition of the projection $\Pi^1$ consists of computing on each $\Omega_j$:

$$\hat{F}_j = \frac{1}{|\Omega|} \int_{\Omega_j} F(\vec{x}) \, d\vec{x}$$

where $j = 1, \ldots, M^d$. This transformation from a polynomial of order $N$ on $\Omega$ to a collection of $M^d$ finite volumes is well-defined for all combinations $N \geq 0, M \geq 1$ and is conservative.

The transformation $\Pi^2$ from the collection of $M^d$ finite volumes to a polynomial $G(\vec{x}) \in P_N$ can be defined as follows. From the derivation of the forward transformation given above, we see that the transformation $\Pi^1$ can be written in matrix form as $\vec{F} = A \vec{\hat{F}}$ where $\vec{F} = \hat{F}_j$, $j = 1, \ldots, M^d$, $\vec{\hat{F}} = \hat{F}_i$, $i = 1, \ldots, R$, and where $A$ is a matrix of size $M^d \times R$ formed using equation (7.1). The least-squares pseudo-inverse [66], $\tilde{A}$, of this transformation is given by $\tilde{A} = (A^T A)^{-1} A^T$. This transformation $\tilde{A}$ will be used as the transformation $\Pi^2$.

We will now present some observations concerning the transformations $\Pi^1$ and $\Pi^2$:

- If $M^d = R$ then $A$ is a non-singular square matrix with $A^{-1} = \tilde{A}$. In this case, $\Pi^2$ is the true inverse transformation of $\Pi^1$ and hence both $\Pi^1$ and $\Pi^2$ are conservative. Upon examination of the formulas for computing $R$, one observes that this case can be accomplished for quadrilateral and hexahedral elements, but is quite difficult to be satisfied for triangular, tetrahedral, prismatic, and pyramidal elements where a truncated number of modes is used in the tensor-product expansion [45].

- If $M^d > R$ then $\tilde{A}$ provides a least-squares pseudo-inverse of $A$, and hence $\Pi^2$ is
a pseudo-inverse transformation of $\Pi^1$. For all $M^d > R$, the transformation $\Pi^1$ is well defined and in theory computable. Recently, Wang [72] observed when creating a splitting scheme for what he calls spectral finite volumes that as $M^d$ grows much larger than $R$ the conditioning of the inverse system may become poor. It is speculated that the growth in the condition number of inverse system is due to the uniformly-spaced geometric partitioning used. This type of partitioning is similar to the evenly-spaced partitioning used in least-squares Vandermonde computations, a computation which has been documented to yield a matrix which is poorly conditioned [32]. To alleviate this problem Wang came up with special geometric partitioning which yields a matrix with a reasonable condition number.

- If $M^d < R$ then the system is under-determined, and special care must be taken in computing the transformation $\Pi^2$. The most straight-forward means of computing $\Pi^2$ for this case is to find the largest polynomial order $N^*$ such that $R(N^*) < M^d$ and use the transformation methodology previously described. All other modes between $R(N^*)$ and the original $R$ should be zeroed.

Now that we have defined how to partition an element and how to project the solution from an $N$ order polynomial to a collection of finite volumes and back, we must now comment on how to interface on a non-conforming edge/face (2D/3D respectively) the collection of finite volumes to a high-order element. Following the work of [51], the mortaring interface consists of a collection of Gauss points whose number is specified as the maximum between the number of finite volumes on the interface and the number of modes used on the high-order element interface. To transform between the mortaring interface and the collection of elements, an $L^2$ projection as used in [47] is employed.

### 7.2.1 Numerical Experiments Using Finite Volume $h$-Refinement

To fully understand the splitting and projection scheme presented above, we performed several numerical experiments, some of which we will discuss here. We will now present three groups of numerical experiments concerning the following topics: refinement using the methodology above, de-refinement using the methodology above, and the solution of
the advection equation after having used the refinement scheme above.

