5-2012

Development and Verification of a Navier-Stokes Solver with Vorticity Confinement Using OpenFOAM

Austin Barrett Kimbrell

University of Tennessee, austin.b.kimbrell@gmail.com

Recommended Citation


https://trace.tennessee.edu/utk_gradthes/1173
To the Graduate Council:

I am submitting herewith a thesis written by Austin Barrett Kimbrell entitled "Development and Verification of a Navier-Stokes Solver with Vorticity Confinement Using OpenFOAM." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.

John S. Steinhoff, Major Professor

We have read this thesis and recommend its acceptance:

K. C. Reddy, Joseph C. Yen

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)
Development and Verification of a Navier-Stokes Solver with Vorticity Confinement Using OpenFOAM

A Thesis Presented for the Master of Science Degree
The University of Tennessee, Knoxville

Austin Barrett Kimbrell
May 2012
ACKNOWLEDGEMENTS

I would like to express sincere appreciation to my advisor Dr. John Steinhoff for his guidance in developing this work and providing a challenging, yet enjoyable experience. I would also like to thank my other advising committee members, Dr. K.C. Reddy and Dr. Joseph Yen, for their time and helpful comments. I would like to especially acknowledge Dr. Yen for all of the discussion, insight, and advice he has provided over the past few years on topics both work and school-related, which has helped grow my technical knowledge and expertise in computational methods and aerodynamics. He is not only a friend but a true mentor and I am very grateful to work with him.

I offer special thanks and acknowledgement to Chris Connor and Dr. Ed Duell at Jacobs Technology for all of the support and opportunities for technical advancement given to me. Jacobs Technology has been an invaluable resource the past few years by providing funding for my continued education as well as the available computational resources to complete this work. The working environment at Jacobs is truly unique and I am thankful to be a part of it.

Lastly, I owe the greatest thanks to my wife, Oxana, and my daughter, Sophia, for all of their support and understanding during this whole endeavor. It wasn’t always easy, but they were always there for me to provide encouragement and keep me on the path to success.
Vorticity Confinement (VC) is a numerical technique which enhances computation of fluid flows by acting as negative diffusion within the limit of vortical regions, preventing the inherent numerical dissipation present with conventional methods. VC shares similarities with large eddy simulation (LES), but its behavior is based on a stable negative dissipation of vortical structures controlled by the automatic balance between two parameters, $\mu$ [mu] and $\varepsilon$ [epsilon].

In this thesis, three-dimensional, parallel-computing Navier-Stokes solvers with VC are developed using the OpenFOAM computational framework. OpenFOAM is an open-source collection of C++ libraries developed for computational fluid dynamics. Object-oriented programming concepts are used to develop the finite volume solvers, which introduce the VC source term into the governing equations as a body force. An immersed boundary method is implemented with the VC module to mitigate limitations of body-fitted grids.

The developed solvers are examined using two-dimensional boundary layer simulations, which demonstrate that for a given range of confinement parameters the boundary layer can be relaxed to a desired height to approximate a turbulent boundary layer thickness. Unlike wall function models, however, the VC boundary layer can still separate in an adverse pressure gradient. The application of VC to a two-dimensional advecting compact vortex results in the propagation of the vortex without dissipation. Solutions for a three-dimensional backward-facing step are validated against experimental data. The VC simulations show excellent agreement with experimental data for a fixed value of $\mu$ [mu] and a given range of $\varepsilon$ [epsilon]. Coarsening the mesh increases inherent numerical dissipation and requires using a smaller value of $\mu$ [mu] to show good agreement with experimental data. Turbulence kinetic energy spectra exhibit a $-5/3$ slope inertial wavenumber range indicating proper turbulence cascading using the VC model. Simulation of a Formula One racecar represented by an
immersed surface demonstrates the suitability of VC for fast prototyping. A time-accurate VC analysis on a 3,400,000 cell coarse mesh appears more realistic in the wake region than a steady RANS simulation on a 30,000,000 cell mesh. The VC solution appears visually comparable to an LES solution but represents a fraction of the computational cost.
# TABLE OF CONTENTS

Chapter 1 Introduction ........................................................................................................... 1  
  1.1 Objectives ...................................................................................................................... 3  
  1.2 General Approach .......................................................................................................... 4  
Chapter 2 Vorticity Confinement ............................................................................................ 6  
  2.1 VC1 Formulation ............................................................................................................ 7  
    2.1.1 Surface Confinement ................................................................................................ 9  
  2.2 VC2 Formulation ............................................................................................................ 10  
  2.3 Compressible Vorticity Confinement .............................................................................. 11  
  2.4 Dimensional Analysis of $\varepsilon$ ............................................................................... 13  
    2.4.1 Dynamic Vorticity Confinement ............................................................................. 15  
Chapter 3 Introduction to OpenFOAM ..................................................................................... 16  
  3.1 Introduction to Object-Oriented Programming .............................................................. 17  
  3.2 Programming in OpenFOAM .......................................................................................... 19  
  3.3 OpenFOAM Data Structure ............................................................................................ 20  
  3.4 Standard Case Structure ............................................................................................... 21  
    3.4.1 system Directory .................................................................................................... 24  
    3.4.2 constant Directory ................................................................................................. 24  
    3.4.3 Time Directories .................................................................................................... 26  
  3.5 OpenFOAM Utilities ....................................................................................................... 26  
    3.5.1 Pre-processing ....................................................................................................... 27  
    3.5.2 Meshing ................................................................................................................ 27  
    3.5.3 Domain Decomposition ......................................................................................... 28  
    3.5.4 Post-processing ..................................................................................................... 29  
    3.5.5 Run-time Processing ............................................................................................... 30  
    3.5.6 Solution Monitoring ............................................................................................... 30  
    3.5.7 Additional Community-developed Tools ............................................................... 31  
Chapter 4 Solution Methodology ............................................................................................. 32  
  4.1 Finite Volume Approach ............................................................................................... 33
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5.3 Vorticity Field Comparison</td>
<td>124</td>
</tr>
<tr>
<td>5.5.4 Streamline Comparison</td>
<td>127</td>
</tr>
<tr>
<td>Chapter 6 Conclusion</td>
<td>134</td>
</tr>
<tr>
<td>6.1 Future Work</td>
<td>137</td>
</tr>
<tr>
<td>Bibliography</td>
<td>139</td>
</tr>
<tr>
<td>Appendix</td>
<td>148</td>
</tr>
<tr>
<td>Vita</td>
<td>154</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1. Simulation parameters for initial simulations in 5.4. .............................82
LIST OF FIGURES

Figure 1. OpenFOAM C++ function call identical to Equation (3.1) ..............20
Figure 2. Two cell polyhedral mesh [45]. ................................................22
Figure 3. OpenFOAM linked list mesh representation of two cell mesh in Figure 2 [45]. .................................................................22
Figure 4. OpenFOAM case directory structure. ...........................................23
Figure 5. Flow chart comparison of PISO and SIMPLE algorithms ...............42
Figure 6. OpenFOAM C++ code snippet for the PISO algorithm ...............45
Figure 7. C++ code snippet for harmonic mean algorithm .........................48
Figure 8. Computational mesh for flat plate case study. Processor subdomains have been separately colored to show domain decomposition ...............53
Figure 9. Boundary conditions for flat plate case study .........................53
Figure 10. Velocity magnitude contour for flat plate case without VC ..........55
Figure 11. Vorticity magnitude contour for flat plate case without VC ..........55
Figure 12. Boundary layer profile of flat plate case without VC at various locations along the plate. (x) refers to the distance from the leading edge. 56
Figure 13. Varying VC1 confinement strength and its effect on boundary layer height at downstream station x = 1.875 m. .................................58
Figure 14. Varying VC2 confinement strength and its effect on boundary layer height at downstream station x = 1.875 m. .................................58
Figure 15. Boundary layer profile from the flat plate simulation with VC1 confinement strength ε=0.25 ..............................................59
Figure 16. Boundary layer profile from the flat plate simulation with VC2 confinement strength ε=0.004 ..............................................59
Figure 17. Velocity magnitude contours for VC1 (top) and VC2 (bottom) simulations with confinement strength values of 0.25 (VC1) and 0.004 (VC2). Note that a constant diffusion (ν) of 0.001 is applied. .........................60
Figure 18. Vorticity magnitude contours for VC1 (top) and VC2 (bottom) simulations with confinement strength values of 0.25 (VC1) and 0.004 (VC2). Note that a constant diffusion (ν) of 0.001 is applied. .........................61
Figure 19. Boundary conditions for slanted plate case study. Immersed body is shown superimposed on the mesh. ........................................63
Figure 20. Velocity magnitude contour of immersed plate without VC ..........65
Figure 21. Vorticity magnitude contour of immersed plate without VC ..........65
Figure 22. Velocity magnitude contour for immersed plate with VC1..................................66
Figure 23. Vorticity magnitude contour for immersed plate with VC1..................................66
Figure 24. Velocity magnitude contour for immersed plate with VC2 and confinement strength of 0.004. The boundary layer is exhibiting roll-up behavior.................................................................68
Figure 25. Vorticity magnitude contour for immersed plate with VC2 and confinement strength of 0.004. The boundary layer is exhibiting roll-up behavior.................................................................68
Figure 26. Velocity magnitude contour for immersed plate with VC2 and first-order upwind differencing for the convection term. The added numerical diffusion causes the boundary layer to thicken and suppresses the transient instabilities. ..................................................................................69
Figure 27. Velocity magnitude contour for immersed plate with VC2 and second-order upwind differencing for the convection term, but with a reduced confinement strength of 0.0035. Boundary layer instabilities are no longer present in the solution.................................................................69
Figure 28. Boundary conditions for compact vortex advection case study.................71
Figure 29. Centerline vorticity magnitude after 1 s (100 time steps). Without VC the vortex strength has decreased an order of magnitude..............................73
Figure 30. Contour plots of vorticity magnitude at one second simulation duration (100 time steps). Note reduction of scale for simulation without VC. ..........74
Figure 31. Centerline vorticity magnitude after 10 s (1000 time steps). ..................75
Figure 32. Vorticity magnitude contour after 10 s (1000 time steps). Baseline results not shown because the vortex has been absorbed into the background flow......................................................................................76
Figure 33. Centerline vorticity magnitude after 40 s (4000 time steps). ............77
Figure 34. Vorticity magnitude contour after 40 s (4000 time steps). Low magnitude vorticity concentrations are visible in VC1 solution..........................77
Figure 35. Schematic representation of backward-facing step geometry (h = 0.0254 m)..................................................................................................................80
Figure 36. LES-typical non-uniform mesh used for initial VC studies in 5.4. Image depicts the 248 x 57 cell mesh in the X and Y coordinate directions; 20 uniform grid cells are used in the Z coordinate direction (normal to plane). .81
Figure 37. Iso-volume of mean streamwise velocity showing region of U < 0.0 for the laminar solution (no VC). .................................................................84
Figure 38. Mean streamwise velocity contour for laminar solution (no VC). ......85
Figure 39. Mean streamwise velocity contour for VC1 solution. Contour levels from Figure 38 apply.................................................................85
Figure 40. Mean streamwise velocity profiles from VC1 solution compared to experimental data, LES results, and the laminar solution. Y-axis position has been normalized by step height, while velocity has been normalized by the prescribed inlet velocity. ................................................................. 86

Figure 41. Streamwise RMS velocity profiles from the VC1 solution compared to experimental data, LES results, and the laminar solution. ................................................................. 87

Figure 42. Mean streamwise velocity contours for VC2 solutions; $\varepsilon = 1.6 \nu$ (top), $\varepsilon = 2.0 \nu$ (middle), $\varepsilon = 2.5 \nu$ (bottom). ................................................................. 89

Figure 43. Mean streamwise velocity profiles from the VC2 $\varepsilon = 1.6 \nu$ solution compared to experimental data, LES results, and the laminar solution. .................. 91

Figure 44. Streamwise RMS velocity profiles from the VC2 $\varepsilon = 1.6 \nu$ solution compared to experimental data, LES results, and the laminar solution. .................. 92

Figure 45. Mean streamwise velocity profiles from the VC2 $\varepsilon = 2.0 \nu$ solution compared to experimental data, LES results, and the laminar solution. .................. 93

Figure 46. Streamwise RMS velocity profiles from the VC2 $\varepsilon = 2.0 \nu$ solution compared to experimental data, LES results, and the laminar solution. .................. 94

Figure 47. Mean streamwise velocity profiles from the VC2 $\varepsilon = 2.5 \nu$ solution compared to experimental data, LES results, and the laminar solution. .................. 95

Figure 48. Streamwise RMS velocity profiles from the VC2 $\varepsilon = 2.5 \nu$ solution compared to experimental data, LES results, and the laminar solution. .................. 96

Figure 49. Horizontal section slices at $Y = 0$ height, level with the step leading edge ................................................................. 98

Figure 50. Instantaneous center plane slices of vorticity magnitude, from VISUALIZATION 1, laminar solution (top) and VISUALIZATION 3, VC1 solution (bottom). .................................................................... 99

Figure 51. Instantaneous volumetric-rendered images of vorticity magnitude, from VISUALIZATION 2, laminar solution (top) and VISUALIZATION 4, VC1 solution (bottom). Contour levels below 2000 are invisible to accentuate large coherent structures. Upper wall solution has been clipped to emphasize shear layer structures. ................................................................. 100

Figure 52. Instantaneous center plane slices of vorticity magnitude, from VISUALIZATION 5 with VC2 $\varepsilon = 1.6 \nu$ (top), VISUALIZATION 7 with VC2 $\varepsilon = 2.0 \nu$ (middle), and VISUALIZATION 9 with VC2 $\varepsilon = 2.5 \nu$ (bottom). ............ 102

Figure 53. Instantaneous volumetric-rendered images of vorticity magnitude, from VISUALIZATION 2, baseline laminar solution (top left), VISUALIZATION 6, VC2 $\varepsilon = 1.6 \nu$ (top right), VISUALIZATION 8, VC2 $\varepsilon = 2.0 \nu$ (bottom left), and VISUALIZATION 10, VC2 $\varepsilon = 2.5 \nu$ (bottom right). Contour levels below 2000 are hidden to accentuate large coherent
structures. Upper wall solution has been clipped to emphasize shear layer structures. ................................................................. 103

Figure 54. TKE spectrum versus wavenumber for VC2 $\varepsilon = 1.6\nu$ solution. .... 106

Figure 55. TKE spectrum versus wavenumber for VC2 $\varepsilon = 2.5\nu$ solution. .... 107

Figure 56. Coarsened backward-facing step mesh. 127 x 35 x 20 ................. 108

Figure 57. Mean streamwise velocity contours for coarse mesh solutions; baseline laminar solution (top), VC2 $\varepsilon = 2.0\nu$ (middle), VC2 $\varepsilon = 2.5\nu$ (bottom). ................................................................. 109

Figure 58. Mean streamwise velocity profiles from the coarse mesh comparing experimental data, the coarse laminar solution, the VC2 $\varepsilon = 2.0\nu$ solution, and the VC2 $\varepsilon = 2.5\nu$ solution. .............................................................................. 111

Figure 59. Streamwise RMS velocity profiles from the coarse mesh comparing experimental data, the coarse laminar solution, the VC2 $\varepsilon = 2.0\nu$ solution, and the VC2 $\varepsilon = 2.5\nu$ solution. .............................................................................. 112

Figure 60. Instantaneous center plane slices of vorticity magnitude on the coarse mesh, from VISUALIZATION 11 with the baseline laminar solution (top), VISUALIZATION 13 with VC2 $\varepsilon = 2.0\nu$ (middle), and VISUALIZATION 15 with VC2 $\varepsilon = 2.5\nu$ (bottom)............................................................................... 113

Figure 61. Instantaneous volumetric-rendered images of vorticity magnitude on the coarse mesh, from VISUALIZATION 12, laminar solution (top left), VISUALIZATION 14, with VC2 $\varepsilon = 2.0\nu$ (top right), and VISUALIZATION 16, with VC2 $\varepsilon = 2.5\nu$ (bottom). Contour levels below 2000 are hidden to accentuate large coherent structures. Upper wall solution is clipped as in Figure 53...................................................................................... 114

Figure 62. TKE spectrum versus wavenumber for coarse mesh VC2 $\varepsilon = 2.0\nu$ solution. ............................................................................................... 116

Figure 63. TKE spectrum versus wavenumber for coarse mesh VC2 $\varepsilon = 2.5\nu$ solution. ............................................................................................... 117

Figure 64. 5.5 case study domain with outer boundary dimensions marked. .... 119

Figure 65. Surface representation of the immersed F1 car. ......................... 120

Figure 66. Cartesian mesh used for 5.5 case study containing 3,400,000 cells; zoomed images show the spanwise refinement through the car (top left) and streamwise refinement along the centerline (top right). The full mesh (centerline slice) is pictured in the bottom image. ........................................... 122

Figure 67. Reference geometry for high resolution STAR-CCM+ study including upstream contraction. ................................................................. 123

Figure 68. Instantaneous snapshot of vorticity magnitude from VC2 solution at 3.0 s visualized with ray-tracing technique.............................................. 125
Figure 69. Front perspective view of vorticity magnitude iso-contour at a magnitude of 150 for time-accurate VC solution. ................................................................. 126

Figure 70. Front perspective view of vorticity magnitude iso-contour at a magnitude of 150 for steady-state high resolution RANS solution. .................... 128

Figure 71. Side view of vorticity magnitude iso-contour at a magnitude of 150 for the time-accurate VC solution. ................................................................. 129

Figure 72. Side view of vorticity magnitude iso-contour at a magnitude of 150 for the steady-state high resolution RANS solution. ........................................ 129

Figure 73. Comparison between vorticity magnitude iso-contours from the VC solution (top) and the high resolution RANS solution (bottom) for the turbulent wake region with mesh refinement visible.................................. 130

Figure 74. Streamline visualization based on time-averaged velocity for time-accurate VC simulation, colored by velocity magnitude. ................................. 131

Figure 75. Streamline visualization of steady-state RANS solution, colored by velocity magnitude. ................................................................. 132

Figure A-1. Iso-volume of mean streamwise velocity $U < 0.0$ for VC1 case, 5.4 case study................................................................. 149

Figure A-2. Iso-volume of mean streamwise velocity $U < 0.0$ for VC2 case, 5.4 case study with $\varepsilon = 1.6\nu$. ................................................................. 149

Figure A-3. Iso-volume of mean streamwise velocity $U < 0.0$ for VC2 case, 5.4 case study with $\varepsilon = 2.0\nu$. ......................................................... 150

Figure A-4. Iso-volume of mean streamwise velocity $U < 0.0$ for VC2 case, 5.4 case study with $\varepsilon = 2.5\nu$. ......................................................... 150

Figure A-5. TKE spectrum versus wavenumber for VC1 solution, 5.4 case study. ........................................................................................................ 151

Figure A-6. TKE spectrum versus wavenumber for VC2 $\varepsilon = 2.0\nu$ solution, 5.4 case study................................................................. 152

Figure A-7. Mesh from high resolution RANS study for 5.5 case study containing 30 million polyhedral cells; zoomed images show the local refinement around the car (top left) and streamwise refinement along the centerline (top right). The section of the mesh (centerline slice) corresponding to the VC2 simulation domain dimensions is pictured in the bottom image. .............. 153
LIST OF ATTACHMENTS

VISUALIZATION 1: 5.4 fine no-VC Vorticity magnitude.......................... Vis1.mpg
VISUALIZATION 2: 5.4 fine no-VC Volumetric vorticity .......................... Vis2.mpg
VISUALIZATION 3: 5.4 fine VC1 Vorticity magnitude .............................. Vis3.mpg
VISUALIZATION 4: 5.4 fine VC1 Volumetric vorticity .............................. Vis4.mpg
VISUALIZATION 5: 5.4 fine VC2-1.6 Vorticity magnitude ......................... Vis5.mpg
VISUALIZATION 6: 5.4 fine VC2-1.6 Volumetric vorticity ....................... Vis6.mpg
VISUALIZATION 7: 5.4 fine VC2-2.0 Vorticity magnitude ......................... Vis7.mpg
VISUALIZATION 8: 5.4 fine VC2-2.0 Volumetric vorticity ....................... Vis8.mpg
VISUALIZATION 9: 5.4 fine VC2-2.5 Vorticity magnitude ......................... Vis9.mpg
VISUALIZATION 10: 5.4 fine VC2-2.5 Volumetric vorticity ..................... Vis10.mpg
VISUALIZATION 11: 5.4 coarse no-VC Vorticity magnitude ...................... Vis11.mpg
VISUALIZATION 12: 5.4 coarse no-VC Volumetric vorticity .................... Vis12.mpg
VISUALIZATION 13: 5.4 coarse VC2-2.0 Vorticity magnitude .................... Vis13.mpg
VISUALIZATION 14: 5.4 coarse VC2-2.0 Volumetric vorticity .................. Vis14.mpg
VISUALIZATION 15: 5.4 coarse VC2-2.5 Vorticity magnitude .................... Vis15.mpg
VISUALIZATION 16: 5.4 coarse VC2-2.5 Volumetric vorticity .................. Vis16.mpg
VISUALIZATION 17: 5.5 Formula One car ray-traced vorticity ............... Vis17.mpg
CHAPTER 1
INTRODUCTION

Essentially, all models are wrong, but some are useful.
—George E. P. Box, *Empirical Model-Building and Response Surfaces*

Classically, fluid dynamics has been researched through experimentation and observation of fluid motion in physical environments. The development of the computer has led to the realization of computational fluid dynamics (CFD), the simulation of fluid motion using numerical methods in virtual environments. The motion of fluids is described by systems of partial differential equations (PDEs); however, these equations do not have exact analytical solutions in their general form. An approximate solution for a system may be obtained by discretizing the differential equations over a finite array of discrete locations (grid points) arranged within the domain of interest. To generate solutions over the discretized domain requires the use of mathematical models to approximate the physical processes involved as well as the overall behavior of the fluid. As succinctly stated by George Box, all models intrinsically contain some amount of error due to the approximations. Engineering judgment must be used to determine which models are the most appropriate for a given problem in terms of accuracy and computing speed. Therefore, the practical use of CFD as an engineering tool involves finding the balance between solution accuracy and computational cost.