We begin by examining the refinement algorithm previously presented. Our model problem will be the function $u(x, y) = \sin(0.5\pi x) \sin(0.5\pi y)$ on $\Omega = [-1, 1] \times [-1, 1]$. The following numerical experiment was performed. We begin by first projecting our function $u(x, y)$ onto four elements containing $7^{th}$ order polynomials per element (we will refer to these elements as our macro-elements). The error associated with this projection is $L_\infty = 9.8 \times 10^{-8}$. We then refine the four macro-elements into a collection of $M^2$ finite volumes per element using the algorithm previously given. In figure 7.6 we present the original discretization (left) and two example finite volume discretizations (center and right).

Figure 7.6: Original four element mesh with $7^{th}$ order polynomials per element (left); original mesh partitioned into eight finite volumes per element (center); original mesh partitioned into 32 finite volumes per element (right).

In figure 7.7 we plot the $L_\infty$ error versus the number of finite volume elements per side ($M$) into which each macro-element was partitioned. Algebraic convergence of order one is observed as would be expected from piece-wise constant interpolation of this form [26].

We have yet to address the question, given an element with $R$ degrees of freedom in its polynomial expansion, into how many finite volumes should the macro-element be partitioned? We propose that there are at least two possible answers to this question, one based upon de-refinement and one based upon the use of finite volumes for wave propagation problems. We will now examine these two possibilities.

One possibility for the choice of the refinement parameter $M$ is to find the $M$ such that when the partitioning scheme is applied (projection to finite volumes) and then de-
Figure 7.7: $L_\infty$ error versus the number of finite volume elements per side ($M$) into which each macro-element was partitioned.

refinement is applied (back to the original $R$ degrees of freedom on the macro-element), we recover the original function. To determine this value of $M$, recall that by the mean value theorem we know that there exists a $\xi \in \Omega_j$ such that

$$F(\xi) = \frac{1}{|\Omega_j|} \int_{\Omega_j} F(x) \, dx.$$  \hfill (7.2)

Hence for each finite volume into which we partition, we are guaranteed that for some point $\xi$ inside that finite volume we have interpolated the original function. Hence from interpolation theory we know that as long as we partition the macro-element into $M^d \geq R$ finite volumes (where $R$ is the number of degrees of freedom in the polynomial expansion on the macro-element), the inverse transformation as previously defined yields back the original function. We now demonstrate this fact for both triangle partitioning and quadrilateral partitioning.

In figure 7.8, we demonstrate this concept for triangles by first projecting our function $u(x, y)$ onto two triangular elements, each containing $7^{th}$ order polynomial expansions. We then partition each of the macro-elements into 49 finite volumes using the algorithm above (this number being chosen so that the number of finite volumes is greater than or equal to the number of degrees of freedom on a macro-element). We then employ our inverse transformation $\Pi^2$ to reconstruct the original solution. The error between the reconstructed solution and the original projected solution is less than machine precision.

In figure 7.9, we demonstrate this concept for quadrilaterals by first projecting our
Figure 7.8: Macro-element discretization consisting of two triangles, each with 7th-order polynomial expansions (left). Finite volume partitioning consisting of 49 finite volumes per macro-element (center). Reconstructed solution using $\Pi^2$ operator (right).

function $u(x, y)$ onto four quadrilateral elements, each containing 7th order polynomial expansions. We then partition each of the macro-elements into 64 finite volumes using the algorithm above (this number being chosen so that the number of finite volumes is greater than or equal to the number of degrees of freedom on a macro-element). We then employ our inverse transformation $\Pi^2$ to reconstruct the original solution. The error between the reconstructed solution and the original projected solution is less than machine precision.

Figure 7.9: Macro-element discretization consisting of four quadrilaterals, each with 7th polynomial expansions (left). Finite volume partitioning consisting of 64 finite volumes per macro-element (center). Reconstructed solution using $\Pi^2$ operator (right).

In figure 7.10, we demonstrate the consequences of choosing $M^2 < R$. We begin by projecting our function $u(x, y)$ onto four quadrilateral elements, each containing 7th order polynomial expansions. When we compare the true solution $u(x, y)$ to the projected solution we obtain an $L_\infty$ error of $9.8 \times 10^{-8}$. We then partition each of the macro-elements
into 49 finite volumes using the algorithm above (this number being chosen so that the number of finite volumes is less than the number of degrees of freedom on a macro-element). We then employ our inverse transformation $\Pi^2$ to reconstruct the original solution. When we now compare the exact solution to the reconstructed solution, we obtain an $L_\infty$ error of $5.4 \times 10^{-5}$. This demonstrates that when less finite volumes are used compared to the number of degrees of freedom in the original element, there is a loss of precision in the reconstructed solution.