As noted by Ferziger and Peric [1], approximate solutions obtained by numerical methods inherently contain both discretization and modeling errors. Discretization errors are defined as the difference between the exact solution of the system of PDEs and the exact solution of the set of algebraic equations obtained by discretizing the PDEs over a domain. Improving the discretization accuracy of a CFD simulation typically requires the use of higher order methods, which can be numerically unstable, or an increase in the number of grid points.
within the solution domain. As the grid becomes refined, the discretization error decreases and the truncation error of the approximate solution approaches a minimum limited by the order of accuracy of the solution method. Increasing grid points, however, adds considerable computational cost through the amount of time required to solve the discretized equations and typically requires large computational resources. For a complex engineering problem this increase in cost may be unfeasible due to calendar and budget constraints.

Modeling errors are defined as the difference between actual flow behavior and the solution of the mathematical model used to describe the flow behavior. It is important to realize that even with minimum discretization error the exact solution of the mathematical model may be physically incorrect if actual flow behavior deviates from the modeling assumptions. Traditional CFD has applied models with empirical constants or parameters which are calibrated such that the resulting solution becomes agreeable to experimental data. However, accurate results through calibration are obtainable only for similar types of flow problems, typically with simple geometries. As the physical processes become more complicated, accurately approximating the real flow behavior with a calibrated model becomes more difficult. Typically conventional models have difficulty resolving thin convecting flow structures such as vortices and vortex sheets, which generally dissipate due to the numerical diffusion introduced by modeling and discretization errors.

As an alternative to traditional modeling approaches, Steinhoff [2] has developed a numerical technique called Vorticity Confinement (VC). VC is an Eulerian fixed-grid methodology which exhibits similar characteristics to Lagrangian vortex methods in treating vortices and other thin vortical flow features. VC can work to confine thin regions of high vorticity to 2-3 grid cells and prevent the spreading of vortical structures due to numerical diffusion as they convect through the flow field. In this manner, the effects of the smaller flow structures on the bulk fluid are captured without modeling the details of the structures directly. The basic technique involves adding a forcing term to the
momentum equations which acts within the limit of vortical regions. The behavior of VC is governed by two parameters, \( \mu \) and \( \varepsilon \), representing the balance between isotropic diffusion and isotropic compression, respectively, applied to vortical regions. The application of VC to interacting vortex rings has demonstrated the ability of the method to accurately model merging and interaction of vortices over coarse grids [2,3]. Additional studies have shown considerable promise for the application of VC to a variety of flow conditions, and will be discussed further in detail in Chapter 2 [4,5,6,7]. The VC methodology is intended to provide a simple, efficient model which can be applied to many types of flow features such as boundary layers, free shear layers, and separated flows with minimal computational overhead.

1.1 Objectives

The inherent characteristic of potentially higher accuracy for lower computational cost suggests that Vorticity Confinement is an ideal model for fast prototyping and simulation during the engineering design process. However, to date the method has not been widely utilized in mainstream CFD simulation within the engineering community. This is possibly due to a lack of clarity regarding the proper range of values for the confinement parameters. Previous studies have suggested that the effects of VC are subjective and that the confinement parameters may be problem dependent [8,9,10]. More research is needed to evaluate the dependencies and behavior of the method and determine consistent values for the confinement coefficients which can apply to a wider range of flow conditions.

To perform simulations necessary for such an investigation, the primary focus of this thesis is the development and validation of a fully parallelized Vorticity Confinement solver using the OpenFOAM (Open Field Operation And Manipulation) computational framework. OpenFOAM is an open-source collection of C++ libraries, classes, and methods which together form a fully functional CFD package based on finite volume discretization. OpenFOAM has
seen extensive use in the academic community, and in recent years many commercial companies have begun using OpenFOAM for CFD simulation supporting a variety of industries [11]. The open-source nature of the package encourages users to freely share source code with the extended OpenFOAM community, thereby benefitting all users equally. The presence of a VC solver within OpenFOAM would potentially create interest in more applications of VC to different flow problems and expose a wider portion of the engineering community to the benefits of the methodology. Furthermore, tapping into the OpenFOAM user community would provide a resource for the calibration of the VC model parameters for a wide range of engineering problems.

After developing the flow solver, the complementary focus of this thesis is the exploration of the effects and behavior of Vorticity Confinement through simulation of both simple benchmark cases as well as more complex flow configurations. It is intended through this course of study to determine acceptable ranges of the confinement parameters for realistic flow simulation. Ultimately, with sufficient calibration and understanding of the confinement parameters it is hoped that the developed flow solver can be utilized as a production CFD code.

1.2 General Approach

The content of this thesis is separated into chapters pertaining to specific topics. Chapter 2 focuses on the development and background of the Vorticity Confinement method. The VC1 and VC2 formulations are discussed and various extensions of the method are treated in detail, including dimensional and dynamic confinement, in order to give the reader a full overview of all developments using the method. Chapter 3 gives an introduction to the open-source CFD simulation package OpenFOAM. This chapter serves primarily as a record for various details learned during the course of study regarding the workings of OpenFOAM. An overview of the general structure of the included libraries is presented along with a description of programming using OpenFOAM. A typical case file structure is outlined in order to provide the reader a level of
familiarity with the package. Chapter 4 describes the finite volume discretization of the conservation equations used within the developed flow solver and provides an overview of the pressure-correction methodology used in OpenFOAM to treat time-accurate and steady-state simulation. The implementation details of the VC source term and a parallel harmonic mean subroutine necessary for the VC2 formulation are described along with the immersed boundary formulation implemented for use with VC. Chapter 5 highlights verification of the flow solver with the presentation of results from five simulations conducted with the solver. A flat plate boundary layer is simulated in 5.1 to show the effect of VC on boundary layer flows, while in 5.2 an immersed slanted plate simulation demonstrates the viability of the implemented immersed boundary method. The effect of VC on an advecting compact vortex is shown in 5.3. Validation of the flow solver is performed in 5.4 with the comparison of a backward-facing step simulation to experimental data. The case study in 5.5 demonstrates the viability of the flow solver to perform fast three-dimensional complex flow simulation by modeling a Formula One racecar within a wind tunnel domain using the immersed boundary method. Chapter 6 discusses the subsequent findings and conclusions from this work, as well as future plans of study.
CHAPTER 2
VORTICITY CONFINEMENT

As stated in Chapter 1, Vorticity Confinement (VC) is a numerical technique conceived by Steinhoff [2], which has been developed to model thin vortical flow structures and convect them over long distances while minimizing dissipation. VC emerged from earlier work in the late 1980’s with vortex capturing methods such as the Vorticity Embedding technique, which was developed and validated for hovering rotor studies in helicopter research [12,13,14]. Similar to vortex capturing methods, VC does not attempt to resolve the internal features of a vortical structure; the Eulerian VC model confines the internal structure to 2-3 grid cells and prevents dissipation while preserving the general effect of the overall structure on the surrounding flow field. This can have the effect of reducing the computational requirement for complex flow simulations by enabling the resolution of vortical flow features on a coarser grid than required by other modeling techniques. VC acts through a correctional forcing term within the limit of the vortical regions and has no direct effect in the far field non-vortical solution. Furthermore, the method does not require any special discretization scheme, and can be used with both structured and unstructured grids, making it possible to integrate VC into established flow solvers.

Steinhoff and co-workers have developed two formulations of the Vorticity Confinement term which they name VC1 and VC2, respectively [15]. The VC1 formulation, which uses a first-order derivative in the confinement term, is treated in Section 2.1. Surface Confinement, a special formulation of VC1 developed for boundary layer flows, is discussed in Section 2.1.1. Section 2.2 introduces the VC2 formulation, an alternate form of the confinement term which uses a second-order derivative and explicitly conserves momentum [16]. A formulation of Vorticity Confinement has also been developed for compressible flows and is described in Section 2.3; while the compressible flow regime is out of the scope...
of this thesis, the formulation describes a method to introduce the confinement term to the integral form of the momentum equations, which is of central importance with respect to the finite volume formulation used in OpenFOAM. Finally, Section 2.4 details research of alternate formulations achieving a dynamic confinement strength parameter \( \varepsilon \), as opposed to the constant parameter used by Steinhoff in the original formulation.

2.1 VC1 Formulation

Much of the available literature concerns studies using the VC1 formulation. Steinhoff and Underhill [3] showed excellent agreement to experimental data for the collision of two coplanar vortex rings in a three-dimensional time-accurate calculation. With the addition of the confinement term, the rings were shown to automatically merge while preserving local vorticity magnitude. Vorticity Confinement has been applied to a variety of incompressible complex flow configurations using body-fitted grids as well as immersed boundaries on uniform grids. Steinhoff, Wenren and Wang [4] presented encouraging results for a pitching airfoil in dynamic stall, a Boeing 747 wing, and an Apache helicopter in forward flight. Simulations involving a helicopter fuselage [17], wake flows behind a three-dimensional cylinder and representative aircraft carrier [5], and flow over a 6:1 ellipsoid [18] exhibited good correlation to data (when available for comparison) and improved vortical flow predictions with the inclusion of the confinement term. A recent study implementing VC1 into a proprietary OpenFOAM solver showed a reasonable correlation to data for flow over an “Ahmed body” vehicle shape with various slant configurations, demonstrating the applicability of VC for blunt body flow separation [10]. Recently, VC1 has been implemented in a “smoothed particle hydrodynamics (SPH)” model for bubble prediction [19] and in a two level simulation (TLS) flow-scale separation method for water turbulence simulation [20], both of which demonstrate the versatility of Vorticity Confinement to integrate with various solution routines.
The basic VC methodology involves the addition of a discrete term to the discretized momentum conservation equations. The vector form of the conservation equations for unsteady, incompressible, isothermal flow, including the vorticity confinement term, is shown in Equations (2.1) and (2.2):

\[ \nabla \cdot \mathbf{U} = 0 \]  \hspace{2cm} (2.1)

\[ \partial_t \mathbf{U} + (\mathbf{U} \cdot \nabla) \mathbf{U} = -\nabla (p/\rho) + \nabla \cdot (\nu \nabla \mathbf{U}) - \varepsilon \mathbf{s} \]  \hspace{2cm} (2.2)

where \( \mathbf{U} \) is the velocity vector, \( p \) is pressure, \( \rho \) is density, and \( \nu \) is kinematic viscosity (equivalent to \( \mu/\rho \)). The confinement strength coefficient is represented by \( \varepsilon \). The parameter \( \nu \) represents both physical viscosity and explicit numerical viscosity necessary for the VC formulation. Dissipation multiplied by \( \nu \) and “anti-dissipation” multiplied by \( \varepsilon \) automatically balance over a range of values. The confinement term \( \mathbf{s} \) takes different forms depending on the confinement formulation. For the VC1 formulation, the corresponding term is shown in Equation (2.3):

\[ \mathbf{s} = \hat{n} \times \vec{\omega} \]  \hspace{2cm} (2.3)

where \( \hat{n} \) is a unit vector and \( \vec{\omega} \) is vorticity, calculated as the curl of the velocity field. For confinement in a vortical flow field away from a solid boundary, \( \hat{n} \) takes the form in Equation (2.4):

\[ \hat{n} = \frac{\nabla |\vec{\omega}|}{|\nabla |\vec{\omega}||} \]  \hspace{2cm} (2.4)

This is a unit vector which points towards the maximum gradient of vorticity. Effectively, the confinement term serves to convect vorticity back along the unit normal towards regions of high vorticity. This action energizes the vortical regions of the flow field, such as vortices and shear layers, and acts as a
counter to numerical dissipation inherent within the solution. It has been shown by Steinhoff [2] that with VC both mass and vorticity are globally conserved. It should be noted that the VC1 formulation does not conserve momentum exactly due to the self-induced velocity of vortical regions; however, this error is minimal for most problems and becomes significant only when long-term vortex trajectory accuracy is a concern [7]. The later development of the VC2 formulation, discussed in Section 2.2, provides an alternative that explicitly conserves momentum.

2.1.1 Surface Confinement

As previously noted, a special formulation of VC1 was developed for boundary layer regions [4,17]. This form explicitly convects vorticity back towards the boundary surface, defined by a function \( F \). If an immersed boundary method is used, then \( F \) is a smooth level set function which is defined as 0 at the boundary surface, positive outside the boundary in the flow field, and negative inside the boundary (interior of the immersed body). If body fitted grids are used then \( F \) can represent the distance to the wall. Studies of the application of Surface Confinement to missile aerodynamics showed improvement over simple Euler solutions, and some calibrations of the confinement strength parameter \( \varepsilon \) have been carried out for this specific application [21,22]. With Surface Confinement the following form of \( \hat{n} \) is used in cell layers near the boundary:

\[
\hat{n} = \frac{\nabla |F|}{\| \nabla |F| \|}
\]  

(2.5)

In theory, the form of \( \hat{n} \) in (2.4) should work in the boundary layer as well, since it acts to convect vorticity towards the highest local value which should be at the no-slip wall. However, in practice the form in (2.4) can become unstable close to the wall as the confinement strength coefficient is increased. The form of \( \hat{n} \) in (2.5) explicitly directs vorticity towards the wall and helps stabilize VC1 in the
boundary layer. The VC2 formulation, discussed in the following section, does not require any such explicit treatment in the boundary layer.

2.2 VC2 Formulation

The elements comprising the VC2 formulation were first introduced as a method to treat thin vortical flow features as “solitary waves”, such that the internal structures remain thin over a minimal region 2-4 grid cells wide, irrespective of time or distance traveled [23]. Steinhoff et al. [16] formalized the method for multidimensional calculations as a new Vorticity Confinement formulation (VC2) which explicitly conserves momentum for vortical regions. Results for simple three-dimensional blunt bodies (cylinder, sphere, and a half-sphere shell) indicated better vortical flow prediction using the new VC formulation which simulated higher Reynolds number effects [6]. Additional studies detailed by Steinhoff and Lynn [7] and Steinhoff et al. [18] concerning wing tip vortices and wake profiles behind various simple bluff bodies showed good agreement with experimental data as well as VC1 simulation results (where available). Lynn [24] developed turbulence models based on the VC2 framework which automatically determine an appropriate confinement strength based on conservation of kinetic energy, similar to dynamic sub-grid scale LES models, and validated the models against experimental and numerical solutions for the Taylor-Green vortex, a flat plate boundary layer, and a backward-facing step. A higher-order extension of the VC2 formulation has also been developed recently and has been verified against analytical pulse solutions [25].

As stated previously, the VC2 formulation has been developed to explicitly conserve momentum which is a shortcoming of the VC1 formulation. The formulation introduced by Steinhoff involves a similar additional confinement term as VC1 in Equation (2.2) with a new definition:

\[ \vec{s} = \nabla \times \vec{W} \]  

(2.6)
The term in the cross product is given for the discretized momentum equations as follows:

\[
\vec{W} = \frac{\vec{\omega}}{\vec{\omega}} \bar{h}(\vec{\omega}) \tag{2.7}
\]

\[
\bar{h}(\vec{\omega}) = \left[ \frac{\sum_i (\vec{\omega}_i)^{-1}}{N} \right]^{-1} \tag{2.8}
\]

\[
\vec{\omega} = |\vec{\omega}| + \delta \tag{2.9}
\]

The \(\vec{W}\) term in (2.7) can be interpreted as the normalized vorticity vector multiplied by the harmonic mean in (2.8). The value \(\delta\) in (2.9) is a small constant added to prevent division by zero in the \(\vec{W}\) term in (2.7). The harmonic mean is calculated over a localized stencil of \(N\) cells; for a uniform hexahedral mesh with face-based cell connectivity, this involves the center cell and all 4 (2D) or 6 (3D) neighboring cells. Depending on the underlying structure of the computational code (i.e., node-based connectivity) additional neighbors are possible in the harmonic mean calculation.

### 2.3 Compressible Vorticity Confinement

Traditional compressible flow solvers solve the conservation equations as a coupled system, rather than using the “split-velocity” methods popular for enforcing conservation in incompressible flow. As a result, the velocity correction used in incompressible VC must be introduced as a source term. A consistent methodology for compressible Vorticity Confinement was first suggested by Hu [26]. Hu’s research suggests that the confinement term may be interpreted as a body force, which can then be extended directly to the integral momentum equations and the rate of work done by the body force may be taken into account in the energy conservation equation. Previously, VC had been implemented in finite difference schemes; the integral formulation allows extension to finite-
volume schemes which form the basis of many commercial CFD codes. Hu and Grossman [27] displayed overall good quantitative and qualitative comparison to experimental data and numerical LES simulations for massive separation from bluff bodies using their compressible Vorticity Confinement formulation and uniform coarse grids. For uniform refined grids and an equivalent confinement parameter $\varepsilon$, however, the comparison to experimental data indicated some dependency of $\varepsilon$ on the grid density. Jafarian and Fard [9] further extended the compressible Vorticity Confinement formulation suggested by Hu with variable confinement parameters as functions of the spectral radii of the Jacobian matrices and the Jacobian matrices themselves. This extension, however, still required tuning the confinement parameter, although the resulting range was smaller than in the original compressible formulation.

In contrast to the compressible Vorticity Confinement suggested by Hu, Costes [28] showed substantial evidence that in fact the correct application of the confinement term to compressible flow does not require the additional work term in the energy conservation equation. Costes noted that, for the advection of a two-dimensional vortex, including the additional term in the energy equation resulted in a diffusion of density towards the exterior portion of the vortex. Removing the source term from the energy equation recovered the correct density prediction. This is not entirely surprising, as Hu [26] noted in the original research that this additional term had detrimental effects on total temperature. Costes surmised that associating physical quantities such as work from the body force is unnecessary because Vorticity Confinement is a numerical treatment without a physical analog.

While compressible flow is outside the scope of this thesis, the principles developed concerning the application of Vorticity Confinement to the integral momentum equations, and thereby finite-volume methods, will be followed. The finite-volume formulation detailed in Chapter 4 adheres to the suggestion of Costes [28] with the addition of the confinement term to the momentum equations alone.
2.4 Dimensional Analysis of $\varepsilon$

Previous studies [8, 9] have indicated that inconsistent values of $\varepsilon$ can result with the VC1 formulation when varying the mesh density or flow conditions. In the VC1 formulation, the confinement parameter $\varepsilon$ introduced in Equation (2.2) requires the dimension of velocity to ensure consistent dimensions with the other conservation equation terms. Towards the derivation of a consistent parameter, Fedkiw et al. [29] introduced a dependency on mesh size, $h$, such that the confinement strength scales with overall mesh size and varying mesh refinement. Löhner et al. [30] performed dimensional analysis on the confinement parameter and determined three potential scale factors which ensure a dimensionless confinement parameter for VC1:

$$\varepsilon_{ND} = \varepsilon |\mathbf{U}|$$  \hspace{1cm} (2.10)

$$\varepsilon_{ND} = \varepsilon h |\mathbf{\omega}|$$  \hspace{1cm} (2.11)

$$\varepsilon_{ND} = \varepsilon h^2 |\nabla |\mathbf{\omega}|$$  \hspace{1cm} (2.12)

Löhner et al. determined that the scaling in Equation (2.12) is desirable because it cancels the denominator of the unit normal vector in Equation (2.4), reducing the number of mathematical operations necessary for the confinement term. They suggested a refined length scale $h$ based on the characteristic length in the direction of the vorticity gradient, as well as an explicit cutoff threshold based on a local vorticity-based Reynolds number to remove the effect of Vorticity Confinement from highly refined boundary layer regions. Robinson [31] further extended the work of Löhner by factoring out vorticity magnitude from the confinement term, and using the dimensional scaling form in Equation (2.10). Robinson noted that the product of velocity and vorticity magnitude has identical dimensions to helicity, and suggested an alternate VC1 confinement term with the quantity of helicity as follows:
For Surface Confinement, rather than using the cutoff threshold suggested by Löhner, Robinson used the scaling term in Equation (2.12) to take into account mesh size reduction in the boundary layer region. Using this combination of confinement terms Robinson showed improvement for missile force and moment predictions using the formulation. The Robinson formulation has a key weakness, however, in that helicity is zero for two-dimensional simulations or for vortical structures normal to the freestream flow direction (tornado-like structures). This deficiency is not present in the Löhner scaling formulation. Furthermore, scaling the confinement parameter with helicity alone does not provide a dependency on mesh size which is necessary when using a non-uniform mesh. Butsuntorn and Jameson [32,33] extended Robinson’s helicity formulation to compressible flow rotorcraft calculations by including a non-dimensional scale factor based on mesh size:

\[
\bar{s} = \varepsilon_{ND} |\mathbf{U} \cdot \bar{\omega}| \left[ \left( 1 + \log_{10} \left( 1 + \frac{V}{V_{avg}} \right) \right)^{1/3} \left[ \frac{\nabla|\bar{\omega}|}{|\nabla|\bar{\omega}|} \times \frac{\bar{\omega}}{|\bar{\omega}|} \right] \right]
\]  

(2.14)

The formulation suggested by Butsuntorn and Jameson showed modest improvement to lift and drag coefficient predictions for rotorcraft flow [32]. They noted that the confinement parameter \( \varepsilon \) was dependent on the numerical scheme used in each calculation; as a result, because different numerical schemes have varying discretization error the confinement parameter value may vary by up to one order of magnitude between simulations. They suggested that the confinement parameter should be scaled based on numerical diffusion and discretization errors to improve the consistency of calibrated confinement strength values.
2.4.1 Dynamic Vorticity Confinement

As suggested by Butsuntorn and Jameson [32], for a consistent formulation to counter numerical error the confinement parameter should be based on numerical diffusion and discretization errors present for the given discretization scheme. Costes and Kowani [34] developed for VC1 a dynamic confinement parameter $\varepsilon$, constant over each grid cell, which was based on the assumption that the confinement term must cancel the truncation error of the momentum equations. This definition also resulted in the correct dimensionality of the confinement parameter. However, due to compressibility effects (the study was in the compressible flow regime) and the singularity of their formulation at the vortex center, they observed anomalies in the vorticity and density contours near the center of the vortex. Costes [28] reported improved results for a similar study using the VC2 formulation and a modified dynamic confinement formulation which resulted in the correct dimensionality for the VC2 confinement parameter and a constant value for the confinement strength averaged over the entire vortex.

Hahn and Iaccarino [35] detailed a similar approach towards creating a dynamic confinement parameter. They reasoned that the primary purpose of the VC term in the conservation equations is the cancellation of numerical diffusion and discretization errors; to that effect, they applied principles from dynamic LES modeling to determine a dynamic confinement strength and estimated numerical viscosity to be equivalent to the difference between central and upwind discretization schemes for the convection term. Results for the collision of two vortex rings and a counter-rotating vortex pair rebounding from a wall showed excellent agreement to analytical and experimental data using this formulation.
CHAPTER 3
INTRODUCTION TO OPENFOAM

OpenFOAM (Field Operation And Manipulation) is an open-source collection of C++ class libraries developed for simulating continuum mechanics [36]. The CFD package originates from work begun in the 1990’s at Imperial College in London under Prof. David Gosman and Dr. Raad Issa [37]. The principal developers, Henry Weller and Hrvoje Jasak, began marketing FOAM and in 2005 moved to two separate paths of development utilizing the open-source business model [37]. Weller maintains the official version of OpenFOAM which has closed-source development and periodically releases open-source version updates [38]. Jasak maintains a fork of OpenFOAM called the Extend-Project (also known as OpenFOAM-extend) which includes additional modeling capability not found in the official version and is centered on community-driven development [39]. The solver developed in this thesis utilizes libraries from OpenFOAM-extend version 1.6.

One of the difficulties experienced during the development of the current work was gathering the necessary knowledge to operate pre-existing OpenFOAM solvers as well as develop a new Vorticity Confinement solver for this work. OpenFOAM, like many open-source projects, is sparsely documented and the included manuals provide an incomplete description of components within the CFD package. Therefore, this chapter serves as a record of the information learned from various sources during the development of this work and provides a brief introduction to OpenFOAM with the intention of explaining specific fundamentals to the uninformed reader. Key aspects of the C++ programming language which are used throughout the libraries of OpenFOAM are presented in Section 3.1. The source code is an excellent example of proper C++ programming practices and can serve as its own documentation when included functionality is not described by the manuals. Key programming
concepts used in OpenFOAM are presented in Section 3.2. The OpenFOAM library structure and mesh structure are discussed in Section 3.3. Case file structure aspects which are pertinent to setting up, solving, and post-processing a simulation are presented in Section 3.4. Finally, descriptions of common utilities included in OpenFOAM are given in Section 3.5.

3.1 Introduction to Object-Oriented Programming

An explanation of the basic concepts of object-oriented languages is necessary to highlight the advantages of the C++ programming language. Object-oriented programming (OOP) typically makes use of the following concepts [40]:

- **Classes and Objects** – a class defines a data structure and all of the functions which operate on the data structure. An object is an instance of a class (instantiated at run-time). Objects share the same functions with other objects of the same class, but each object contains a copy of the original class data structure.

- **Member functions** – these are functions which are declared inside the class definition and are carried as part of the object. Two special member functions exist for each class: the constructor, which is called on initialization of the object, and the destructor, called on deletion of the object.

- **Private vs. Public members** – both member functions and data within a class require access permissions. A public member can be read or written by any function, for data, or called by any object, for functions. Private members can only be used by objects of the same defining class.

These concepts allow efficient organization and design of programs and are essential when dealing with very complex systems. Embodying the OOP paradigm, the C++ programming language was developed in the early 1980’s by Bjarne Stroustrup with the aim of combining the flexibility and efficiency of C
systems programming with OOP paradigms [41]. Support for the following paradigms is a cornerstone of the C++ programming language:

- **Data abstraction**: focuses on user-defined data types, in contrast to built-in types. Abstraction requires that a full set of operations be provided for each type of data [42]. Central concepts to this style are operator overloading and polymorphism. Operator overloading allows the user to define specific actions when using built-in operators with user-defined data types. Polymorphism is the ability to define an abstract base interface which applies to multiple objects at run-time.

- **Object-oriented programming**: introduces the concept of inheritance with class hierarchy. A base class defines generic properties of a class of objects and is specialized by derived classes which inherit the base class properties. Classes contain both data and functions, providing a method of encapsulation in which the classes are self-contained objects. This concept takes advantage of commonality in systems by using classes as building blocks to construct more complex objects.

- **Generic programming**: allows for algorithms to be parameterized for working with a variety of types and data structures, such that the algorithms are independent of the representative details [42]. Generic programming introduces the concepts of templates and container classes. Templates are classes or functions which are written with generic types, allowing them to work with multiple data types without being rewritten for each type. Containers are classes that can hold a collection of other objects.

While C++ was designed to support these programming paradigms in addition to traditional C systems programming techniques, it does not force any one particular style on all users [42]. The language simply makes it convenient to use any standard programming paradigm which fits the task at hand. This freedom is part of what makes C++ such a powerful scientific programming language.
3.2 Programming in OpenFOAM

By taking advantage of the aforementioned programming concepts, the developers of the OpenFOAM C++ library have taken a unique approach to solving PDEs by mimicking standard vector and tensor notation with high-level code, thus allowing complicated mathematical and physical models to be represented by high-level mathematical expressions [36]. OpenFOAM includes such C++ concepts as the use of polymorphism to represent subsets of turbulence models and boundary conditions with similar code interfaces, inheritance with a base class for mesh data and derived classes for access, and templates allowing code re-use for operations on different data types [36,43].

As an example of an OpenFOAM representation of a PDE, the momentum equation for unsteady isothermal flow is given as an example in Equation (3.1):

\[
\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot \rho \mathbf{U} - \nabla \cdot \mu \nabla \mathbf{U} = -\nabla \mathbf{p} \quad (3.1)
\]

The OpenFOAM code snippet in Figure 1, identical to Equation (3.1), is valid C++ syntax using the OpenFOAM high-level classes [44]. Operator overloading allows tensor algebra to be defined using basic mathematical operators for easily understandable equation syntax. Even though some of the syntax may not be immediately recognizable to one unfamiliar with OpenFOAM, the code clearly describes a time derivative term, a divergence term (convection), a laplacian term (diffusion), and a gradient term, and therefore provides an exact description of the discretized PDE. The terms in the equations are constructed to behave like their mathematical counterparts and hide the numerical details of the implementation by encapsulation [36]. Solution variables such as velocity and pressure are represented by tensor field data types which can describe scalars, vectors, and tensors. All physical variables are denoted by tensor fields sharing meaningful names: rho for density, mu for viscosity, U for velocity, and p for pressure. This design allows the user to focus on the correct description of the
PDE to be solved without worrying about discretization details which remain encapsulated within the high-level code. Furthermore, when dealing with unfamiliar models the code syntax can serve as its own documentation.

3.3 OpenFOAM Data Structure

The OpenFOAM lower-level libraries outline generic interfaces for data input and output to ensure consistency throughout the simulation environment and provide re-usable components as building blocks for higher-level libraries. For example, the lower-level libraries define discretization and parallelization details which are then implicitly used in higher-level code. Standalone executables are generated for utilities and solvers; generally, standalone solvers are written to manage one specific physics regime, such as buoyancy simulation or combustion. Multi-physics simulation requires combining existing solvers to create new customized solvers. This component-based approach gives the user a great amount of flexibility which may not be available in commercial CFD codes, and the availability of all source code eliminates the ‘black box’ factor in commercial codes in which the details of model implementation are hidden.

To discretize the problem domain, OpenFOAM uses an internal unstructured mesh format which supports tetrahedral, hexahedral, and polyhedral cell types. All variables are defined within the same control volumes, an approach which is typically called a collocated or non-staggered grid.
arrangement [1]. All meshes are explicitly three-dimensional; one- or two-dimensional calculations are performed by using special boundary conditions on any plane(s) normal to the coordinate direction(s) of interest which switch off the solution algorithm in the unnecessary direction(s). OpenFOAM requires the following mesh quality criteria to be met, as stated in the official User Guide [44], for a mesh to be computationally valid:

- **Contiguous** – cells must cover the full computational domain and not overlap one another.
- **Convex** – every cell must be convex such that the cell center must be contained within the cell.
- **Closed** – each cell should be closed, such that the sum of all face area vectors oriented outwards for each cell should equal zero, and each edge is used by exactly two faces of the given cell.
- **Orthogonality** – for each internal face, the angle between the face area vector and the center-to-center vector must be less than 90°.

Figure 2 and Figure 3 detail a polyhedral cell illustration and its corresponding OpenFOAM mesh representation. As shown in Figure 3, the polyhedral mesh in OpenFOAM is represented with four linked lists: a list of point coordinates, a list of cell faces with references to the point coordinates, a list of owner cells for each face, and a list of neighbor cells for each face [45]. This well-defined approach is optimal for interfacing with the unstructured mesh during the solution process.

### 3.4 Standard Case Structure

An OpenFOAM simulation is defined by a specific file structure which holds the setup information, mesh definition, and solution files. The standard file structure for a serial (single processor) case is shown in Figure 4. Each simulation directory holds three sub-directories: the `system`, `constant`, and ‘time’ directories (named using the physical time step or iteration step identifier).
Figure 2. Two cell polyhedral mesh [45].

Figure 3. OpenFOAM linked list mesh representation of two cell mesh in Figure 2 [45].
Figure 4. OpenFOAM case directory structure.
3.4.1 system Directory

This directory contains discretization and solution procedure information. General solver and utility settings in OpenFOAM are controlled by dictionary files which have the -Dict suffix. These files are read dynamically during solver execution and allow parameters to be varied during runtime. The following files are located within the system directory:

- controlDict – this dictionary contains run control parameters, such as start/end time, time step controls, and data output settings. Additional data output can be defined here using pseudo-code user functions described in Section 3.5.5.
- fvSchemes – this file defines the discretization schemes used for each algebraic tensor operation and each high-level mathematical expression.
- fvSolution – the iterative equation solver settings, tolerances, and specific algorithm control parameters are defined within this file.
- decomposeParDict – this file defines domain decomposition parameters for use with the decomposePar tool for parallel simulation.

3.4.2 constant Directory

The constant directory holds the mesh description and any files with the -Properties suffix describing required physical properties, such as viscosity. Additionally, turbulence model parameter files are located here if necessary for the simulation. The polyMesh directory contains the following mesh representation files:

- points – the points file is an ordered list of vectors describing the three-dimensional location of each point of the mesh in sequential order. Each row contains the global x, y, and z coordinates of the point in dimensions of meters.
- faces – the faces file contains the faces defined by the vertices in the points file. Each row of the list corresponds to a face in sequential
order and is comprised of a counter indicating the number of vertices making up the face, followed by a vector with the corresponding vertex labels. Because OpenFOAM supports polyhedral meshes, faces can have more than three (tetrahedral mesh) or four (hexahedral mesh) points.

- **owner** – each face in OpenFOAM is associated with an ‘owner’ cell and a ‘neighbor’ cell. The owner file is a sequential list of cell numbers which directly corresponds to the list in the faces file.

- **neighbour** – the neighbour file is a sequential list of cell numbers, similar to the owner file, except the listed cell numbers correspond to the neighbor cells for each face. Boundary faces are given a -1 index within this file as each boundary face belongs to an owner cell only.

- **boundary** – this file contains a list of patches (geometrical definitions of the boundaries) for the simulation. The number of corresponding patches is required at the beginning of the list. Each patch is given a name and described by a dictionary containing three entries. The geometrical definition of the patch is the type entry, and takes values of patch (used to apply most boundary conditions), wall (denotes boundaries for distance to wall calculations), and empty (defines patches to be ignored for one- or two-dimensional calculations). The faces comprising the patch are defined by a beginning face index with the startFace entry and the number of subsequent faces in the nFaces entry.

The polyMesh directory also contains the blockMeshDict file which is used for input to the blockMesh mesh generation utility. This utility will be discussed in greater detail in Section 3.5.2. The other directory in the constant directory, but not required at simulation runtime, is the triSurface directory. Any necessary surface definitions, such as Stereolithography (STL) files, are located here.
3.4.3 Time Directories

The time directories are named using the specified numerical format for the time step in the *controlDict* file in time-accurate simulations, or the iteration number in steady-state simulations. All data necessary for post-processing the simulation and for restarting at a specified time step or iteration are contained within the time directories. These directories are created dynamically by the running solver as the solution is generated; if a directory already exists, the solution files contained within the existing directory will be overwritten.

For each simulation, initial conditions are defined within the *0* directory. Every initialized simulation field is required to have a corresponding file in the *0* directory; for example, the velocity and pressure fields are defined by *U* and *p* files, respectively. Each file follows a specific formatting scheme. Dimensions of each field are specified by a vector which corresponds to a list of SI units of measure. The initial conditions of the field are specified with an *internalField* entry. Boundary conditions on each patch for each field are set with a *boundaryField* entry which lists the patches defined in the *boundary* file in the *polyMesh* directory. Each boundary condition requires various inputs which are discussed in further detail in the official User Guide [44].

Each subsequent time directory after the *0* directory contains all outputted simulation field results in a single file for each field for post-processing or for restarting the simulation. The outputted fields follow the same file format described above; boundary condition information for each field is carried in a *boundaryField* entry as in the initial condition files. In each field file, simulation data for all cells are recorded following the *internalField* entry in a sequential list based on cell number. This enables the same files to be used for both simulation restart as well as post-processing results from the simulation.

3.5 OpenFOAM Utilities

An assortment of utilities which provide various levels of functionality is included with the standard OpenFOAM package. Each utility is a separate
executable which is called within the case directory to perform certain operations. Sample dictionary files are incorporated with most utilities which allow utility settings to be specified. Pre-processing utilities cover field mapping between meshes, setting initial conditions, and other various initialization functions. Meshing utilities are included for mesh generation, mesh conversion, and mesh manipulation. Post-processing utilities cover data conversion to various third-party packages, basic field calculations, and data sampling. Some of the common utilities are discussed in this section; a complete list of utilities is published in the official User Guide [44].

### 3.5.1 Pre-processing

Most of the pre-processing utilities perform initialization of fields for various purposes. The `setFields` utility allows the user to set non-uniform initial values of fields based on bounding shapes or other parameters defined in the `setFieldsDict` file. The `createTurbulenceFields` tool creates and initializes all of the fields necessary for turbulence modeling. Another important pre-processing tool is the `mapFields` utility, which allows fields to be mapped between meshes sharing similar geometry, or from a coarser mesh to a finer mesh.

### 3.5.2 Meshing

Basic mesh generation is provided by the `blockMesh` and `snappyHexMesh` utilities. The `blockMesh` utility allows the user to generate multi-block meshes and reads a geometry definition comprised of points, blocks, and patches from the `blockMeshDict` file in the `polyMesh` directory as mentioned in Section 3.4.2. Blocks are specified by listing the points comprising each block according to the right-hand rule and defining grid spacing requirements [44]. For a typical user it is relatively difficult to generate complex meshes using `blockMesh` because the tool is based fully on text file input and has no graphical interface to visualize the
designed mesh. However, the tool performs adequately for simple meshes or for an initial mesh as a basis for trim meshing with snappyHexMesh.

The snappyHexMesh utility provides an alternative method for generation of three-dimensional body-fitted hexahedral grids. The tool is inherently three-dimensional and requires an initial mesh describing the computational domain and a surface representation of the complex body in STL format. From the initial base mesh, snappyHexMesh automatically generates a mesh conforming to the surface with optional snapping of cells to the surface and optional extrusion layer generation. The utility also allows for parallel mesh generation with a load balancing step during each iteration [44].

If the mesh has already been generated using third-party meshing software, OpenFOAM provides many utilities to convert the mesh to the OpenFOAM mesh format. Conversion utilities exist for meshes from the primary commercial CFD packages of Fluent, STAR-CD, STAR-CCM+, and CFX. Additional conversion is possible from meshes generated by standalone meshing suites such as Gambit, Pointwise and IcemCFD.

After converting a mesh into OpenFOAM format, many utilities exist for modifying, fixing, or otherwise preparing a mesh for a simulation. There are utilities to refine, renumber, or rotate the mesh, as well as create baffles or additional patches. OpenFOAM also allows the definition of cellZones and faceZones, special sets of cells or faces which are automatically decomposed for parallel processing.

### 3.5.3 Domain Decomposition

Parallel calculations require domain decomposition, which is the process of splitting the computational mesh for allocation among a number of processors. Communication between processors is accomplished through the use of a Message Passing Interface (MPI) which is a communications protocol for high-performance parallel computing. OpenFOAM is built with the open-source
OpenMPI protocol by default; however, other MPI protocols can be supported with some minor changes in the package compilation process.

The process of splitting the domain for parallel computation is performed in OpenFOAM by the decomposePar utility. Several mesh splitting algorithms are available. ‘Simple’ decomposition allows the user to define split regions by equivalent sectioning of the domain based on the Cartesian directions. ‘Metis’ and ‘Scotch’ decomposition are standardized partitioning algorithms based on the minimization of the number of interface boundaries for each processor.

The decomposePar utility is run in the case directory and generates local processor directories for every processor. Each directory contains separate time directories which hold the field values corresponding to cells allocated to that processor. During the solution process, each processor writes out the necessary fields into its local directory. After the simulation is complete, executing the reconstructPar utility in the case directory recombines all of the solution files into common time directories in the case directory.

3.5.4 Post-processing

The standard OpenFOAM installation is packaged with a post-processing utility called Paraview, an open-source data visualization package produced by Kitware and based on the Visualization Toolkit (VTK) format for data rendering and post-processing. Paraview version 3.10 includes a built-in data reader plugin for OpenFOAM data which provides an option for working directly with decomposed data, eliminating the reconstructPar step described in Section 3.5.3. For large cases a foamToVTK utility is available to convert OpenFOAM data directly to the VTK format for faster post-processing if necessary. If other post-processing tools are available OpenFOAM includes utilities to convert data to popular third-party post-processing utilities such as EnSight, Fieldview, Tecplot and Fluent.

In addition to visual post-processing, OpenFOAM also includes utilities for numerical post-processing of simulation field files. Utilities such as Mach and
streamFunction calculate quantities from the stored velocity field in each time directory and output additional field files for the calculated values. Tensor operations such as divergence, magnitude, and splitting of a field into components are available to operate on stored field files using the foamCalc utility. Lastly, solution sampling can be conducted on stored field files at defined survey locations in the domain using the sample and probeLocations utilities.

3.5.5 Run-time Processing

In addition to post-processing, OpenFOAM allows users to conduct run-time processing using function objects. Function objects are pre-compiled libraries which can be added to the controlDict file to perform various processing tasks during run-time. The range of available functionality is extensive and includes sampling of data at point locations and along survey lines, calculation of force coefficients on bodies, and generation of elements for visual post-processing, such as section planes and iso-surfaces [46]. Furthermore, writing only the data of interest requires a much smaller memory footprint than the full dataset, reducing the time required to post-process a solution as well as the overall disk space requirements for a time-accurate solution.