![Figure 7.10](image)

Figure 7.10: Macro-element discretization consisting of four quadrilaterals, each with 7th polynomial expansions (left). Finite volume partitioning consisting of 49 finite volumes per macro-element (center). Reconstructed solution using $\Pi^2$ operator (right).

From the study above we conclude that for the de-refinement algorithm to yield the original solution we must have that the number of finite volumes $M^d \geq R$ where $R$ is the number of degrees of freedom in the macro-element being split. Additional studies have shown that these results hold for all straight-sided elements: elements with curvature will require special attention.

The number of finite volumes derived above comes from interpolation theory estimates and does not consider how the finite volumes will be used. Another criterion that can be used for determining how many finite volumes into which to partition each macro-element is to consider a Kreiss-type advection argument [36], in which we consider how many finite volumes are necessary when implementing a finite volume advection scheme to maintain the same level of accuracy as the original high-order macro-element (or other high-order elements used throughout the solution domain) employing a high-order methodology like the discontinuous Galerkin method.
To illustrate this idea, we solve the one-way wave equation \( u_t + u_x = 0 \) on \([-2, 2]\) with periodic boundary conditions and with the following initial condition:

\[
u(x, t = 0) = 0.1 \mathcal{H}(-x)(\cos(2.0\pi x) - 1.0) + \mathcal{H}(x - 0.25)\mathcal{H}(0.5 - x)
\]

where \( \mathcal{H}(x) \) is the Heaviside function. This solution has two components, a smooth component which we will resolve with high-order elements, and a discontinuous component which we will attempt to resolve using finite volumes. The rationale behind this example is to illustrate a case in which high-order elements should be used for resolving smooth regions and low-order elements should be used for resolving regions with low regularity.

The solution domain is partitioned into four regions as shown in figure 7.11. The three high-order element regions consist of three \(11^\text{th}\) order elements. Three different finite volume cases were considered: Case one in which \(M = 40\) finite volumes were used on the interval \([0, 1]\), case two in which \(M = 80\) finite volumes were used on the interval \([0, 1]\), and case three in which \(M = 100\) finite volumes were used on the interval \([0, 1]\).

![High-Order Element](image)

Figure 7.11: Diagram showing which discretization methods were used in the solution of the problem discussed above.

In figure 7.12 we compare the exact solution to the three cases at time \(T = 0.3\). For all three cases, in the high-order elements, the numerical approximation matches the exact solution to four significant digits. As we successively refine the number of finite volumes, the dissipation and dispersion errors inherent in the low-order finite volume method become less pronounced.

To maintain the same level of accuracy in the finite volume region as is being attained in the high-order elements would require thousands of finite volumes, far more than the interpolation theory estimate derived earlier. From this we conclude that in order to maintain accuracy, far more finite volumes are required than the interpolation estimates require, and hence by using this criterion we will always meet the interpolation criterion.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 7.12: Comparison of the exact solution to the three cases discussed in the text evaluated at $T = 0.3$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
given previously. This criterion, however, is solution dependent, and hence a means of determining this number automatically needs future researching.

7.2.2 Flow Simulations

To demonstrate the use of the non-conforming finite volume refinement discussed above, we present both two-dimensional and three-dimensional computations of supersonic flow past a NACA 0015.

Supersonic Compressible Flow Past a 2D NACA 0015

We now present the two-dimensional supersonic compressible flow past a NACA 0015. The chord Reynolds number is $Re = 10,000$, and the Mach number is $Ma = 1.2$. The original mesh before refinement is shown in figure 7.13. It consists of 400 quadrilateral elements.

![Figure 7.13: Full domain (left) and local tessellation (right) for the supersonic simulation around the NACA 0015 airfoil. All dimensions are in units of chord length.](image)

In figure 7.14 we present the mesh after non-conforming finite volume refinement. The refinement region was chosen to capture the bow shock created by the airfoil geometry when supersonic conditions are used.