3.5.6 Solution Monitoring

Monitoring solution progress is important to ensuring computational accuracy and determining solution convergence. During execution of a general simulation, OpenFOAM outputs convergence residual statistics as well as local and global Courant number information for each iterative solution process. This output can be sent to a log file using the foamJob script file to initiate the solver as a background process; the log file can then be monitored in real-time with the tail UNIX command. The foamLog script file processes the log file and generates files corresponding to each monitored residual for graphical comparison.
3.5.7 Additional Community-developed Tools

As seen in the previous sections, OpenFOAM contains a great deal of functional tools already built into the existing code database. The customization offered by the open-source code has also allowed users within the community to develop their own tools and distribute them separate from the original CFD package. Two prominent examples of useful user contributions are PyFoam and swak4Foam [47]. PyFoam is a Python library that has been developed to control OpenFOAM simulations and manipulate data in a command line fashion using Python commands. It includes utilities for advanced solution monitoring, clearing and duplication of case files, and manipulating dictionary files and field definitions to quickly modify case settings. The swak4Foam (“Swiss Army Knife for FOAM”) library expands beyond the basic setFields utility by enabling the user to enter non-uniform and analytical boundary conditions and field values using parsed complex mathematical expressions, including variables and conditional statements. Both of these tools are used by a significant portion of the OpenFOAM community to support analysis and execution of simulations and have been used in this thesis to assist with setup and monitoring tasks for the simulations in Chapter 5. The simulation monitoring and input file modification capabilities provided by these tools are necessary to ensure efficiency using OpenFOAM in an engineering production environment.
CHAPTER 4  
SOLUTION METHODOLOGY

In this chapter, an overview of the methodology used in developing the Vorticity Confinement solver in OpenFOAM is presented. In actuality two separate solvers have been developed in the current work to stay true to the component-based approach used in OpenFOAM; pisoVcFoam is a time-accurate, implicit time-stepping solver based on the PISO (Pressure-Implicit with Splitting of Operators) algorithm, while simpleVcFoam is a steady-state iterative solver based on the SIMPLE (Semi Implicit Method for Pressure Linked Equations) algorithm. Both numerical algorithms have been standardized in most of the pressure-based OpenFOAM solvers included in the CFD package; the inclusion of the VC1 and VC2 methodologies within the PISO and SIMPLE frameworks, however, is a main development of this work.

The finite volume formulation of the Navier-Stokes equations and its treatment in OpenFOAM is described in Section 4.1. Because OpenFOAM uses a collocated (non-staggered) mesh arrangement, a typical Rhie-Chow correction is needed to couple the velocity and pressure solutions with the PISO and SIMPLE operator-splitting algorithms. OpenFOAM does not perform this correction explicitly, but instead uses a similar process to correct the solution which can be described as “in the spirit of Rhie-Chow” [48]. The Rhie-Chow implementation and the details of the PISO and SIMPLE algorithms are presented in Section 4.2. The implementation of the Vorticity Confinement model is similar for both solvers and is discussed in Section 4.3. Additionally, the immersed boundary treatment of no-slip boundaries within the flow field is presented in Section 4.3.2.
There exist three typical discretization approaches for the numerical solution of PDEs: finite difference, finite element, and finite volume. Each method requires solving over a grid of points defining the domain of interest; the differences between approaches arise in the solution methodology. Short descriptions of the finite difference and finite element methodologies are given here in order to highlight the benefits of the finite volume approach.

The finite difference approach solves the conservation equations in differential form by approximating the partial derivatives at each grid point in terms of the surrounding nodal values, creating an algebraic equation at each node [1]. Its main advantage is the ease in obtaining higher-order approximations using Taylor series expansions; however, conservation is not explicitly enforced with finite difference equations and requires special treatment to ensure a conservative solution [49].

The finite element approach is similar to finite volume in that it is based on the integral formulation of the conservation equations; however, with finite element methods the conservation equations are multiplied by a weighting function based on a linear shape function within each element to ensure continuity between elements. The equations are then solved by requiring that derivatives at each nodal value equal zero, which results in a solution based on minimization of the shape function residuals [1].

To contrast with the aforementioned solution methodologies, the discretization approach used within OpenFOAM and the current work is the finite volume method, which is based on the integral form of the conservation equations. The solution domain is subdivided into control volumes with the solution points defined at the centroids of each control volume. Values from the centroids are interpolated to the surfaces (faces) of the control volumes. Surface and volume integrals can be evaluated from the values on the faces and an algebraic equation can be obtained for each control volume, which also has the benefit of ensuring explicit conservation [1]. The integral form of the conservation
equations also allows for the presence of discontinuities, such as shocks, within control volumes, whereas the differential form used for finite difference methods requires special treatment for discontinuities because it assumes that flow properties are continuous [50]. The main disadvantage of finite volume methods is the difficulty extending the schemes to higher than second order due to the interpolation requirements for control volume surface values [1]. For most engineering calculations, however, second-order accuracy is sufficient to provide an accurate result and higher order accuracy becomes illusory in the presence of thin vortical structures or turbulence.

4.1.1 Conservation Equations in Integral Form

The integral conservation laws governing an isothermal, constant density, incompressible fluid for a control volume fixed in space can be written as follows:

\[
\oint_S \mathbf{U} \cdot n \, dS = 0 \tag{4.1}
\]

\[
\frac{\partial}{\partial t} \int_V \mathbf{U} \, dV + \oint_S \mathbf{U} \cdot n \, dS = \oint_S \mathbf{\sigma} \cdot n \, dS + \int_V \mathbf{f} \, dV \tag{4.2}
\]

where \( \mathbf{U} \) is the velocity vector and \( \mathbf{f} \) includes body and gravitational forces. The stress tensor, \( \mathbf{\sigma} \), includes pressure and viscous stresses and is given as follows:

\[
\mathbf{\sigma} = -p I + \nu (\nabla \mathbf{U} + (\nabla \mathbf{U})^T) \tag{4.3}
\]

Equation (4.2) is a mix of surface and volume integrals which makes it somewhat difficult to evaluate numerically. To manipulate the surface integrals, the generalized form of Gauss' theorem produces the following useful identities [48]:

34
\[
\int_v \nabla \cdot \mathbf{a} \, dV = \oint_S \mathbf{a} \cdot \mathbf{n} \, dS 
\]  \hspace{1cm} (4.4)

\[
\int_v \nabla \phi \, dV = \oint_S \phi \, dS 
\]  \hspace{1cm} (4.5)

\[
\int_v \nabla \mathbf{a} \, dV = \oint_S \mathbf{a} \, dS 
\]  \hspace{1cm} (4.6)

\( S \) represents the bounding surface of the control volume, where \( dS \) is a surface element on \( S \) with a surface normal \( \mathbf{n} \) directed outwards from the control volume. Tensor quantities are represented by \( \mathbf{a} \), while scalar quantities are described by \( \phi \). Using the identities in Equations (4.4)-(4.6), the surface integrals in Equations (4.1) and (4.2) can be converted to volume integrals to be evaluated across the control volumes and the conservation equations can be written symbolically in differential form:

\[
\nabla \cdot \mathbf{U} = 0 
\]  \hspace{1cm} (4.7)

\[
\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{UU}) = -\nabla p + \nabla \cdot (\nu \nabla \mathbf{U}) + \mathbf{F} 
\]  \hspace{1cm} (4.8)

To simplify the discussion, the terms in Equation (4.8) can be arranged into Equation (4.9) according to generic discretization treatment within the finite volume approach. Terms requiring explicit evaluation such as the contribution of pressure and external sources are placed on the right hand side of the equation. Terms on the left hand side of the equation can be treated using implicit evaluation which allows for a more stable calculation. The terms in the equation are, from left to right: the temporal derivative, convection term, diffusion term, contribution of the pressure, and the explicit source term.

\[
\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{UU}) - \nabla \cdot (\nu \nabla \mathbf{U}) = -\nabla p + \mathbf{F} 
\]  \hspace{1cm} (4.9)
In the case studies presented in Chapter 5, the temporal derivative is discretized using either an implicit backward Euler time scheme or an implicit Crank-Nicholson method, both of which are second-order accurate in time. A full explanation of the backward Euler and Crank-Nicholson time scheme implementations for OpenFOAM, and hence the finite volume approach, is provided in Hrvoje Jasak’s thesis [48] and is not discussed here further.

An explanation of the spatial term discretization is necessary to understand the implementation of the Rhie-Chow correction, discussed in Section 4.2.3, which is necessary for accurate pressure-velocity coupling on the OpenFOAM collocated mesh structure. A discretized form of Equation (4.4) is derived by assuming linear (second-order) variation across the control volume and assuming that the faces of the control volume are flat surfaces based on the mesh quality criteria detailed in Section 3.3 [48]:

\[(\nabla \cdot \mathbf{a})V_p = \sum_f S \cdot a_f\] (4.10)

In this discretized form, \(V_p\) represents the volume of a discrete cell with centroid point \(P\). The \(f\) label denotes a face value of the quantity of interest, while \(S\) represents the discrete surface area of the face. Applying this form of Gauss’ theorem in Equation (4.10) to the convection term recovers the following:

\[
\int_{V_p} \nabla \cdot (\mathbf{uu}) \, dV = \sum_f S \cdot (\mathbf{uu})_f \\
= \sum_f S \cdot (\mathbf{u})_f \mathbf{u}_f \\
= \sum_f F \mathbf{u}_f \] (4.11)
In the final manipulation of Equation (4.11), \( F \) represents the mass flux through each face, and \( \mathbf{U}_f \) is the velocity at the face interpolated from the control volume centroid. The flux must satisfy continuity in Equation (4.7) which creates a non-linear relationship. To linearize the convection term, the flux contribution (and resulting non-linearity) is lagged and uses the existing velocity field to calculate the mass flux. This requires iteration over the non-linear terms to accurately resolve the non-linear effects which will be discussed in Section 4.2.

The diffusion term is discretized in a similar manner to the convection term. Applying Equation (4.10) to the diffusion term and assuming a linear variation across the control volume results in the following:

\[
\int_{V_p} \nabla \cdot (\nu \nabla \mathbf{U}) \, dV = \sum_f S \cdot (\nabla \mathbf{U})_f \\
= \sum_f \nu_f S \cdot (\nabla \mathbf{U})_f
\]  

(4.12)

The gradient of velocity at each face is interpolated based on the values of velocity at the two unique cell centers on either side of each face. The most accurate interpolation occurs when the mesh is orthogonal and the face normal vector is parallel to the vector between cell centroids; if the mesh is non-orthogonal, correction factors are necessary to maintain sufficient accuracy or the solution may become unbounded.

4.2 Pressure-Correction Methods

Pressure-correction methods form the basis for the solution of the Navier-Stokes equations in OpenFOAM for incompressible flow. Pressure-correction methods were first introduced by Chorin [51] for incompressible flow regimes and are a subset of projection methods using the operator-splitting approach. The solution is advanced by a predictor step in which the momentum equation is solved for a divergent intermediate velocity field. The algorithms rely on the
assumption that the intermediate known velocity field can be decomposed into unknown divergence-free (solenoidal) and irrotational parts [52]:

\[ q = q_{\text{div}} + \nabla \varphi \]  

(4.13)

The irrotational part can also be represented by the gradient of a potential shown in (4.13). Enforcing continuity requires the divergence-free contribution equal zero by definition and recovers a Poisson equation for the unknown potential based on the known velocity field.

\[ \nabla \cdot q = \nabla^2 \varphi \]  

(4.14)

After solving for the potential, the new divergence-free velocity field is calculated by rearranging (4.13) and correcting the previous velocity field with the known potential to advance the solution to the next time step or iteration:

\[ q_{\text{div}} = q - \nabla \varphi \]  

(4.15)

Solving for the velocity field using a pressure-correction method requires assembling a system of equations at each solution point in the domain. For each cell, all temporal and spatial discretization operations in Equation (4.9) can be condensed into the following algebraic equation [48]:

\[ a_p u_p^n + \sum_N a_N u_N^n = r_p \]  

(4.16)

To solve Equation (4.16) at the current time level \( n \) for the value at the cell centroid \( P \) requires the values of the neighboring \( N \) cells at time level \( n \) and the explicit source term, \( r_p \). Each control volume is associated with a coefficient \( a_p \) and neighboring control volume coefficients \( a_N \). The solution requires the
assembly of the entire system of equations for all control volumes in block-matrix form:

$$[A][U] = [r] \quad (4.17)$$

In this form, the $A$ matrix is a sparse matrix constructed of diagonal coefficients $a_p$ and off-diagonal coefficients $a_N$, the $U$ vector matrix has one element per control volume, and the $r$ vector matrix holds the source term for each control volume. Using this terminology enables tensor operations spanning the entire mesh to be written with symbolic equations. The pressure-correction approach requires the derivation of an explicit pressure equation which begins with the solution of a predictor step written in Equation (4.18):

$$a_p U_p^n + \sum_N a_N U_N^n - \frac{U^o}{\Delta t} = F_p - \nabla p \quad (4.18)$$

In this step, the old and new time levels are designated by $o$ and $n$, respectively. Depending on the selected temporal discretization there could potentially be additional time levels; however, for derivation purposes only one is shown here. The $a$ coefficients encapsulate necessary dimensional details such as the cell volume and time step. To express the equation as a solution for $U_P$, an additional term $H(U)$ is defined which holds the off-diagonal contributions as well as any additional source terms [48]:

$$H(U) = -\sum_N a_N U_N^n + \frac{U^o}{\Delta t} + F_p \quad (4.19)$$

$$U_P^n = \frac{H(U)}{a_p} - \frac{1}{a_p} \nabla p \quad (4.20)$$
Enforcing continuity in Equation (4.7) on Equation (4.20) allows for the derivation of the pressure equation. The right hand side is transformed by applying the identity in Equation (4.10):

$$\nabla \cdot \left( \frac{H(U)}{a_p} - \frac{1}{a_p} \nabla p \right) = 0$$

$$\nabla \cdot \left( \frac{1}{a_p} \nabla p \right) = \nabla \cdot \frac{H(U)}{a_p}$$

$$\nabla \cdot \left( \frac{1}{a_p} \nabla p \right) = \sum_f S \cdot \left( \frac{H(U)}{a_p} \right)_f$$  \hspace{1cm} (4.21)

The final discretized system of equations requires the solution of the Poisson system in Equation (4.21) in a similar manner to the diffusion term in Equation (4.12):

$$\sum_f \left( \frac{1}{a_p} \right)_f S \cdot (\nabla p)_f = \sum_f \left( \frac{1}{a_p} \right)_f S \cdot (H(U))_f$$  \hspace{1cm} (4.22)

To calculate the velocity for the new time step, Equation (4.20) is transformed by applying the identity in Equation (4.5) to the pressure term, which allows the use of the solved pressure from Equation (4.22) on the faces:

$$u_p^n = \frac{H(U)}{a_p} - \frac{1}{a_p} \sum_f S(p)_f$$  \hspace{1cm} (4.23)

This completes the iteration and the calculation can either continue iterating or move on to the next time level, depending on the convergence criterion. The following sections describe the time-accurate and steady-state pressure-correction algorithms used in OpenFOAM.
4.2.1 PISO Algorithm for Time-Accurate Simulation

The PISO (Pressure-Implicit with Splitting of Operators) algorithm is an implicit time-marching scheme first developed by Issa which uses operator splitting to separate the solution of the velocity and pressure equations, such that the solutions of each field are close approximations of the exact solution with formal second-order temporal accuracy [53]. The solution procedure is as follows:

1. Solve the discretized momentum equation implicitly for the predictor velocity using the current values of velocity and the pressure. For the first iteration the pressure from the previous time step is used.
2. Compute the mass fluxes based on the predictor velocity.
3. Assemble and solve the pressure-correction equation for the corrective pressure using the predictor velocity field.
4. Correct the mass fluxes to be divergence-free. This divergence-free field is used in the discretized convection term in the next time step.
5. Correct the predictor velocity explicitly to obtain a divergence-free velocity field.
6. Repeat steps 2-5 using the new velocity field as a predictor velocity for a set number of iterations. Issa recommends a minimum of two predictor-corrector iterations to achieve an accurate solution [53].
7. Update boundary conditions, increase the time step and repeat from step 1.

A flow chart representation of the PISO algorithm compared with the SIMPLE algorithm (discussed in Section 4.2.2) is shown in Figure 5. The main differences between SIMPLE and PISO are the lack of relaxation and the additional iteration on the pressure equation in PISO. Issa [53] provides evidence that the errors associated with the method due to operator splitting are small enough that the algorithm does not require iterative solution of the momentum equation.
Figure 5. Flow chart comparison of PISO and SIMPLE algorithms.
Favorable results for both time-accurate incompressible and subsonic compressible flows were presented by Issa et al. [54] in which it was demonstrated that PISO is also stable for larger time steps approaching pseudo-steady simulation.

4.2.2 SIMPLE Algorithm for Steady-State Simulation

The SIMPLE (Semi Implicit Method for Pressure Linked Equations) algorithm was introduced by Patankar [55] for incompressible flow simulation. While it is capable of solving time-accurate flow problems, other methods (such as PISO) are more efficient and as a result the SIMPLE algorithm has popularity primarily with steady-state simulation. The solution procedure in OpenFOAM is as follows:

1. Solve the discretized momentum equation with a relaxed velocity field and the pressure field from the previous time step to compute an intermediate velocity field.
2. Compute the mass fluxes based on the intermediate velocity.
3. Solve the pressure equation and apply under-relaxation to the pressure field.
4. Correct the mass fluxes to be divergence-free. This field is used in the discretized convection term in the next time step.
5. Correct the velocity field using the new pressure field.
6. Update the boundary conditions and repeat with step 1 until convergence.

On a per-iteration basis the SIMPLE algorithm is faster than PISO because it is solving the pressure equation only once; however, in a time-accurate calculation the SIMPLE algorithm must apply several iterations per time step to obtain an accurate solution. The increased iterations result in a longer calculation time per time step compared to the PISO algorithm.
4.2.3 Rhie-Chow Treatment in OpenFOAM

In the past, the use of pressure-correction algorithms on collocated meshes has exhibited decoupling of the velocity and pressure fields and subsequent oscillations within the solution. Normal central differencing on collocated grids causes a ‘checkerboard’ effect in which alternate grid points are skipped in the differencing operation which leads to the oscillations [49]. Rhie and Chow [56] proposed a collocated grid arrangement which counters this effect. In their formulation, the momentum equation is solved for velocity as in Step 1 of the PISO and SIMPLE algorithms. An intermediate velocity from which the pressure contribution is explicitly removed is then interpolated from the cell nodes to the cell faces. To find the corrected velocity at the faces, the pressure gradient is then added back to the interpolated velocity; however, the important point is the pressure gradient is evaluated at the faces of each cell rather than calculated from opposing cell nodes. This correction is effectively a higher-order pressure correction which eliminates the oscillations and satisfies continuity. OpenFOAM does not have such an explicit correction factor, but as previously stated employs a methodology described as “in the spirit of Rhie-Chow” [48].

Figure 6 displays a high-level C++ code snippet which details the PISO algorithm including the implicit Rhie-Chow correction; while the code is fairly understandable purely from the syntax, each operation will be explained for clarity. Lines 1-6 initialize the coefficients of a finite volume matrix UEqn created from the temporal, convection, and diffusion terms. The momentum equation predictor using the UEqn object along with the gradient of the current pressure field is solved for an intermediate velocity in line 8; the for loop in lines 9-24 is the iterative pressure corrector step. Two functions UEqn.A() and UEqn.H() are called on lines 11-12, which are equivalent to the \( a_p \) and \( H(\mathbf{U}) \) terms in Equation (4.20). Line 12 is the explicit removal of the pressure from the momentum predictor for the Rhie-Chow correction; the \( H(\mathbf{U}) \) object in Equation (4.19) does not include the pressure. Because the momentum predictor is implicit this result is different than simply solving Equation (4.18) without the pressure entirely.
Following the Rhie-Chow methodology, this velocity is interpolated to the faces in line 14; solving for the flux at each face involves a dot product between the normal mesh surface and the interpolated velocity field. Lines 16-20 describe the solution of the pressure equation which is equivalent to Equation (4.21). The mass flux is corrected on line 21. Line 22 completes the implicit Rhie-Chow correction by correcting the velocity without a pressure contribution from line 12 with the gradient of pressure based on the face values. To summarize, Rhie-Chow is implemented implicitly in OpenFOAM by removing the effect of pressure from the flux $\phi$, solving the divergence of the pressure term in Equation (4.21) using the gradient of pressure on the cell face, and correcting the predictor velocity using the pressure gradient calculated from the cell face values.

### 4.3 Vorticity Confinement Implementation

The implementation of Vorticity Confinement for the finite volume approach follows the methodology suggested for compressible flows, in which

```cpp
fVVectorMatrix UEqn
(
    fvm::ddt(U)
    + fvm::div(phi, U)
    - fvm::laplacian(nu, U)
);

solve(UEqn == -fvc::grad(p));
for (int corr=0; corr<nCorr; corr++)
{
    volScalarField rUA = 1.0/UEqn.A();
    U = rUA*UEqn.H();
    phi = fvc::interpolate(U) & mesh.Sf();
    fVScalarMatrix pEqn
    (        fvm::laplacian(rUA, p) == fvc::div(phi)
    );
    pEqn.solve();
    phi -= pEqn.flux();
    U -= rUA*fvc::grad(p);
    U.correctBoundaryConditions();
}
```

*Figure 6. OpenFOAM C++ code snippet for the PISO algorithm.*
the VC term is introduced to the integral equations as a body force source term. As stated in Chapter 2, the implementation used within this work follows the suggestion of Costes [28] in which the VC term is left out of the energy equation and added to the momentum equation alone. This treatment is appropriate because the VC term has no true physical representation and is purely a numerical treatment for vortical regions. An identical implementation approach is used for both VC1 and VC2 formulations which takes advantage of similarities between the two VC methods, with the difference occurring in the formulation of the source term as outlined in Section 2.1 and 2.2.

The similarities between the PISO and SIMPLE algorithm flowcharts displayed in Figure 5 suggest that a general Vorticity Confinement approach for pressure-correction methods can be developed. One concern with these methodologies, however, is the treatment of non-linear source terms. Equation (4.19) indicates that the source term is placed in the \( H(\mathbf{U}) \) operator, which is acceptable for implicit source terms because the \( H(\mathbf{U}) \) operator is reevaluated based on the current velocity field at the beginning of each pressure correction step. For explicit source terms, however, any non-linear behavior in the source term is lagged and is calculated using the previous time step velocity field when the term is initialized. This treatment is somewhat problematic for Vorticity Confinement; the intention of the additional term is to provide anti-diffusion in vortical regions and in the original formulation by Steinhoff [2] the term is calculated using the current time step velocity field. Lagging the VC term results in anti-diffusion based on the previous time step which may not be optimal for the calculation.

To address this issue a novel formulation has been developed for pressure-correction methods which allows the VC term to be calculated based on the velocity field originating from the momentum predictor step rather than the previous time step velocity. This approach is in the spirit of the original VC approach which uses a fractional-step solution methodology where each operator is solved sequentially. The algorithm begins by solving for the momentum
predictor without the VC term using Equation (4.18) as in the normal pressure-correction method:

\[ a_p U^n_p + \sum_{N} a_N U^n_N - \frac{U^0}{\Delta t} = F_p - \nabla p \]

The off-diagonal terms and source terms, with the exception of the pressure gradient, are collected into the \( \mathbf{H}(\mathbf{U}) \) operator as shown in Equation (4.19). The VC term is then introduced in a modified version of Equation (4.20):

\[ U^n_p = \frac{\mathbf{H}(\mathbf{U})}{a_p} - \frac{1}{a_p} \nabla p + \frac{\varepsilon S}{a_p} \quad (4.24) \]

The VC term is calculated before Equation (4.24) based on the resulting intermediate velocity from the momentum predictor solution of \( U_x, U_y, \) and \( U_z \). Because the VC term is a numerical correction rather than a physical term, inserting it at this point in the solution process causes no ill effects to solution stability. Effectively the VC term is correcting the momentum predictor before proceeding to the pressure corrector. The solution algorithm progresses through the steps outlined in Equations (4.21) through (4.23), with the VC term joining with the \( \mathbf{H}(\mathbf{U}) \) operator term in the equations. In this manner the VC correction is calculated based on the current time step velocities and avoids the lagging behavior of the explicit source term. Results presented in Chapter 5 demonstrate the viability of the implementation.

4.3.1 Parallelization

Parallel computation is critical for present-day CFD engineering applications. The size and scope of industrial problems necessitates using millions of cells to resolve the flow to an acceptable level of accuracy, which in turn requires solving on multiple processors to obtain a solution in a reasonable amount of physical time. To address this requirement, one of the goals of the
current work mentioned in the introduction is the development of a fully parallelized solver.

Parallelization is implemented in OpenFOAM from lower-level classes, which implies that most user-developed applications require no specific coding to realize parallel operation. Most of the operations in the calculation of the VC source terms involve manipulation of fields spanning the entire mesh using tensor mathematics. These operations are inherently parallel from the lower-level classes. Manipulation of fields on a cell-by-cell basis, however, requires specific coding to ensure proper communication between processor domains. In the VC2 formulation, the harmonic mean of vorticity is used which must be calculated over each cell. An efficient algorithm has been developed to calculate the harmonic mean in Equation (2.8) and is detailed in Figure 7. Lines 1-4 create the fields used in the calculation; the SMALL constant is used here to prevent divide by zero errors. The forAll loop on line 5 orders all processors to iterate over the cells in

```cpp
volScalarField reciprocal = 1 / (magVorticity + SMALL);
volVectorField VC2_wn = vorticity;
scalarField sum(mesh.nCells(), SMALL);
scalarField count(mesh.nCells(), 0);
forAll(mesh.C(), celli) {
    const labelList& neighbors = mesh.cellCells()[celli];
    forAll(neighbors, n) {
        sum[celli] += reciprocal[neighbors[n]];
    }
    sum[celli] += reciprocal[celli];
    count[celli] += neighbors.size() + 1;
}
forAll(mesh.boundaryMesh(), patchi) {
    const fvPatchScalarField& pf = reciprocal.boundaryField()[patchi];
    const unallocLabelList& faceCells = pf.patch().faceCells();
    if (pf.coupled()) {
        scalarField neighborReciprocal = pf.patchNeighbourField();
        forAll(faceCells, facei) {
            sum[faceCells[facei]] += neighborReciprocal[facei];
            count[faceCells[facei]]++;
        }
    }
}
VC2_wn.internalField() == count / sum;
```

Figure 7. C++ code snippet for harmonic mean algorithm.
each decomposed portion of the mesh; a list of neighboring cells is created on line 6 and is iterated in lines 7-9 for a summation of the reciprocal term. The contribution of the current cell is included in lines 10-11.

The forAll loop in lines 13-23 encompasses the portion of the harmonic mean which requires inter-processor communication. In the decomposition process, the processor boundaries are marked with a particular type of boundary condition. Extracting the coupled boundaries on line 16 allows for values from the neighboring processor to be retrieved using the patchNeighbourField() method. The harmonic mean on processor boundary cells requires these values for proper calculation. On line 24, the count and sum fields, which have been holding the number of cells in each stencil and the summation of the reciprocal of vorticity over the stencil, can be multiplied with the vorticity as a field operation to create the $W$ term in Equation (2.7) for the VC2 formulation. This completes the parallel algorithm for computing the harmonic mean.

4.3.2 VC as an Immersed Boundary Method

Immersed boundary methods are alternative methods for viscous flow simulation in which boundaries are embedded within non-conforming grids, as opposed to conventional solution approaches using body-fitted grids. The basic approach involves modifying the equations for cells in the vicinity of the immersed boundary to take into account the desired boundary conditions; in this manner, a generic grid of specific resolution can be used for different geometry configurations without requiring repeated grid generation. Mittal and Iaccarino [57] provide a full overview of conventional immersed boundary methods and discuss both advantages and disadvantages of the various methodologies. Conventional methods involve two different formulations, namely the continuous forcing approach and the discrete forcing approach. The continuous forcing approach involving the application of an analytical forcing function was first suggested by Peskin [58] for dynamic heart simulation and has since been extended to rigid body simulation [59]. Discrete forcing methods are the focus of
recent research and involve extracting a discrete forcing function directly from the numerical solution of surrounding cells. These methods include the indirect forcing approach of Mohd-Yosuf [60], the ghost-cell approach [61,62,63], and the cut-cell approach [64].

In the current work, an immersed boundary method is implemented similar to that proposed by Fan and Steinhoff [6] and employed by Lynn [24]. The previous formulations utilize a level set function which is essentially a signed distance field marking cells inside and outside the body. During the calculation, the solution algorithm proceeds through all cells and velocities in interior cells are simply set to zero. The current effort uses a customized pre-processing tool specifically developed to mark ‘inside’ and ‘outside’ cells as separate sets. The set identification for each cell is determined by comparing its location to the surface normal of a watertight CAD surface embedded within a background volume mesh; after all cells have been separated between the two sets, the sets can be converted to ‘zones’ which are automatically decomposed during a parallel computation. As in the aforementioned formulations, the velocity in interior cells is set to zero during the calculation to generate the immersed boundary. This process is rather crude numerically and without VC provides only a first-order approximation of the boundary with considerably higher discretization error and ‘staircase’ effects along the immersed boundary interface. Applying the unique VC formulation compresses the vorticity towards the immersed boundary, containing the discretization error within the boundary layer, and diffuses vorticity along the boundary, smoothing any ‘staircase’ behavior along the immersed boundary interface allowing a sharp boundary definition to emerge. In Chapter 5, the immersed boundary method is verified in 5.2 with an immersed slanted plate simulation and shown to be suitable for complex geometry simulation in 5.5 with the time-accurate simulation of an immersed Formula One racecar at yaw.
CHAPTER 5
RESULTS AND DISCUSSION

This chapter presents an overview of the results of three two-dimensional verification cases as well as two applications demonstrating complex three-dimensional simulation. The primary focus of these simulations is to establish and verify the implementation of Vorticity Confinement presented within this thesis and quantify the effects of the VC methodology on the resulting flow field. According to the AIAA Standards Guide for the Verification and Validation of Computational Fluid Dynamics Simulations [65], verification is defined as the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model. To demonstrate verification of the developed VC algorithm, results from the verification cases are shown with and without VC and the effects of the method on the resulting flow field are discussed in context with results shown by Steinhoff and others in similar studies [4,15,17,18].

Each of the presented case studies utilizes either of the developed solvers, pisoVcFoam (time-accurate solver) or simpleVcFoam (steady-state solver), depending on the nature of the temporal approximation. Discretization of the governing equations follows standard Gaussian integration described in Chapter 4, with the cell-to-face interpolation scheme determining the order of accuracy of the solution. Both solvers use a second-order upwind scheme for interpolation of the convection term and a second-order central differencing scheme for interpolation of the diffusion term. Two separate discretization schemes are used for the temporal term in the pisoVcFoam solver. For cases in 5.3 and 5.5, an implicit second-order Crank Nicholson scheme is used which takes the mean of the \( n-1 \) and \( n \) solution values. This scheme requires only two time levels and is unconditionally stable but can become unbounded and produce a false solution if the time step is too large. An implicit second-order
backward differencing scheme is used in the 5.4 case study to match temporal discretization used in the large-eddy simulation (LES) study provided for comparison. This scheme uses three time levels (n, n-1, and n-2 solution values) and ensures boundedness, but requires significantly more memory overhead.

5.1 Flow over a Flat Plate

This verification case involves flow traveling over a two-dimensional grid-aligned flat plate with a no-slip boundary condition. A boundary layer is generated due to the no-slip wall condition and the thickness can increase in height as the flow progresses downstream along the plate. Applying Vorticity Confinement as a boundary layer model has shown in previous studies [7,24] that with increasing values of $\varepsilon$, VC reduces the boundary layer thickness to 2-3 grid cells by providing a compressive effect on the vorticity in the boundary normal direction. This simple case provides an opportunity to calibrate the confinement strength parameter to an appropriate value which can be used as a starting point for future cases.

5.1.1 Setup Details

The two-dimensional computational mesh used in this verification case is shown in Figure 8. The domain is 2.5 m x 2.5 m and is discretized by a 0.01 m uniform mesh resulting in 62,500 cells. Although a problem of this size is easily solvable using a serial process, the mesh has been decomposed into four equal subdomains using simple geometric decomposition to demonstrate the parallel capability of the developed VC solver. The flow is initialized with a uniform velocity of (1, 0) where the x-axis is parallel to the plate and the y-axis is normal to the plate; flow moves from left to right. The applied boundary conditions are shown in Figure 9. A slip wall is used for the first 0.5 m away from the uniform inlet to avoid spurious boundary interactions. A no-slip wall continues for 2 m to the domain outlet. A constant pressure boundary condition is applied at the outlet and Neumann boundary conditions are used at all other boundaries for pressure.
Figure 8. Computational mesh for flat plate case study. Processor subdomains have been separately colored to show domain decomposition.

Figure 9. Boundary conditions for flat plate case study.
For the velocity field, a prescribed uniform velocity is used at the inlet boundary and a freestream boundary condition with flow parallel to the inlet velocity is set for the far field boundary. The steady-state solver simpleVcFoam is used because only the steady-state nature of the boundary is of interest for this case. Isotropic numerical diffusion is explicitly applied using a value of $\nu = 0.001 \text{ m}^2/\text{s}$. It is necessary with VC to impose isotropic diffusion on the vortical features of the flow which can then be counteracted by the isotropic compression imposed with the calibrated constant confinement strength [66]. Using the aforementioned settings, a baseline case without VC has been developed and is compared against three VC1 cases and three VC2 cases. The VC1 cases utilize a combination of Field Confinement and Surface Confinement and apply the same confinement strength value to both formulations. The VC2 formulation does not require any special treatment near the boundary. The confinement strength is varied in each VC case; in this manner, a range of confinement parameters can be calibrated for each formulation and the effects of the VC model on the flow field can be observed.

### 5.1.2 Baseline Results

The velocity magnitude and vorticity contours for the baseline case without VC are shown in Figure 10 and Figure 11. On all contour plots the domain has been clipped with the upper half hidden; this portion of the domain is only necessary to prevent unnatural constraints on the pressure solution and does not add value to the boundary layer solution. Based on the velocity distribution in Figure 10, the boundary layer begins developing at the no-slip wall and continues growing until it reaches the outlet on the right side of the domain. The vorticity in Figure 11 has the highest value at the initial onset of the no-slip boundary and then diffuses outward away from the wall. An effective low Reynolds number solution is achieved due to the high numerical diffusion.
Figure 10. Velocity magnitude contour for flat plate case without VC.

Figure 11. Vorticity magnitude contour for flat plate case without VC.
Figure 12. Boundary layer profile of flat plate case without VC at various locations along the plate. (x) refers to the distance from the leading edge.

A plot of the boundary layer profile in Figure 12 at various distances from the leading edge of the plate shows the development of the boundary layer along the plate. The streamwise velocity (x-axis) has been normalized by the local freestream value at each station along the plate, and distance from the wall (y-axis) has been normalized by the uniform grid cell size. Based on this correlation, at the 1.875 m station the height of the boundary layer has reached approximately 20 cells. Without a turbulence model the boundary layer assumes a laminar boundary layer shape, growing much faster and becoming much thicker than a turbulent boundary layer.

5.1.3 Effect of VC on Boundary Layer

A typical near-wall boundary layer solution for conventional CFD uses wall functions which prescribe a velocity distribution based on experimental and analytical correlations; however, being based on average quantities this type of
model has difficulties predicting separation and other turbulent phenomena. Vorticity Confinement provides a means to overcome this difficulty by providing a framework in which the boundary layer relaxes to a defined shape but is still allowed to react to separation-inducing effects. VC compresses the boundary layer to the height of 2-3 grid cells and relaxes to a thin state. The solution within the boundary layer region is assumed to be unimportant to the macroscopic flow field; the localized effect of keeping the boundary layer thin simulates the shape of the turbulent boundary layer.

Results for the boundary layer modeled with varying parameters of VC1 and VC2 are shown for the 1.875 m downstream station in Figure 13 and Figure 14. In both formulations as the confinement strength is increased the resulting boundary layer height decreases, approaching the 1/7 power law relationship of a turbulent boundary layer. Increasing the confinement strength to a suitably high value confirms that the boundary layer can be reduced to the thickness of only a few grid cells, overcoming the explicit numerical diffusion as well as any numerical diffusion inherent within the solver. This boundary layer shape is held constant along the length of the plate to approximate the shape of the 1/7 power law relationship by keeping the boundary layer thin. Figure 15 and Figure 16 depict the boundary layer profiles for the VC1 and VC2 simulations with the highest established confinement strength values. For both formulations, the boundary layer relaxes to a corresponding thickness and remains at that height along the remainder of the plate. In this case there are no perturbations in the flow which disturb this behavior. In a three-dimensional turbulent environment, however, this thickness will vary based on the surrounding flow effects and can allow separation as necessary in response to an adverse pressure gradient but will revert back to the relaxed state when unperturbed.

Figure 17 and Figure 18 depict velocity magnitude and vorticity magnitude contours for both the VC1 and VC2 simulations with the highest established confinement strength values. Both results are exceedingly similar; the only difference which stands out is a slight increase in velocity just above the
Figure 13. Varying VC1 confinement strength and its effect on boundary layer height at downstream station $x = 1.875$ m.

Figure 14. Varying VC2 confinement strength and its effect on boundary layer height at downstream station $x = 1.875$ m.
Figure 15. Boundary layer profile from the flat plate simulation with VC1 confinement strength $\varepsilon=0.25$.

Figure 16. Boundary layer profile from the flat plate simulation with VC2 confinement strength $\varepsilon=0.004$. 
Figure 17. Velocity magnitude contours for VC1 (top) and VC2 (bottom) simulations with confinement strength values of 0.25 (VC1) and 0.004 (VC2). Note that a constant diffusion \((\nu)\) of 0.001 is applied.
Figure 18. Vorticity magnitude contours for VC1 (top) and VC2 (bottom) simulations with confinement strength values of 0.25 (VC1) and 0.004 (VC2). Note that a constant diffusion ($v$) of 0.001 is applied.
boundary layer in the VC2 simulation which is evident in the boundary layer profile in Figure 16. The slight acceleration is due to the momentum conserving aspects of the VC2 formulation which will be visited in more detail in 5.3. This effect is comparable to the accuracy provided by many other boundary layer models and is of little concern.

For VC1 the established stable confinement strength parameter range is between 0.01 and 0.25, while VC2 shows stable behavior between 0.001 and 0.004. Values above this range for the flat plate boundary layer result in a chaotic flow field with large-scale enhancement and excitation of vortical structures. The investigated confinement strength ranges for VC1 and VC2 are comparable to values reported in previous studies [18]. In particular, the stable values for the VC2 formulation lie within the approximate range suggested by Steinhoff et al. of $1 \nu \leq \varepsilon \leq 4 \nu$ [67]. The agreement of the confinement parameter with previous studies and the demonstrated behavior of the solver provide confidence that the method has been implemented correctly using the OpenFOAM framework.

5.2 Flow over a Slanted Plate

The case in 5.1 demonstrates the ability of Vorticity Confinement to confine the boundary layer to a region with a thickness of 2-3 grid cells. The ability to confine the vorticity into a thin sheet can be extended to produce an immersed boundary method using the VC model. Using the process described in 4.3.2 to implement the immersed boundary, a sheet of vorticity is implicitly generated at the interface between the ‘inside’ cells being set to zero velocity and the ‘outside’ cells containing the flow solution. The VC model is then able to confine this vorticity sheet back to the immersed surface (point of highest vorticity) and provide an accurate representation of surface effects without a body-fitted grid. Defining the surface irrespective of the surrounding fluid grid cells has advantages for rapid prototyping, such that the surface can be easily changed while the surrounding mesh remains unaltered. This allows many
similar simulations to be run in a short amount of time and eliminates the manual interaction required to generate a new volume mesh.

5.2.1 Setup Details

The 5.2 case study involves the simulation of an immersed flat plate at a 15° angle with the base of the domain. This case utilizes the identical mesh from the 5.1 case study as well as the steady-state solver simpleVcFoam and therefore the details of the simulation setup described in 5.1.1 apply to the 5.2 setup as well. Identical details will not be repeated but the differences between the two simulations will be highlighted. Figure 19 displays the boundary conditions used for the immersed flat plate. The immersed surface is represented by a discrete CAD surface and is shown superimposed on the background mesh; the velocity components for each cell within this body are set to zero at each step during the iterative process. The inlet flow direction is set parallel to the

![Figure 19. Boundary conditions for slanted plate case study. Immersed body is shown superimposed on the mesh.](image)
immersed surface to reduce upstream effects from the no-slip immersed boundary. Due to the angled inlet flow direction, a constant pressure boundary condition is used at the top of the domain instead of the freestream boundary used in the 5.1 case study. All other boundaries remain similar to the 5.1 setup.

5.2.2 Immersed Boundary with VC1

An initial study is performed using a simple laminar calculation (no turbulence model) without the contribution of VC. The results are displayed in Figure 20 and Figure 21 detailing the velocity and vorticity magnitude contours. The contours are shown with values for all cells, including those found within the immersed body. The blue region with zero velocity in Figure 20 is within the immersed body. Setting the velocity to zero effectively creates a no-slip wall condition at the interface between the immersed region and the flow region, generating a boundary layer which diffuses away from the immersed surface. This boundary layer also includes ‘staircase’ effects, abnormalities within the flow which are aligned to the grid cell shapes near the immersed surface and due to the step function approach of the immersed boundary method. This is slightly evident in the vorticity contour in Figure 21; however, most of the ‘staircase’ effects in this problem are smoothed by the numerical diffusion. Conventional immersed boundary methods treat this effect with a variety of ad-hoc interpolation methods which prescribe a velocity in the cells near the immersed surface. Vorticity Confinement, on the other hand, requires no special treatment and instead convects the vorticity directly back to the immersed surface, ensuring a smooth boundary shape. Velocity and vorticity magnitude contours for the VC1 formulation are found in Figure 22 and Figure 23. With the VC1 model, a smooth boundary layer is developed over the slanted plate which remains at a fixed height and is distributed over only a few grid cells at the immersed surface. The maximum value established for the VC1 confinement strength parameter in the flat plate boundary layer case is used for both field and surface confinement and proves to be sufficient to fully confine the boundary layer to a thin sheet.
Figure 20. Velocity magnitude contour of immersed plate without VC.