To clarify the position of the non-conforming elements in figure 7.14, we present in figure 7.15 a diagram showing which discretization methods were used and their positions...
Figure 7.14: Full domain (left) and local tessellation (right) for the supersonic simulation around the NACA 0015 airfoil after non-conforming refinement has been accomplished. Non-conforming refinement for case one is shown. All dimensions are in units of chord length.

with respect to the airfoil.

Four computations were accomplished in this study. The first computation (finite volume), our control computation, consists of a 12,000 element finite volume mesh. The second computation (case 1) consists of non-conforming finite volume refinement as shown in figure 7.14, with each partitioned element being split into 25 finite volumes and 4th order polynomials being used in all unsplit elements. The third computation (case 2) consists of non-conforming finite volume refinement as shown in figure 7.14, with each partitioned element being split into 36 finite volumes and 11th order polynomials being used in all unsplit elements. The fourth computation (case 3) consists of non-conforming
finite volume refinement as shown in figure 7.14, with each partitioned element being split into 121 finite volumes and 11th order polynomials being used in all unsplit elements.

In figure 7.16 we present density contours for both the finite volume case and case two. In figure 7.17 we present Mach number contours for both the finite volume case and case two. All three mixed discretizations exhibit a larger density bubble in front of the airfoil compared to the finite volume simulation. Adding more finite volumes to the mixed discretization does not appear to change the size of this bubble dramatically. To more fully examine, we will next examine a line slice of both the density and the Mach number at a position slightly above the airfoil.

In figure 7.18 we compare line slices of the density (left) and the Mach number (right) taken at $y = 0.2$ (the airfoil centerline lies at $y = 0$) at one instant of time. The airfoil lies between $x = 0$ and $x = 1$. Case two and three, with increased refinement in the wake, yields similar results to the finite volume solution near the region above the airfoil body. Cases one through three do not have as sharp a shock interface as the fully finite volume solution, and even with adding more finite volumes the sharpness of the interface does not increase dramatically. We speculate that this may be due to two reasons. The first reason is that a relatively low number of finite volumes were used in the mixed discretization case.
Figure 7.17: Mach number contours for supersonic flow past a NACA 0015. The finite volume solution using 12,000 quadrilaterals (left) and the case two mixed non-conforming finite volume/high-order element computation (right) are presented.

compared to the fully finite volume case. The second reason may be that at this Reynolds number the choice of the viscous fluxes may affect how sharp the interfacing is. Both are areas to be examined in the future.

Supersonic Compressible Flow Past a 3D NACA 0015

We now present in figure 7.19 the three-dimensional supersonic compressible flow past a NACA 0015. The chord Reynolds number is $Re = 10{,}000$, and the Mach number is $Ma = 1.2$. The original mesh was created as an extrusion of the two-dimensional mesh shown in figure 7.13. Four extrusion layers were added creating a 1600 hexahedra mesh in which the wing length is twice the chord length. Non-conforming finite volume refinement was then performed in the same region as shown in figure 7.16, with each hexahedron being split into eight finite volume hexahedra. Fourth-order polynomials were used in all non-refined elements.

This example demonstrates the flexibility of the non-conforming finite volume refinement when solving fully three-dimensional supersonic compressible flows.
Figure 7.18: Line slices of the density (left) and Mach number (right) taken at $y = 0$ (where the airfoil centerline is at $y = 0$. All four computations are presented.

Figure 7.19: Density contours for three dimensional supersonic flow past a NACA 0015. On the left we present line contours of the density at two positions on the wing (at 20% and 80% of the length). This is done so that the full length of the wing can be seen. On the right, we present a flood contour cut-plane of the density taken at the middle of the wing.
Chapter 8

Summary

In this thesis we have achieved the following goals:

- We examined the ramifications of different flux choices when solving elliptic and parabolic problems using the discontinuous Galerkin method.

- We extended the arbitrary Lagrangian-Eulerian (ALE) work of Lomtev et al. [52] to encompass all element types by formulating a generalized graph theory algorithm for mesh movement.

- We presented the first example of coupling a spectral/hp element fluids code (NekTao) with the hp-FEM structural code StressCheck.

- We identified polynomial aliasing as a potential problem in the simulation of high Reynolds number fluid flow computations, and proposed over-integration rules which alleviate the problems associated with aliasing.

- We extended the spectral vanishing viscosity (SVV) work of Karamanos et al. [43] to filter on an orthogonal set of modes, and demonstrated the quality of this extension in the computation of turbulent channel flow.

- We formulated SVV for the discontinuous Galerkin method and applied it to computation of the compressible Navier-Stokes equations.
• We proposed an alternative form of LES using dynamic SVV and provided an example of its use.

• We successfully implemented non-conforming discretizations within the discontinuous Galerkin framework for solving the compressible Navier-Stokes equations under supersonic conditions.

In concluding this thesis, we will now suggest some areas of future research which follow from some of the work presented herein.

We believe that future emphasis will need to be focussed on the development of fast, easily parallel algorithm. Some of the issues to be addressed are the following:

• Development and implementation of good preconditioners for the LDG method should be examined. Although accurate, the time step restriction which is experienced when explicit time stepping is used becomes computationally prohibitive. Implicit LDG may be a solution, but in solving the system implicitly one destroys the nice locality property which LDG provides.

• Computational complexity should be analyzed systematically for the three fluxes discussed. Although locality arguments give some indication of the communication necessary when employing these fluxes, it does not answer the question of how costly each flux is to compute.

• Stabilizing factors for diffusive and elliptic problems need further study to determine what constraints can be placed on the tuning parameter. Studies such as those done for penalty terms (see [31]) should be mimicked for the DG stabilization factors.

Concerning robustness, there has been much interest in both spectral vanishing viscosity and spectral filtering as presented in [33] as a means of stabilizing solutions. In [33] a discussion demonstrating the equivalence of SVV and polynomial filtering under certain constraints are examined. One future area of research is to examine the differences between SVV and polynomial filtering when solving non-linear hyperbolic problems. The following areas should be addressed:
• For non-linear problems, are there clearly distinguishable features between SVV from polynomial filtering? Quantitative studies should be performed which examine the accuracy when SVV or filtering is used.

• An operation count study should be performed to determine the cost difference between implementing these two methodologies. If there are little differences between the solutions obtained using SVV and filtering, then the computationally cheaper of the two should be employed.

Proposals can also be made by considering recent trends in computer science. A future area of research could be using computational agents for determining when in simulations certain methods should be employed. Agent technology has been successfully used both in computational commerce and in large-scale engineering modeling. The idea of *programming agents* which would utilize *a posteriori* error estimators and regularity indicators to determine when particular methods should be used would greatly enhance current static procedures which are employed in most computations. Extending the idea of dDNS presented in [45], agent technology would make simulations robust by determining such things as when and when not to use SVV or filtering, when or when not to use high-order or low-order methods, or when or when not to remesh, and acting upon its decisions. And by recording its decision making process, it provides a clear record of what actions were taken and why. No longer will the question be whether someone uses high-order or low-order methods, or whether someone uses filtering or not, but rather, how were all of these techniques combined appropriately to yield an accurate solution in a robust manner? We hold that only in the synergistic combinations of all of these concepts will flexible, accurate, and robust methodologies be devised.
Appendix A

Documentation of the Compressible $N\varepsilon\kappa T\alpha r$ Code

A.1 Introduction

In figure A.1 we present the current status of the compressible $N\varepsilon\kappa T\alpha r$ code.

Figure A.1: The current functionality tree for the compressible $N\varepsilon\kappa T\alpha r$ code.

In the figure above, the following notation was used:

$ALE = $ Arbitrary Lagrangian-Eulerian

$SVV = $ Spectral Vanishing Viscosity

$NC = $ Non-Conforming
In the software tree described above, the 2D/3D bifurcation is enforced by having two different source directories. Compress2d and Compress3d and a CommonCode directory containing routines common to both programs. All functionality in the subtree below 2D is contained within Compress2d and the CommonCode directory, and all functionality in the subtree below 3D is contained within Compress3d and the CommonCode directory. In figure A.1 solid lines designate features which have been tested for all element types; dashed lines denote features which have only been used at the current time with certain element types.

For a discussion of the functionality of the code presented above, see [73],[51],[52], and previous sections of this work.