Figure 21. Vorticity magnitude contour of immersed plate without VC.
Figure 22. Velocity magnitude contour for immersed plate with VC1.

Figure 23. Vorticity magnitude contour for immersed plate with VC1.
5.2.3 Immersed Boundary with VC2

The velocity and vorticity magnitude contours for the VC2 formulation using the maximum established confinement strength from 5.1 are shown in Figure 24 and Figure 25, respectively. The solution exhibits boundary layer roll-up near the end of the plate, indicated by the vortices forming within the boundary layer region. Such transient behavior is the result of combining a second-order numerical scheme for the convection term with the highest confinement strength value which minimizes the numerical diffusion and allows the appearance of transient instabilities. To test this relationship, an additional case using the same VC2 confinement parameters and first-order upwind differencing is developed. The velocity magnitude contour displayed in Figure 26 illustrates that the additional numerical diffusion added by the first-order upwind differencing eliminates the transient behavior and causes the boundary layer to thicken. A reduction in the VC2 confinement strength should likewise effectively increase the numerical diffusion within the simulation and show a similar response. Figure 27 details the velocity magnitude contour of a VC2 simulation using the original second-order convection scheme with confinement strength of 0.0035 (the previous value was 0.004). The boundary layer instability no longer appears in the solution but the boundary layer remains thin as the confinement strength was only reduced by 12.5%. The sensitivity of the VC2 model to the variation in numerical schemes confirms the findings of Butsuntorn and Jameson [32] that the confinement strength value is dependent on the numerical scheme being used. Using at least a second-order scheme serves to reduce the discretization error present and minimize the effect of numerical diffusion; as a result, the variance of the confinement parameter between simulations with a higher-order scheme is expected to be much less than with lower-order schemes. The transient instabilities created at higher confinement strength values are also potentially indicative of higher Reynolds number effects. At higher Reynolds numbers the turbulent boundary layer is a thin sheet of vortices and appears similar to the images in Figure 24 and Figure 25.
Figure 24. Velocity magnitude contour for immersed plate with VC2 and confinement strength of 0.004. The boundary layer is exhibiting roll-up behavior.

Figure 25. Vorticity magnitude contour for immersed plate with VC2 and confinement strength of 0.004. The boundary layer is exhibiting roll-up behavior.
Figure 26. Velocity magnitude contour for immersed plate with VC2 and first-order upwind differencing for the convection term. The added numerical diffusion causes the boundary layer to thicken and suppresses the transient instabilities.

Figure 27. Velocity magnitude contour for immersed plate with VC2 and second-order upwind differencing for the convection term, but with a reduced confinement strength of 0.0035. Boundary layer instabilities are no longer present in the solution.
As shown in previous cases, the balance of confinement strength with numerical diffusion provides for varying effects on the flow. This can be used to an advantage to study Reynolds number effects on the solution; at higher confinement strength values, the flow becomes more chaotic which is indicative of an effective higher Reynolds number. Smaller confinement strength values result in a lower effective Reynolds number, and removing Vorticity Confinement altogether results in a laminar Navier-Stokes solution. These effects will be studied more thoroughly in 5.4.

5.3 Advection of a Compact Vortex

The previous two case studies have used a steady-state solver to solve boundary layer problems and provide an initial estimate for the confinement strength parameter. The true applicability of Vorticity Confinement to influence vortical regions, however, is through enhancement of transient aspects of the flow and thus the implicit time-stepping time-accurate solver pisoVcFoam has been developed for the remaining case studies. One of the characteristics of VC is the ability to amplify and sustain vortical features by treating them as 'solitary waves' which ride on the convecting fluid. The ‘solitary wave’ concept is similar to the ‘soliton’ approach used by condensed matter physicists, in which dispersive effects spreading a wave-packet are balanced by self-steepening terms creating a stable configuration [68]. Locally, the vortical feature modified by VC behaves as the full asymptotic solution, whereas conventional methods supply the intermediate asymptotic solution which eventually asymptotes to a constant value of zero. VC effectively creates a non-zero constant asymptote in vortical regions. To demonstrate this behavior, a single compact vortex is advected through a periodic flow domain with a fixed background velocity.

5.3.1 Setup Details

The advection of the compact vortex is presented in the domain pictured in Figure 28. A uniform 401 x 201 cell mesh is used to discretize the two-
dimensional domain which is decomposed for parallel computation over 8 processors. Domain decomposition is performed by the METIS library which is a decomposition method that minimizes the number of processor boundaries. The odd number of cells in each coordinate direction is necessary for the vortex center to be defined at exactly (0, 0) within a single cell; otherwise, four equal point vortices are created around the (0, 0) point which behave as four compact vortices instead of a single vortex. The compact vortex is imposed on top of a background velocity of 0.1 m/s which convects the vortex through the domain. The domain is bounded by symmetry planes in the spanwise direction and periodic boundaries in the streamwise direction which allow the vortex to pass from one side of the domain to the other during the calculation. Explicit diffusion of $\nu = 0.001 \text{ m}^2/\text{s}$ is added for the VC model. Each time-accurate simulation is run for 40 seconds with a streamwise CFL number of 0.1; based on the background velocity this should result in a full transit of the vortex through the domain (4 m). Three cases are studied with explicit numerical diffusion of 0.001 m$^2$/s: a baseline case without VC, and two cases with the VC1 and VC2 formulations, respectively. Due to the low background velocity relative to the vortex strength, the vortex simulations provide an opportunity to examine the conservative aspects of both the VC1 and VC2 formulations.

![Figure 28. Boundary conditions for compact vortex advection case study.](image)
5.3.2 Numerical Stability Requirements

This case study is the first time-accurate simulation examined in this work and provides an opportunity to investigate the stability of the Vorticity Confinement algorithm with a time-accurate solver and determine whether the calibrated confinement strength coefficients from the steady-state simulations are applicable. The gradient discretization schemes used in the steady-state simulations for gradient calculations were found to be unstable during initial tests of the vortex simulation. To ensure stability, cell limiting was enabled for the gradient computation in all VC terms as well as the convection term. Cell limiting limits the extrapolated face values for the gradient computation based on the maximum and minimum cell and neighbor values, which helps prevent numerical instability by adding a slight amount of local dissipation. Using cell limiting with the VC1 scheme proved to be sufficient to maintain the highest calibrated confinement strength of 0.25 without observing any oscillations in the solution.

For the VC2 confinement term harmonic differencing (additional to harmonic mean) is also necessary for stability during time-accurate simulation. This differencing scheme interpolates the inverted vorticity field from the cell centers to the faces for gradient computation and is weighted by the magnitude of vorticity at each cell, minimizing the gradient calculation when neighboring values are small. The operation performs a stabilizing function similar to the method suggested by Steinhoff and Lynn [7] in which VC2 terms exhibiting alternating signs with neighboring values are set to zero. With the harmonic differencing scheme the VC2 confinement strength was also reduced to 0.003 (3ν) to obtain a stable vortex within the calculation. At higher confinement strength values the flow becomes perturbed near the vortex and results in a fully chaotic flow field with large scale vortices appearing in the solution. This behavior is not unexpected; due to its conservative properties the VC2 formulation conserves angular momentum and generates higher velocities near the center of the vortex with higher confinement strength values. Higher velocities tend to cause shearing of the axisymmetric vortex structure. The effect becomes strong
enough at the highest confinement strength to cause vortex breakdown, which may also be indicative of higher Reynolds number flow behavior [69].

5.3.3 Simulation Results

The streamwise vorticity magnitude across the centerline of the domain is presented in a one-dimensional plot for all three vortex cases at 1 s duration (100 time steps) in Figure 29. Due to the lack of a spanwise background velocity component the vortex does not deviate from the centerline of the domain during the calculation. In the baseline case the vortex dissipates and spreads as it is advected by the background flow; the vorticity magnitude has decreased one order of magnitude after only one second. Using either the VC1 or VC2 formulation, however, results in a much higher vorticity magnitude with the vortex focused around a region 2-3 grid cells thick. Figure 30 displays the corresponding vorticity magnitude contour plot at one second duration. The scale for the baseline case is reduced to highlight the spreading of the vortex.

![Figure 29. Centerline vorticity magnitude after 1 s (100 time steps). Without VC the vortex strength has decreased an order of magnitude.](image)
Figure 30. Contour plots of vorticity magnitude at one second simulation duration (100 time steps). Note reduction of scale for simulation without VC.
Figure 31 displays the streamwise centerline vorticity plot for all three cases at 10 seconds duration (1000 time steps). In the baseline case the vortex has effectively disappeared within the background flow, while both VC1 and VC2 prevent the vortex from spreading beyond a 2-3 grid cell region spanning the centroid of the vortex. By 10 seconds it has also become evident that the VC1 solution has begun to lag both the VC2 and no-VC solutions. Steinhoff et al. [15] have previously shown in analytical form that VC1 conserves total amplitude but not the position of the centroid, while VC2 conserves the position of the centroid. As a result, the position of a vortex will deviate from the correct value based on the original governing equations when using the VC1 formulation over a long period of time in a slow moving background flow. The VC2 formulation, however, is able to maintain the position of the centroid and as a result does not exhibit the lagging behavior. Figure 32 displays the contour plots of vorticity magnitude at the 10 second duration corresponding to Figure 31.

![Figure 31. Centerline vorticity magnitude after 10 s (1000 time steps).](image-url)
Besides the change in position, the vorticity magnitude contours for both cases are not significantly different from the previous contours at one second in Figure 30. The VC model effectively behaves as a non-zero full asymptotic solution within the local region of the vortex and does not allow further dissipation. The contour for the baseline case, however, is not even shown because the vortex has dissipated fully such that it essentially blends with the background flow. Without VC the intermediate asymptotic solution of the vortex tends towards the asymptotic solution of zero vortical flow within the constant background solution.

The streamwise centerline vorticity plot at 40 seconds duration (4000 time steps) is shown in Figure 33 along with the corresponding vorticity magnitude contours in Figure 34. Based on the background velocity the vortex should make a full transit through the periodic boundary back to the initial starting point after 40 seconds.
Figure 33. Centerline vorticity magnitude after 40 s (4000 time steps).

Figure 34. Vorticity magnitude contour after 40 s (4000 time steps). Low magnitude vorticity concentrations are visible in VC1 solution.
As expected, the VC1 solution deviates considerably from the unperturbed location of the vortex centroid, achieving only approximately 80% of the necessary travel distance. The VC2 solution, on the other hand, achieves approximately 97.5% of the anticipated travel distance. Low-magnitude asymmetric oscillations are evident in the VC2 solution near the vortex centroid; it is surmised that over the 40 second duration the asymmetries are sufficient to perturb the centroid position by the marginal amount. Low magnitude vorticity concentrations are evident in the VC1 contour plot in Figure 34 at 40 seconds which were not evident in previous plots. These low magnitude secondary vortices are numerical artifacts which result from small amounts of vorticity generated from the impulsive start of the initial condition. Over the full duration of the calculation these pockets of diffused vorticity are compressed to form secondary vortical regions.

To complete the full 40 second simulation on 8 processors required a total run time of 2831 seconds for the baseline case, 3195 seconds for the VC1 case, and 3202 seconds for the VC2 case. The time required for the VC routine does not account for the 13% increase in required simulation time. The additional time results from the solution of the Poisson equation solver for the pressure-correction step, which is the main driving component for solution speed in the PISO and SIMPLE algorithms. The pressure correction step is solved in the vortex simulation using a Conjugate Gradient Krylov subspace solver with a diagonal incomplete-Cholesky preconditioner. In this case, the inclusion of VC appears to cause increased sub-iterations of the Conjugate Gradient solver, suggesting that a different preconditioner could be more efficient for the simulation. In other situations VC may assist the preconditioner and drive the solution faster towards convergence. This behavior is similar to the effects of other numerical methods and cannot be predicted but must be evaluated on a case-by-case basis to determine the optimal combination of solution algorithms.
5.4 Flow over a Backward-facing Step

The previous studies have demonstrated the performance of the developed Vorticity Confinement solvers to approximate thin turbulent boundary layers and convect vortices over long distances. These simple simulations, although educational regarding the observed model characteristics, are only two-dimensional whereas true turbulent flow is three-dimensional. A classical three-dimensional problem studied numerically and experimentally is the “backward-facing step” or “rearward-facing step”, a simple geometry which nonetheless induces a variety of flow regimes including boundary layers, a mixing layer, reattachment and flow reversal. Due to the complexity of the generated flow, the backward-facing step has proven to be an excellent test case for the validation of turbulence models, in particular large-eddy simulation (LES) models. Experimental studies conducted by Pitz and Daily [70] and Keller et al. [71] with non-reacting and reacting flows using a standard backward step geometry have been replicated numerically for the purpose of validating LES combustion models [72,73]. For this case study, experimental and numerical results from the non-reacting studies will be compared against time-averaged data from studies utilizing the VC1 and VC2 formulations. These data include mean velocity and RMS velocity profiles at consecutive vertical spans downstream of the step, as well as reattachment lengths. RMS velocity indicates the level of turbulent fluctuations present in the solution, while reattachment length represents the distance to the point of boundary reattachment for the shear layer. Reattachment length has been found in previous studies to be a function of Reynolds number as well as the separating boundary layer characteristics [71]; as a result, it is expected that varying the confinement strength parameter will have an effect on reattachment length. To maintain consistency with prior studies for comparison, reattachment length is defined as the extent of the reverse streamwise velocity in the circulation zone.
5.4.1 Setup Details

The backward-facing step geometry replicating the experimental setup of Pitz and Daily [70] is shown in Figure 35. The streamwise (x), normal (y), and spanwise (z) dimensions have been normalized by the step height \( h = 0.0254 \) m. The experimental Reynolds number \( Re = 22,100 \) is based on the step height and an average inlet velocity \( U = 13.3 \) m/s. The domain extends for approximately 8 step lengths before narrowing in a smooth contraction to the exit plane. This geometry is identical to the domain used for the LES study performed by Park and Ko [73]. Figure 36 highlights a representation of the mesh, a non-uniform grid comprised of 248 x 57 x 20 cells in the streamwise, normal, and spanwise directions, which is similar to that used in the Park and Ko LES study. Wall functions are used for the boundary layer in the LES study, and as a result the mesh is not overly refined near the boundaries. This non-uniform LES-typical mesh is used for the initial Vorticity Confinement studies to ensure an accurate comparison to both experimental data as well as the numerical studies. To account for the variation in cell size, the confinement strength is scaled by a non-dimensional proportion of local cell volume to the average cell volume for the mesh, similar in concept to the scaling method proposed by Butsuntorn [33]. Wall boundaries are set to a no-slip condition while the spanwise-normal surfaces are

![Figure 35. Schematic representation of backward-facing step geometry (h = 0.0254 m).](image-url)
specified as periodic boundaries to simulate the larger spanwise distance between walls in the experiment.

As stated previously, an implicit second-order backward differencing temporal scheme is used instead of the implicit Crank Nicholson scheme to match the setup of the LES simulation. Each case is run for an initial duration of 0.1 seconds which allows for approximately four flow exchanges through the domain to establish a regular flow pattern. Statistical quantities are subsequently time-averaged over an additional 0.1 seconds with a time step of 1E-05 s, equivalent to a maximum CFL number of approximately 0.2, to ensure a sufficient statistical dataset.

5.4.2 Mean Flow Comparison

The case studies presented here are compared to the experimental results of Pitz and Daily [70] and the numerical LES results of Park and Ko [73] at profiles of streamwise mean velocity and streamwise RMS velocity defined at specific locations downstream of the step, as well as measured and computed reattachment lengths. The LES simulation is chosen as an example of accurate conventional turbulence modeling using a similar methodology to VC. LES resolves the large scales of turbulence directly using cell size as an explicit scale filter and computes interactions with the smaller scales through an explicit sub-grid model [74]. The sub-grid model acts to dissipate the under-resolved smaller scales. In comparison, an implicit large-eddy simulation (ILES) approach requires
no sub-grid model, but instead relies on the inherent numerical dissipation to provide proper turbulence decay. Vorticity Confinement can be generalized as a type of ILES method which resolves scales corresponding to the cell size and models the effects of the sub-grid scales through capture of boundary layers and vortex sheets, assumed to be the smallest scales [15]. With proper resolution and confinement parameters VC should result in a similar accurate solution to the explicit LES simulation.

To provide a more direct comparison with the VC models, an additional case is solved on the mesh in Figure 36 with VC switched off resulting in a laminar Navier Stokes solution on the backward-facing step geometry with no explicit dissipation other than that provided by the physical viscosity necessary to achieve the appropriate Reynolds number given in the experiment. The case provides a baseline solution to contrast with the VC results. Differences highlight the characteristics of the flow which are truly affected by the VC model while similarities establish which base flow phenomena are resolved due to the level of grid refinement.

A list of the studied cases detailing simulation parameters is given in Table 1 and includes the computed reattachment lengths based on the time-averaged streamwise velocity. For the backward-facing step the stability of the VC1 model

<table>
<thead>
<tr>
<th>Methodology</th>
<th>$\varepsilon$ value</th>
<th>$\nu$ value</th>
<th>Reattachment length X/h* (accuracy ±0.5h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No VC</td>
<td>0.0</td>
<td>1.536E-5</td>
<td>5.8</td>
</tr>
<tr>
<td>VC1</td>
<td>0.25</td>
<td>1.536E-5</td>
<td>6.5</td>
</tr>
<tr>
<td>VC2</td>
<td>0.0016</td>
<td>0.001</td>
<td>5.8</td>
</tr>
<tr>
<td>VC2</td>
<td>0.0020</td>
<td>0.001</td>
<td>5.5</td>
</tr>
<tr>
<td>VC2</td>
<td>0.0025</td>
<td>0.001</td>
<td>4.4</td>
</tr>
</tbody>
</table>

*Reattachment length computed from time averaged streamwise velocity varies over time with stated accuracy. LES and experimental studies report reattachment lengths of 6.8 and 7.0, respectively, with similar accuracy.
was found to be invariant with respect to the value of $v$ and as a result $v$ was set equal to the laminar viscosity. Since the $v$ parameter represents both physical and numerical viscosity this suggests that the implicit numerical diffusion inherent in the discretization scheme was sufficient to balance the confinement parameter without any added explicit diffusion. The behavior of the VC2 scheme, however, was found to be heavily dependent on the ratio of $\varepsilon$ to $v$. The value of $v$ was held at a constant 0.001 m$^2$/s while various confinement values between 1.5$v$ and 3.0$v$ were investigated. The VC2 cases mentioned here represent the most appropriate range for discussion matching the Pitz and Daily experimental data.

As stated in Table 1, the experimental and LES studies report reattachment lengths of 7.0 and 6.8, respectively, with an accuracy of ±0.5h. Reattachment length grows with increasing laminar Reynolds number, but reduces to a constant value dependent on geometry after transition to turbulent Reynolds number [75]. For the studies here, reattachment length is measured by observing the extent of the recirculation zone defined by the mean streamwise velocity. A representative iso-volume plot is shown in Figure 37 for the laminar solution; iso-volume plots for the remaining cases are found in the Appendix in Figure A-1 through Figure A-4 for reference. The iso-volume gives a volumetric representation of all values of mean streamwise velocity less than zero. While the reattachment point for the VC1 case lies within the tolerance band defined in the experiment, the VC2 cases exhibit a reduction in reattachment length as $\varepsilon$ is increased, deviating from the constant experimental value around 7.0. Variation of the reattachment point has been studied by Nait Bouda et al. [76] who report that the reattachment point varies in time about its mean position and is affected by the large eddies generated within the shear layer. Furthermore, their research indicates that larger eddies can cause the reattachment point to fluctuate and cause the mean reattachment length to decrease. The reduction in reattachment length for the higher $\varepsilon$ VC2 cases suggests that the VC2 model is enhancing the large eddies in the shear layer enough to cause a reduction in mean reattachment length, similar to the effect reported by Nait Bouda et al [76].
The time-averaged streamwise velocity contour for the laminar solution (no VC) in Figure 38 is compared with a streamwise contour for the VC1 case in Figure 39. The VC1 solution exhibits the most significant difference in the confinement of the boundary layer of the upper wall near the beginning of the contraction; in the laminar solution the boundary layer thickens at this point which accompanies a shorter recirculation zone directly below at the lower wall. The time-averaged flow characteristics of the VC1 case compared to experimental data, LES results, and the laminar solution are given in Figure 40 and Figure 41. The mean flow profiles for the laminar solution and the VC1 solution in Figure 40 compare well with both experimental data and the LES solution. The VC1 model shows small regions of improvement in mean velocity profile at X/h = 1 and X/h = 3 within the recirculation region (-1 < Y/h < 0) compared to the laminar solution which suggest that the VC1 model more accurately captures the circulatory behavior. The streamwise RMS velocity profiles for the laminar solution and VC1 solution in Figure 41 show satisfactory comparison to the experimental data.
Figure 38. Mean streamwise velocity contour for laminar solution (no VC).