A.2 Compilation Guide

All of the functionality described in figure A.1 is contained within the following seven directories:

/ Hybrid/V eclib
/ Hybrid/ Hlib
/ Hybrid/ include
/ Hybrid/ Flags
/ Hybrid/ CommonCode
/ Hybrid/ Compress2d
/ Hybrid/ Compress3d

The first two directories Veclib and Hlib contain mathematical operations and spectral/hp element operation common to both the incompressible and compressible versions of NekTar. No special compilation options are needed for Veclib. Special compilation options (see below) are needed for compiling a version of Hlib appropriate for the compressible code. The directory include contains user-defined header files used by both the library and source codes. The directory Flag contains compilation files having the appropriate compilation flags for different architectures. The directory CommonCode contains code common to both the 2D and 3D compressible implementations (ALE functionality.
SVV functionality, and NC functionality). The directory Compress2d contains the source files for the compressible 2D code, and the directory Compress3d contains the source files for the compressible 3D code.

We will now present a brief discussion on how to compile the compressible code.

A.2.1 Veclib

Inside the Veclib directory, type:

```bash
gnake
```

Move libvec.a library which is created to the appropriate architecture directory where the Hlib library will be stored (see below).

A.2.2 Hlib

Inside the Hlib directory, determine which architecture on which you will be running (for the purposes of this example, we will assume that we are compiling on an IRIX64 (SGI) architecture). In the example below, we will assume that we are compiling a debug (dbx) version of the code. If an optimized version is desired, type opt instead of dbx in the examples below.

If compiling a serial version of the libraries, enter the IRIX64C directory and type:

```bash
gnake COMPRESS=1 dbx
```

If compiling a parallel version of the libraries, enter the IRIX64CP directory and type:

```bash
gnake COMPRESS=1 PARALLEL=1 dbx
```

Observe that ‘C’ is used to denote the Hlib library compilation to be used for the compressible code, and ‘P’ denotes the Hlib library compilation to be used for running the parallel version of the code.

A.2.3 Source Directories

Inside of the Compress2d or Compress3d directory, determine which architecture on which you will be running (for the purposes of this example, we will assume that we are compiling
on an IRIX64 (SGI) architecture). In the example below, we will assume that we are compiling a debug (dbx) version of the code. If an optimized version is desired, type opt instead of dbx in the examples below.

To compile the serial version with no special options type:

```bash
gmake dbx
```

To compile with special options, use the flag convention as below:

```bash
gmake <FLAGS> dbx where flags can be
```

1. ALE=1
2. PARALLEL=1
3. SVV=1
4. NCELEMENT=1

Hence, to compile a parallel, ALE, SVV version of the code you would type:

```bash
gmake PARALLEL=1 ALE=1 SVV=1 dbx
```

Only those combinations denoted in figure A.1 have been tested.

### A.3 Execution Guide

#### A.3.1 The rea File

The rea file used by compressible NekTAR is almost identical to the one used by the incompressible version. In this discussion, we will point out some of the modifications made for the compressible version.

**Header Information**

The basic header information for a compressible rea file is as follows:

```
***** PARAMETERS *****
GRIDGEN --> Nektar 2D
```
2 DIMENSIONAL RUN
9 PARAMETERS FOLLOW

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>MODES</td>
</tr>
<tr>
<td>1.0/10000.0</td>
<td>KINVIS</td>
</tr>
<tr>
<td>10000</td>
<td>NSTEPS</td>
</tr>
<tr>
<td>0.0001</td>
<td>DT</td>
</tr>
<tr>
<td>100000.</td>
<td>IOSTEP</td>
</tr>
<tr>
<td>4.</td>
<td>EQTYPE</td>
</tr>
<tr>
<td>1.</td>
<td>INTYPE</td>
</tr>
<tr>
<td>100</td>
<td>HISSTEP</td>
</tr>
<tr>
<td>1.2400794</td>
<td>WTEMP</td>
</tr>
</tbody>
</table>

where the parameters above are as follows:

- **MODES** - number of modes (per tensor product direction) to use
- **KINVIS** - kinematic viscosity
- **NSTEPS** - number of steps to run
- **DT** - timestep
- **IOSTEP** - how often (in steps) to dump the checkpoint file
- **EQTYPE** - equation type: Euler = 3, Navier-Stokes = 4
- **INTYPE** - time integration order to be used (either 1.2 or 3)
- **HISSTEP** - how often (in steps) to dump history/force information
- **WTEMP** - wall temperature to use for wall boundary condition

Parameters also sometimes found in compressible rea files

- **FFX** - scalar forcing in x-direction
- **FFY** - scalar forcing in y-direction
• FFZ - scalar forcing in z-direction

• TREF - if this parameter is set, then the viscosity is calculated based upon Sutherland's Law (see [51])

Admissible Boundary Condition Information

Below we present the admissible boundary conditions for the compressible code. When a set of equations follows a boundary condition, this implies the lines similar to the ones listed must be present (with modifications to the exact values changed to meet the properties of the run desired). Below, 'rho' stands for density, 'u' stands for momentum in the x-direction, 'v' stand for momentum in the y-direction, and E stands for total internal energy. The examples below are given for 2D - in 3D, a 'w' condition denoting the momentum in the z-direction should be added following the 'v' condition.

• 'E' and 'P': elemental/periodic boundary conditions

• 'v': strong Dirichlet boundary condition

\[
\begin{align*}
\rho &= 1.0 \\
u &= 1.0 \\
v &= 0.0 \\
E &= 1.7400794
\end{align*}
\]

• 's': weak/flux Dirichlet boundary condition (used for both inflow and outflow)

\[
\begin{align*}
\rho &= 1.0 \\
u &= 1.0 \\
v &= 0.0 \\
E &= 1.7400794
\end{align*}
\]

• 'W': wall condition (strong form): no slip and no penetration

• 'm': moving wall condition (used in ALE)
\[ u = 1.0 \]

\[ v = 0.0 \]

- 'L': moving wall condition (used with structural solver)

As a reference, to compute the Mach number, the following equation is used:

\[
E = \frac{\rho(u^2 + v^2)}{2} + \frac{\rho(u^2 + v^2)[\gamma M^2]^{-1}}{\gamma - 1}
\]

where \( M \) denotes the Mach number, \( \rho \) is the fluid density, \((u,v)\) denotes the local fluid velocity and \( \gamma \) is the gas constant normally taken to be 1.4 for air. If given the Mach number, one can use the equation above to compute the appropriate energy to use in the simulation. If given the energy, one can compute the Mach number of the flow. Notice that Mach number is not specified explicitly in the compressible code - it is specified tacitly through the boundary/initial conditions for energy.

### A.3.2 Running the Code

To execute the code, type the following:

```
nektar2d -chk example.rea
```

where `nektar2d` is the executable name (which may vary depending on which versions you have compiled), `-chk` is a flag designating that the run should output checkpoint files, and `example.rea` denotes the rea file which you are using.

### A.4 Programming Guide

Within both the Compress2d and Compress3d directories, the following files with provide the basic functionality can be found:

- `drive.C`: contains the `main` function.
- `prepost.C`: pre-processing routines called from `drive.C`.
- *Compress.C*: pre-processing routines used for setting up edge/face structures.

- *advection.C*: routines for computing the advection operator.

- *visc.C*: routines for computing the viscous (diffusion) operator.

- *analyzer.C*: routines for outputting post-processing files (*checkpoint*, *history*, *forces*, etc.).

- *forces.C*: routines for computing forces on the body (wall conditions in stationary version and moving walls in ALE/structural versions).

### A.4.1 Miscellaneous Programming Notes

We now present a collection of programming notes which have been compiled to help people when working with the compressible code:

- Almost all operations in this version of the compressible code are done in physical space. Storage is done in the `double **h` array for 2D elements and the `double ***h_3d` array for 3D elements.

- Upwind fluxes [52] are used for the advection terms.

- Bassi-Rebay [7] fluxes are used for the viscous terms.

- To accomplish the addition of the volume and surface integration, the trick of Warburton ([73] section 4.1.2) is employed. This trick consists of using the routine "Add_Surface_Contrib" combined with the "Ofwd" and "Obwd" routines.
Bibliography


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