Figure 39. Mean streamwise velocity contour for VC1 solution. Contour levels from Figure 38 apply.
Figure 40. Mean streamwise velocity profiles from VC1 solution compared to experimental data, LES results, and the laminar solution. Y-axis position has been normalized by step height, while velocity has been normalized by the prescribed inlet velocity.
Figure 41. Streamwise RMS velocity profiles from the VC1 solution compared to experimental data, LES results, and the laminar solution.
On average the laminar solution under-predicts the level of turbulent fluctuations in the free stream, seen as a deficit in the RMS velocity plot at X/h = 5 compared to the experiment profile. Further deviations of the laminar solution from experimental data are seen as an over-prediction of RMS velocities in the region near the boundary layer at all profiles downstream of X/h = 1. In contrast to the laminar solution, the VC1 model appears to more accurately predict the RMS velocity profile, as indicated by a close correspondence to the shape of the experimental data in the free stream. In particular the result at X/h = 1 matches the peak experimental velocities within the shear layer. Due to the application of Surface Confinement the freestream velocity is maintained to a point very close to the wall, resulting in an over-prediction of boundary layer velocities near the bottom wall at X/h = 5 and X/h = 7. This behavior is expected as VC does not allow the boundary layer to diffuse. In this VC1 study both Field Confinement and Surface Confinement use a confinement value of $\varepsilon = 0.25$; the over-prediction at the boundary layer indicates better results may be achieved near the boundary by varying the confinement strength for Surface Confinement.

Mean streamwise velocity contours for the VC2 solutions with varying confinement strength values are presented in Figure 42. It becomes apparent from the comparison between cases that at lower $\varepsilon$ values the boundary layer is still allowed to thicken, similar to the behavior seen in the 5.1 case study. An increase in the value of $\varepsilon$ results in a domain-wide decrease in boundary layer thickness, as well as an increase in velocity along the top wall. The behavior is similar to an increase in Reynolds number, in the sense that VC2 at higher confinement strengths counteracts more numerical diffusion within the solution, resulting in a lower effective flow viscosity (and therefore higher effective Reynolds number). Due to this effect, there can be a fine line between too little confinement and too much confinement with VC2 when comparing to experimental data obtained at a certain Reynolds number.
Figure 42. Mean streamwise velocity contours for VC2 solutions; $\varepsilon = 1.6\nu$ (top), $\varepsilon = 2.0\nu$ (middle), $\varepsilon = 2.5\nu$ (bottom).
The time-averaged flow characteristics of the VC2 cases compared to experimental data, LES results, and the laminar solution are presented in Figure 43 through Figure 48. The mean velocity profiles from the \( \varepsilon = 1.6 \nu \) solution in Figure 43 do not correlate with the experimental data any better than the laminar solution, with the exception of the \( X/h = 7 \) profile in which the VC2 solution lies directly on the experimental profile. The RMS velocity profiles in Figure 44 at this confinement strength, however, provide the closest match to the experimental data by resolving both the peak RMS magnitudes as well as the overall profile shapes at each measurement location. Realistically, the underlying laminar solution already provides satisfactory resolution of the mean velocity profile; the marginal level of confinement is enough to affect the RMS fluctuations but not provide any improvement in the mean velocity profiles. The time-averaged velocity profiles for \( \varepsilon = 2.0 \nu \) in Figure 45 show some improvement with the higher confinement strength at \( X/h = 1 \), where the recirculation velocity has shifted to correspond to the given data. The shapes of the mean velocity profiles at both \( X/h = 1 \) and \( X/h = 3 \) provide a better correspondence to the experimental data within the shear layer than even the reference LES solution. However, in Figure 46 the RMS velocity profiles begin to show evidence of over-prediction within the recirculation region and near the boundaries. This indicates that the level of RMS fluctuations has exceeded what was measured during the experiment. Increasing the confinement strength further to \( \varepsilon = 2.5 \nu \) leads to excessive overshoot near the boundaries in both the mean velocity profiles in Figure 47 and the RMS velocity profiles in Figure 48. The over-prediction of the RMS velocity is present in only the vortical regions of the flow, as evidenced by the RMS profile at \( X/h = 1 \) in which the portion of the profile crossing the inlet stream matches the experimental measurements, while the lower portion of the profile within the recirculation zone exhibits more than 2x increase in RMS magnitudes. The behavior suggests that for this type of internal flow where the boundary layer thickness is important a more conservative value of the confinement strength may be necessary to provide a reasonable correlation to experimental data.
Figure 43. Mean streamwise velocity profiles from the VC2 $\varepsilon = 1.6\nu$ solution compared to experimental data, LES results, and the laminar solution.
Figure 44. Streamwise RMS velocity profiles from the VC2 $\varepsilon = 1.6\nu$ solution compared to experimental data, LES results, and the laminar solution.
Figure 45. Mean streamwise velocity profiles from the VC2 $\varepsilon = 2.0\nu$ solution compared to experimental data, LES results, and the laminar solution.
Figure 46. Streamwise RMS velocity profiles from the VC2 $\epsilon = 2.0\nu$ solution compared to experimental data, LES results, and the laminar solution.
Figure 47. Mean streamwise velocity profiles from the VC2 $\varepsilon = 2.5\nu$ solution compared to experimental data, LES results, and the laminar solution.
Figure 48. Streamwise RMS velocity profiles from the VC2 $\varepsilon = 2.5\nu$ solution compared to experimental data, LES results, and the laminar solution.
5.4.3 Instantaneous Flow Comparison

Time-averaged experimental data comparisons provide a good method to gauge the accuracy of simulation results; however, to gauge the potential of a particular model instantaneous flow field comparisons can also be an effective tool. Figure 49 displays horizontal section slices of the downstream vorticity flow field at the $Y = 0$ height (level with the step height) for each case at the 0.2 s time step. The plots effectively provide a cross-section of the shear layer for each case, allowing direct comparison of the baseline laminar solution with the VC1 and three VC2 solutions. The flow is characterized by the separation at the leading edge of the step; the shed vorticity sheet rolls up into a three-dimensional vortical structure which initially spans the domain and then breaks up into smaller vortices. This is observed in Figure 49 as an initial continuous sheet of high vorticity near the step which breaks up into individual sheets and vortices with lower vorticity. The laminar solution, VC1 solution, and VC2 $\varepsilon = 1.6\nu$ solution show similar behavior of the initial vorticity sheet which is supported by the previous correlations with experimental data. The VC2 $\varepsilon = 2.0\nu$ and $\varepsilon = 2.5\nu$ solutions exhibit much earlier break-up of the initial vorticity sheet which explains the poor correlation in the shear layer at $X/h = 1$ in Figure 47; at this station the shear layer has already begun to mix and shows less separation compared to the experiment. The higher $\varepsilon$ VC2 simulations overall show evidence of a more chaotic, robust flow field downstream compared to the laminar solution.

To provide an encompassing picture of the temporal aspects of the flow field, flow visualizations of vorticity for each case spanning 0.1 second duration are included as attachments. Each case includes a vertical center plane visualization of vorticity magnitude and a three-dimensional volumetric-rendered visualization of vorticity. Instantaneous images which correspond to the flow visualizations are displayed in the text for reference. Figure 50 displays center plane vorticity slices from VISUALIZATION 1 and VISUALIZATION 3 for the laminar and VC1 solutions, while Figure 51 displays three-dimensional vorticity images from VISUALIZATION 2 and VISUALIZATION 4 for the same cases.
Figure 49. Horizontal section slices at $Y = 0$ height, level with the step leading edge.
Figure 50. Instantaneous center plane slices of vorticity magnitude, from VISUALIZATION 1, laminar solution (top) and VISUALIZATION 3, VC1 solution (bottom).
Figure 51. Instantaneous volumetric-rendered images of vorticity magnitude, from VISUALIZATION 2, laminar solution (top) and VISUALIZATION 4, VC1 solution (bottom). Contour levels below 2000 are invisible to accentuate large coherent structures. Upper wall solution has been clipped to emphasize shear layer structures.
The laminar solution exhibits fewer vortical structures which begin to dissipate by approximately $X = 6h$. The VC1 solution, on the other hand, shows an enhancement of the vortical structures from the beginning of the shear layer continuing past the beginning of the contraction. VC1 appears to magnify the vortical structures compared to the laminar solution and the effect of surface confinement with VC1 prevents the thickening of the boundary layer across the top wall which is evident in the laminar solution.

Similar instantaneous images of vorticity are presented from VISUALIZATION 5 through VISUALIZATION 10 for the three VC2 solutions in Figure 52 and Figure 53. Increasing the confinement parameter $\epsilon$ clearly has an enhancing effect on the vortical structures, causing the shear layer to mix sooner downstream of the step and finer vortical structures to appear at the highest $\epsilon$ values. Particularly evident in VISUALIZATION 7 and VISUALIZATION 9 is boundary layer separation from the top wall induced by nearby vortical structures in the flow field. As stated in the 5.1 case study discussion, the boundary layer is allowed to separate under the influence of an adverse pressure gradient but can still reattach downstream due to the effects of the VC model. Another notable effect seen in VISUALIZATION 10 is the appearance of random temporary gaps in the flow field. This interesting result demonstrates that the VC model does not prescribe the behavior of the vortical structures, but simply enhances the structures as they propagate downstream, preserving the random and chaotic nature of the turbulent flow field.

In the $\epsilon = 2.0\nu$ and $\epsilon = 2.5\nu$ solutions both large and small scale structures are accentuated within the shear layer; the VC2 model prevents the structures from dissipating through the exit of the domain, causing the boundary layer to reattach sooner in the high $\epsilon$ VC2 cases. It is possible that continued enhancement downstream is a result of the non-uniform cell correction; within the contraction larger cells are present which create larger corrections to account for the cell size variation. As a result, the VC model may be causing the downstream flow in this region to recover too quickly compared to the experiment.
Figure 52. Instantaneous center plane slices of vorticity magnitude, from VISUALIZATION 5 with VC2 $\varepsilon = 1.6\nu$ (top), VISUALIZATION 7 with VC2 $\varepsilon = 2.0\nu$ (middle), and VISUALIZATION 9 with VC2 $\varepsilon = 2.5\nu$ (bottom).
Figure 53. Instantaneous volumetric-rendered images of vorticity magnitude, from VISUALIZATION 2, baseline laminar solution (top left), VISUALIZATION 6, VC2 $\varepsilon = 1.6\nu$ (top right), VISUALIZATION 8, VC2 $\varepsilon = 2.0\nu$ (bottom left), and VISUALIZATION 10, VC2 $\varepsilon = 2.5\nu$ (bottom right). Contour levels below 2000 are hidden to accentuate large coherent structures. Upper wall solution has been clipped to emphasize shear layer structures.
5.4.4 Turbulence Kinetic Energy Spectra

To provide qualification of the turbulent solution, a turbulence kinetic energy (TKE) spectrum versus wavenumber can be generated from the velocity flow field. Real turbulent flows are characterized by the transfer of energy continuously from the largest scales governed by the mean flow to the smallest molecular scales, where energy is dissipated as heat on the molecular level. This energy cascade from large to small scales was observed by Kolmogorov to exhibit similarity at the smallest scales for turbulent Reynolds numbers [77]. The Kolmogorov similarity hypotheses state that for sufficiently high turbulent Reynolds number there is an eddy range known as the inertial sub-range which falls between largest and smallest eddies. Within this range, the turbulent motions are statistically isotropic and the energy cascade process is independent of both mean flow and viscous effects, resulting in a universal form with a transfer rate equal to -5/3 logarithmic slope [77]. Extending this theory to theoretical and numerical calculation, a turbulent solution must exhibit a -5/3 slope region in some portion of the inertial sub-range, otherwise the solution suffers from improper turbulence energy cascading which may result from incorrect turbulence modeling [78]. The TKE spectrum as a function of wavenumber is generated by a Fourier transform of temporal data from a single point by using Taylor’s frozen turbulence approximation, which assumes that spatial correlations (wavenumber) can be approximated by temporal correlations (frequency). In regions of regularly periodic turbulence it provides a necessary metric for turbulence quantification [77].

The non-dimensional TKE spectra as a function of non-dimensional wavenumber are presented for the VC2 $\varepsilon = 1.6\nu$ solution in Figure 54 and $\varepsilon = 2.5\nu$ solution in Figure 55. Additional TKE spectra for the VC1 case and the VC2 $\varepsilon = 2.0\nu$ solution can be found for reference in the Appendix in Figure A-5 and Figure A-6. Each plot presents the respective TKE spectrum along with a spectrum generated from the laminar solution at the same measurement location; the -5/3 slope has been included for reference. The measurement
location has been chosen at approximately $X/h = 2$ to ensure the presence of
turbulence for the correlation. The spectrum and wavenumber have
been non-dimensionalized by the reference step height $h$, the mean velocity over
the measurement period, and the mean TKE magnitude over the measurement
period. Both VC2 spectra in Figure 54 and Figure 55 exhibit the -5/3 slope in the
approximate center of the wavenumber range, consistent with the power-law
spectrum relationship expected by the Kolmogorov hypotheses. The laminar
solution also exhibits a region of -5/3 slope; however, this covers a narrower
range of wavenumbers and quickly begins to steepen in slope at relatively low
wavenumber. Steepening of the slope occurs as the wavenumber enters the
dissipation range. Faster falloff of the -5/3 slope does not necessarily indicate
that the solution is incorrect; however, it can be a precursor to poor resolution
[79]. For all cases the TKE spectra asymptote in the dissipation range to a non-
similar slope compared to the theoretical Kolmogorov TKE spectrum. This
flattening of the dissipation slope is a behavior shared with ILES-type methods;
at best only the inertial range for an ILES fluid is expected to be similar to the
behavior of a real fluid and thus the non-similarity in the dissipation range can be
disregarded [80].

Another observation regarding the TKE spectra of the VC2 solutions is the
expansion of the characteristic -5/3 slope inertial wavenumber region. Energy
cascading begins at lower wavenumbers compared to the laminar solution;
furthermore, as $\varepsilon$ is increased the -5/3 slope region extends further to higher
wavenumbers when comparing the $\varepsilon = 1.6\nu$ and $\varepsilon = 2.5\nu$ spectra. The behavior is
again similar to that of an ILES fluid, such that an enlargement of the -5/3 slope
inertial range is indicative of higher effective Reynolds number flow [80]. This
suggests that the turbulent structures observed in the visualizations at higher $\varepsilon$
values are truly the result of an effective higher Reynolds number. Extension of
the inertial range to higher wavenumbers also suggests that the VC model is
resolving subscale structures not captured in the laminar solution.
Figure 54. TKE spectrum versus wavenumber for VC2 $\varepsilon = 1.6\nu$ solution.
Figure 55. TKE spectrum versus wavenumber for VC2 $\varepsilon = 2.5\nu$ solution.
5.4.5 Coarsened Mesh Solutions

One of the potential effects of the VC2 model resolving more turbulent structures is the relaxation of mesh density requirements while still providing a reasonably accurate solution. To investigate this possibility, the backward-facing step mesh was coarsened in the X and Y coordinate directions. The subsequent mesh is shown in Figure 56 and represents a relatively uniform mesh of 127 x 35 x 20, for a total cell count of approximately 85,000 cells. The refinement in the Z direction was left identical to the original mesh with 20 spanwise grid cells in order to preserve reasonably isotropic cell shapes. While the larger mean cell size will allow a larger time step without exceeding CFL criteria, the same time step was used to ensure conformity of statistical data to the previous results. Initial studies with an identical $\nu$ parameter from previous cases indicated excessive dissipation within the solution; the coarser mesh inherently generates higher numerical dissipation. As a result, for the coarse mesh cases $\nu$ is halved to 0.0005 and confinement strength values of $\varepsilon = 2.0\nu$ and $\varepsilon = 2.5\nu$ are chosen for comparison to a baseline laminar solution.

The mean streamwise velocity profiles for the coarse mesh solutions are depicted in Figure 57. The laminar solution shows slightly more boundary layer separation at the top wall compared to the fine mesh solution in Figure 38, indicative of the higher dissipation present on the coarse mesh. The two VC2 solutions, however, are dramatically different from the VC2 solutions on the fine mesh.

![Figure 56. Coarsened backward-facing step mesh. 127 x 35 x 20](image-url)
Figure 57. Mean streamwise velocity contours for coarse mesh solutions; baseline laminar solution (top), VC2 $\varepsilon = 2.0\nu$ (middle), VC2 $\varepsilon = 2.5\nu$ (bottom).
mesh in Figure 42 with identical confinement strength. The previous solutions tend to over-predict the velocity, whereas the coarse mesh solutions now show a good comparison with the baseline laminar fine mesh contour. This difference highlights the effect of the tuned balance between $\varepsilon$ and $\nu$; achieving the correct balance between the two parameters leads to a realistic result.

The time-averaged characteristics of the coarse mesh cases are portrayed in Figure 58 and Figure 59. The LES comparison has been omitted since the mesh is no longer similar. With mean velocity profiles both VC cases perform well in the shear layer at $X/h = 1$, with the $\varepsilon = 2.5\nu$ case exhibiting a better prediction of the mean reverse velocity magnitude in the circulation zone than the laminar solution. At $X/h = 3$ all the coarse solutions appear to recover in the shear layer too quickly compared to the experiment; however, the VC solutions still predict the correct reverse velocity magnitude. Further downstream it is clear that in the VC cases the flow reattaches to the bottom wall sooner than the experiment suggests, causing a deviation of the mean flow profile. The RMS velocity profiles in Figure 59 indicate a much closer match between the VC solutions and the experimental data with regards to turbulent fluctuations; the laminar solution in general under-predicts the RMS fluctuations and does not even match the profile shape at the $X/h = 5$ and $X/h = 7$ stations, while both VC solutions show excellent agreement with the experimental profiles at all four stations.

Flow visualizations of vorticity are also generated for the coarse mesh cases to explore the temporal variations of the flow. Instantaneous center plane vorticity magnitude contours and volumetric-rendered contours are shown from respective visualizations in Figure 60 and Figure 61 for the baseline laminar solution and two VC2 solutions on the coarse mesh. The enhancement of turbulent vortical structures within the flow as the value of $\varepsilon$ is increased is particularly evident compared to the baseline laminar solution. The shed vorticity sheet at the beginning of the shear layer in the laminar solution extends almost two step lengths before beginning to roll up and separate. The VC2 cases, on
Figure 58. Mean streamwise velocity profiles from the coarse mesh comparing experimental data, the coarse laminar solution, the VC2 $\varepsilon = 2.0\nu$ solution, and the VC2 $\varepsilon = 2.5\nu$ solution.
Figure 59. Streamwise RMS velocity profiles from the coarse mesh comparing experimental data, the coarse laminar solution, the VC2 $\varepsilon = 2.0\nu$ solution, and the VC2 $\varepsilon = 2.5\nu$ solution.
Figure 60. Instantaneous center plane slices of vorticity magnitude on the coarse mesh, from VISUALIZATION 11 with the baseline laminar solution (top), VISUALIZATION 13 with VC2 $\varepsilon = 2.0\nu$ (middle), and VISUALIZATION 15 with VC2 $\varepsilon = 2.5\nu$ (bottom).
Figure 61. Instantaneous volumetric-rendered images of vorticity magnitude on the coarse mesh, from VISUALIZATION 12, laminar solution (top left), VISUALIZATION 14, with VC2 $\varepsilon = 2.0\nu$ (top right), and VISUALIZATION 16, with VC2 $\varepsilon = 2.5\nu$ (bottom). Contour levels below 2000 are hidden to accentuate large coherent structures. Upper wall solution is clipped as in Figure 53.
the other hand, show progressively quicker roll-up behavior of the vortical sheet in the shear layer and exhibit more turbulent structures within the flow. The boundary layer on the coarse mesh VC2 cases appears thicker than the boundary layer on the corresponding fine mesh cases due to the coarsened grid. Vortex-induced separation of the boundary layer is particularly evident in VISUALIZATION 13 and VISUALIZATION 15. In comparison to the behavior exhibited on the fine mesh, the behavior of the VC model on the coarser mesh is more subdued. The inherent numerical dissipation dominates the solution and reduces the effect of the VC confinement parameters over the same range. Additionally, the structures in the coarse mesh VC2 cases are much larger and more coherent than the finer chaotic structures created in the high $\epsilon$ fine mesh VC2 cases observed in Figure 53. The coarser mesh acts as an implicit filter such that mesh size governs the highest resolvable wavenumber cutoff scale above which turbulence cascading is governed by numerical diffusion. As a result, the smallest turbulent structures resolved on the coarser mesh will be larger than those resolved on the finer mesh. Nonetheless, both coarse and finer mesh cases show a good correlation to mean flow data indicating that the important flow features are captured in both cases.

The corresponding TKE spectra are displayed in Figure 62 for the VC2 $\epsilon = 2.0\nu$ case and Figure 63 for the VC2 $\epsilon = 2.5\nu$ case. The laminar TKE spectrum exhibits a very narrow wavenumber range corresponding to the -5/3 slope. In comparison, both coarse mesh VC solutions reveal a central range of wavenumbers at the -5/3 slope, and increasing $\epsilon$ results in an expansion of the -5/3 slope region similar to the behavior of the fine mesh cases. Compared with the fine mesh spectra, the coarse mesh spectra display a flattening of the slope in the dissipation range beginning at a lower wavenumber. ILES simulations exhibit a wavenumber range with steep decay followed by a flattening of the slope near the grid-scale wavenumber [80]. The trend of flattening of the slope at a lower wavenumber with a coarser grid appears consistent with this finding and confirms further the similarities of VC with ILES methods.
Figure 62. TKE spectrum versus wavenumber for coarse mesh VC2 $\varepsilon = 2.0\nu$ solution.
Figure 63. TKE spectrum versus wavenumber for coarse mesh VC2 $\varepsilon = 2.5\nu$ solution.
5.5 Complex Immersed Body Simulation

Many flows of engineering interest require the resolution of complex physical features while simultaneously solving complicated flow fields at turbulent Reynolds number; to achieve accurate solutions for these types of simulations typically requires careful mesh construction. Conventional methods of mesh generation involve either unstructured or structured (block) meshing. Structured mesh generation involves generating conformal blocks within the flow field containing grid size and spacing information from which the resulting fluid mesh can be defined. For complicated geometries, however, this method of meshing is intensive and can require weeks to generate a proper block mesh. Unstructured meshing, on the other hand, does not require block definitions but instead requires only the surface definition. This approach is faster but requires search routines and other procedural checks within the meshing algorithm to prevent the creation of malformed cells. The resulting volume mesh quality is dependent on the surface mesh quality, and quite often several days are needed to prepare a complex geometry with the proper surface definition necessary for meshing. Most commercial CFD tools utilize unstructured mesh generation.

Both structured and unstructured meshing techniques are body-fitted grid methods; there exists a third option, however, using the immersed boundary methods which were discussed in 4.3.2. The intricacies of generating a grid become less important and the burden of resolving the interface between the fluid and solid bodies is shifted to the discretization algorithm. One benefit of this technique is elimination of malformed or skewed cells; the body can be immersed in a uniform isotropic grid, or for problems of large domain size localized block refinement can ensure proper cell sizes to resolve the flow quantities of interest. This allows for standardization of a background fluid mesh with which varying geometries can be quickly analyzed without the need to redefine the mesh for each surface. Such an advantage has applications for fast prototyping in the automotive and motorsports industries, where varying external vehicle components such as side mirrors and downforce-generating wings may be
quickly analyzed to determine the effect on the general flow around the car. To demonstrate the potential of VC as an implicit immersed boundary method to model a complex body, a Formula One racecar is simulated with the VC model at 8° yaw in a 3 m high by 5 m wide wind tunnel environment.

5.5.1 Time-Accurate VC Study

The simulated domain with outer dimensions marked is shown in Figure 64 facing downstream in a perspective view. The F1 car is a 60% scale vehicle which would typically be tested in a wind tunnel with the given test section dimensions. The vehicle is placed with the front wheel base 2.5 meters from the inlet of the domain. This provides ample distance between the immersed body and the inlet to avoid undesired boundary interactions. The domain extends 12.5 meters downstream of the front wheel base to highlight the important wake region behind the F1 car. The wake is typically the most difficult portion of the

Figure 64. 5.5 case study domain with outer boundary dimensions marked.
flow field to properly resolve with conventional methods due to diffusion and spreading of the vortical features; it is expected that the application of VC in both near and far wake regions will result in an improvement in wake resolution. The detailed immersed vehicle is displayed more closely in Figure 65, and is based on a 2010-2011 season F1 car. Resolving the fine details of the car, such as the exact curvature of the front and rear wings and body or the thin rear deflector plates, is important for estimating surface pressures but is of secondary importance for the wake region. To represent the surface without a body-fitted grid, the fine details are approximated by setting the velocity of any cells within the body to zero; naturally this approach neglects some small details but the overall effect of the surface is taken into account. Applying the VC model results in a smooth vorticity sheet generated from the interface between the interior and exterior cells.

The OpenFOAM utility blockMesh has been used to create a coarse initial base Cartesian mesh which matches the dimensions of the domain in Figure 64.
This mesh has been locally refined in both the wake region and around the immersed surface. The OpenFOAM meshing utility snappyHexMesh was altered to perform this function; the modified tool identifies cells within refinement regions as well as cells intersecting the immersed surface and applies 2:1 refinement to specified cell size levels without removing the cells within the body necessary for the immersed boundary algorithm. The resulting refined Cartesian mesh with 2:1 interfaces between cell sizes is shown in Figure 66 and contains 3,400,000 cells.

Constant refinement is used throughout most of the wake region to mitigate the cell size variation sensitivity of the VC2 model suggested by the 5.4 case study results. The cells near the surface of the vehicle are allowed to refine two levels smaller than the wake region to better approximate the curvature of the body with the immersed boundary algorithm.

The VC2 model parameters are set to $\epsilon = 3.0\nu$ with $\nu = 0.001$. Freestream velocity is set to 50 m/s. Both the inlet and the flow field are set to 50 m/s and the time-accurate simulation is impulsively started with a maximum CFL number of 10. After running one second solution time (3.3x flow pass-through) to wash out the impulsive start effects, the time step is halved for a maximum CFL number of 5 to enhance the temporal flow resolution and the simulation is further advanced to achieve a two second effective solution time.

5.5.2 High Resolution RANS Study

The 5.4 case study demonstrated the ability of VC to resolve more turbulent length scales on a coarser mesh than necessary for conventional methods. To provide a high resolution baseline to assess the effectiveness of the VC2 model to simulate detailed flow around the complex immersed body and within the wake region, the VC2 solution is compared to a high resolution 30,000,000 cell Reynolds-averaged Navier-Stokes (RANS) simulation of the same Formula One car performed with the commercial CFD solver STAR-CCM+. The high resolution study is available from a Jacobs Technology internal report investigating yaw effects within an adaptive wall wind tunnel configuration [81].
Figure 66. Cartesian mesh used for 5.5 case study containing 3,400,000 cells; zoomed images show the spanwise refinement through the car (top left) and streamwise refinement along the centerline (top right). The full mesh (centerline slice) is pictured in the bottom image.
The domain for the high resolution study is shown in Figure 67, and includes the upstream contraction section of the wind tunnel as well as the test section. Besides the upstream contraction, the high resolution simulation also includes rolling road and boundary layer removal system effects, rotating wheels, and engine intake and exhaust. These additional elements are not included in the VC simulation because the current implementation of the immersed boundary in OpenFOAM provides only a no-slip wall condition. However, the objective of the comparison is to show the effect of VC within the wake region and this can be achieved without including the additional effects.

The mesh in the high resolution study is a body-fitted grid comprised of 30 million polyhedral cells with near-wall resolution sufficient to apply a realizable k-ε turbulence model with a two-layer wall function. Further explanation of the origins and merits of the k-ε turbulence model can be found in Wilcox [78] and will not be discussed here. Images of the mesh have been included in Figure A-7 in the
Appendix for reference. It is important to note that the unstructured polyhedral cells in STAR-CCM+ are constructed from an initial tetrahedral mesh; each polyhedral cell is created by joining 6-8 tetrahedral cells, implying that the 30 million polyhedral cell mesh is equivalent to a 180-240 million tetrahedral mesh. A mesh of this overall size is required by conventional CFD methods to attempt to resolve near-body and wake effects. Using VC, however, the resolution of smaller scale behavior is possible on a much coarser mesh than required by conventional methods.

5.5.3 Vorticity Field Comparison

An instantaneous snapshot from VISUALIZATION 17 of the vorticity in the wake of the vehicle at the 3.0 second time step is presented for the time-accurate VC simulation in an isometric view in Figure 68. The vorticity has been visualized with a ray-tracing technique allowing all contour levels to be viewed simultaneously. This provides a high degree of detail compared to an iso-contour which allows for only one contour level to be viewed. In the visualization, higher magnitudes of vorticity are represented by green and light blue shades, while lower levels are dark blue. Vorticity leaves the immersed surface of the vehicle in sheets which separate into smaller vortical structures within the wake as they convect downstream. The tip vortices from the front wing appear to be resolved through the full length of the domain and extend away from the vehicle on either side. Wing tip vortices are exceedingly difficult to capture with conventional modeling techniques and yet play a significant role in the determination of wall pressures for an adaptive wall wind tunnel [81].

To compare the VC solution directly to the high resolution RANS study, an iso-contour of vorticity at a single arbitrary magnitude of 150 is visualized for both simulations. The wake refinement for both cases is comparable or even slightly finer with the high resolution case which allows for an accurate and meaningful comparison using this approach. A front perspective view of the instantaneous vorticity iso-contour from the VC solution is shown in Figure 69.
Figure 68. Instantaneous snapshot of vorticity magnitude from VC2 solution at 3.0 s visualized with ray-tracing technique.
Figure 69. Front perspective view of vorticity magnitude iso-contour at a magnitude of 150 for time-accurate VC solution.
A chaotic mix of both large and small scale structures results from the immersed surface and is convected downstream; rather than dissipating, the structures are sustained by the VC model within the uniformly refined wake region. A perspective view of the same iso-contour value is shown for the steady-state RANS simulation in Figure 70. The flow field in the wake region is much more subdued than seen in the VC simulation, resolving only large scale structures shed from points of separation which dissipate shortly downstream. Typically, steady-state simulations are used to resolve the time-averaged behavior of the flow; however, the true time-averaged behavior of the turbulent wake structures seen in the VC simulation would result in a much more dispersed surface than provided by the steady-state iso-contour.

Additional views of both iso-contours from the side of the domain are provided in Figure 71 and Figure 72, and are overlaid against the background mesh in Figure 73. The steady-state RANS result indicates that the vortical structures dissipate fully after approximately two car lengths downstream, while the time-accurate VC solution shows structures traveling the full distance downstream through the domain until a larger mesh size is encountered. The abrupt truncation of the structures occurs because the effect of the VC term changes with the larger cell downstream, and the absolute value of vorticity magnitude decreases with increasing cell size. Nonetheless, with VC the vortical structures are maintained and enhanced for more than three car lengths.

5.5.4 Streamline Comparison

For a final comparison between the VC and RANS simulations, streamline visualizations are displayed in Figure 74 and Figure 75. The streamlines from the VC solution are based on the time-averaged velocity over two second duration, while the RANS streamlines are based on the steady-state velocity field. The key difference between the two streamline plots is the structure of the wake region. The RANS solution suggests the presence of a tight vortical structure located directly behind the rear wing assembly which extends downstream, while the VC
Figure 70. Front perspective view of vorticity magnitude iso-contour at a magnitude of 150 for steady-state high resolution RANS solution.
Figure 71. Side view of vorticity magnitude iso-contour at a magnitude of 150 for the time-accurate VC solution.

Figure 72. Side view of vorticity magnitude iso-contour at a magnitude of 150 for the steady-state high resolution RANS solution.
Figure 73. Comparison between vorticity magnitude iso-contours from the VC solution (top) and the high resolution RANS solution (bottom) for the turbulent wake region with mesh refinement visible.
Figure 74. Streamline visualization based on time-averaged velocity for time-accurate VC simulation, colored by velocity magnitude.
Figure 75. Streamline visualization of steady-state RANS solution, colored by velocity magnitude.
solution resolves a larger vehicle-sized vortical structure which slowly twists downstream. The RANS solution appears to overemphasize the singular vortical nature of the wake, whereas the wake structure in the VC simulation appears more realistic. Furthermore, the VC solution suggests a set of streamlines turning towards the right side wall (oriented looking downstream). This behavior is not evident in the streamlines from the RANS simulation which indicate full downstream flow directed towards the centerline of the vehicle. Wind tunnel studies of Formula One vehicles at yaw have previously reported interactions between the wall and tip vortices originating from the front wing [82], further supporting the streamline directions observed from the VC solution.

The overall lack of resolved structures in the wake region in the RANS simulation highlights the deficiencies of RANS turbulence modeling to resolve the turbulent wake structures; even if the simulation was unsteady, the RANS model would most likely not resolve the structures to the desired fidelity. For turbulent wake simulation the best conventional modeling approach is large eddy simulation. VC has been shown to exhibit qualities similar to an implicit LES method. However, unlike LES simulations VC does not carry such stringent mesh refinement requirements for vortical structure resolution and can still provide a realistic solution on a coarser mesh.

Using a coarser mesh with VC provides several key benefits over a high resolution mesh required in a RANS application. Overall mesh size is directly proportional to hardware requirements as well as solution time; as a result, the ability to solve a problem with less computational cost represents a significant advantage. In this study, the high resolution simulation on 30 million cells required two days to reach steady-state convergence on 46 processors, while the time-accurate VC simulation with 3.4 million cells achieved a 2.3 second effective solution duration in 18 hours on an equivalent number of processors. Not only does the VC solution represent a savings in computational cost and schedule, it also allows the vortical structures in the wake region to be simulated more accurately than with a steady-state RANS turbulence model.
CHAPTER 6
CONCLUSION

In this thesis a set of parallel-computing, three-dimensional, fully functional Navier Stokes solvers with Vorticity Confinement have been developed in C++ using the OpenFOAM computational framework. The solvers are comprised of two separate algorithms for steady-state and parallel simulation, and because they are based on the OpenFOAM framework they can easily interact with all of the computational utilities and tools offered by OpenFOAM. Both VC1 and VC2 Vorticity Confinement formulations have been implemented and studied through simulation. Additionally, a simple immersed boundary method has been developed to be used specifically with the VC model.

The verification cases studied in Chapter 5 indicate that both formulations of VC act to enhance the vortical features within the flow. The effect of VC on boundary layers has been investigated with steady flow over a flat plate. The results with both VC formulations indicate that for a fixed value of \( \nu \) and a given range of \( \varepsilon \) the thickness of the boundary layer can be controlled to a desired height which approximates the shape of a turbulent boundary layer. Without any outside perturbation the boundary layer remains at the desired height along the boundary but can allow separation as necessary in response to outside influences such as an adverse pressure gradient. As a formality the simulations also verify the parallel implementation of the algorithm and exhibit solution uniformity across processor partition boundaries. An additional boundary layer case study demonstrates the implemented immersed boundary method with simulations of an immersed slanted plate in a steady flow. Through studies varying the numerical accuracy of the convection term, the effect of the VC2 formulation for a given set of \( \nu \) and \( \varepsilon \) values is shown to be dependent on the inherent numerical accuracy of the solver which confirms the findings of Butsuntorn and Jameson [32]. The use of a second-order or higher numerical
scheme lowers the total numerical diffusion which must be taken into account when using a set of $\nu$ and $\epsilon$ values for different simulations.

The ‘solitary wave’ characteristic of VC is demonstrated with the time-accurate advection of a compact vortex through a periodic domain. This case study also provides an opportunity to observe the conservative aspects of both VC formulations. The baseline simulation without VC illustrates that the compact vortex dissipates before traveling one meter from the starting location. With both VC formulations, the compact vortex is treated as a ‘solitary wave’ and convected with the background flow without diffusing or losing definition. The VC model effectively behaves as a non-zero full asymptotic solution within the local region of the vortex, preventing further dissipation. VC1 is verified to be non-conservative of momentum, such that the streamwise motion of the vortex lags and does not complete the full intended transit path within the simulated time. The VC2 formulation completes the full transit, verifying its conservative properties.

The first three-dimensional case study analyzes a backward-facing step to provide a comparison of the VC model to experimental data. The performance of the VC1 formulation for this problem is found to be invariant with respect to the value of $\nu$. For the VC2 formulation the results are shown to be heavily dependent on the proportion of $\epsilon$ to $\nu$. The reattachment length predictions are increasingly poor with higher $\epsilon$ values; interrogating the flow field shows that with higher $\epsilon$ values the shear layer mixes too quickly compared to the experiment and the boundary layer is over-accelerated. This behavior appears to be the result of enhancement of both larger and smaller eddies at higher $\epsilon$ values. The VC1 and VC2 solutions show otherwise good correlation to both experimental and LES data, highlighting the potential of VC to act as an implicit LES method. Furthermore, as a turbulent flow qualifier the TKE spectra as a function of wavenumber from the VC solutions indicate an inertial region with reference slope of $-5/3$ matching the Kolmogorov hypothesis. With increasing $\epsilon$ a tendency of the inertial wavenumber range to broaden over a larger span of wavenumbers
is observed, which is indicative of higher Reynolds number effects. An additional set of simulations is analyzed with VC2 on a coarser mesh to indicate whether the VC model is capable of providing identical quality of solution on coarser meshes. Initial results with identical $\nu$ value from the previous simulations indicate a thicker boundary layer similar to a lower Reynolds number solution. The behavior suggests that on a coarser mesh the value of $\nu$ should be decreased relative to an established $\nu$ value for a finer mesh to account for the increased inherent numerical dissipation on the coarser mesh. The VC2 formulation with half the original value of $\nu$ provides a match to the experimental data on the coarser mesh while the laminar solution does not correlate well to the experimental flow field. This suggests that VC is able to resolve higher Reynolds number turbulent phenomena on a coarser mesh than necessary for conventional methods.

The final case study is a demonstration of the VC model with flow over a complex body. A Formula One racecar is modeled in a 3 x 5 meter wind tunnel domain at an 8 degree yaw configuration. A time-accurate VC solution using VC2 with the developed solver on a 3.4 million cell model is computed on 47 processors for 18 hours resulting in a 3.0 second solution time. This case is compared to a steady-state RANS solution using the commercial CFD solver STAR-CCM+ on a 30 million cell model computed on an equivalent number of processors for 48 hours to achieve convergence. With the VC model, the vortical structures in the wake region are resolved and preserved for more than three car lengths while using the RANS model they dissipate entirely within 1.5 car lengths. Streamline plots suggest that the VC solution clearly shows the effect of the front wing tip vortex in the downstream wake region, while the steady-state RANS simulation exhibits no such effect. The comparison demonstrates the deficiencies of the steady-state RANS turbulence model to accurately resolve the wake flow features, even with a high resolution mesh, and suggests that VC is able to preserve the vortical structures in the wake on the coarser mesh.
6.1 Future Work

The current study has identified a range of acceptable confinement parameters within the same order of magnitude which exhibit comparable behavior for several non-similar simulations. Nonetheless, resolution of specific effects in each case requires minor tuning on the parameters within the identified range. Additional studies are needed to help refine and standardize the confinement parameters to acceptable ranges of values for specific flow regimes before using the developed set of solvers as a production CFD code. This process of calibration is expected for the development of any turbulence model, and requires executing more verification cases. Currently the specification of the confinement parameters is somewhat ad hoc on a case-by-case basis; establishing a well-defined range of acceptable parameters will help formalize the method as a potential tool for turbulent simulation. Based on the results of the 5.4 case study indicating higher Reynolds number effects, parameterization of the confinement parameters versus Reynolds number may be a possibility.

Several areas also exist for improvement in both the VC formulation itself and the developed flow solver in OpenFOAM. The results of the immersed slanted boundary case obviously confirmed that the effects of the VC model are dependent implicitly on the numerical diffusion and as a result vary with discretization scheme. Within OpenFOAM, to measure numerical error there exist two local error estimate methods. The Moment Error Estimate derives the solution error from cell imbalances based on higher moments of the solution equations. The Residual Error Estimate, on the other hand, calculates error based on local inconsistencies between face interpolation and volume integration [48]. Studies are required to formulate an explicit dependence of $\varepsilon$ on the local solution error which would potentially help mitigate variations of the VC model between discretization schemes. This type of enhancement would also be locally cell dependent, leading to a dynamic specification of the VC parameter.

The transient effects in the boundary layer of the VC2 formulation in the 5.2 case study also show a potential for the development of a VC-based
boundary layer model which accurately models turbulence behavior near the boundary. Real turbulent boundary layers are characterized by a multitude of rolling vortical structures within the boundary layer, similar to the demonstrated VC2 results. More study is needed to characterize the effects of VC near the boundary and develop a predictive model for an accurate boundary layer.

More studies are also needed to develop a sufficient scaling parameter for non-uniform meshes. The implemented cell size scaling method in this thesis demonstrated questionable results in the downstream portion of the 5.4 case studies where the mesh exhibited higher aspect ratio differences between cells. A sufficient scaling method should not be detrimental and should also help prevent variations of the VC effect across differences in cell size. However, such a scaling method should also not be considered a panacea for variations in mesh density; near-isotropic cell shapes have been verified to work properly with VC and thus meshes for a VC simulation should be designed with certain cell aspect ratio limitations in mind.
BIBLIOGRAPHY
BIBLIOGRAPHY


[66] Personal communication with John Steinhoff, "Discussion on confinement parameters", February 2012.


Figure A-1. Iso-volume of mean streamwise velocity $U < 0.0$ for VC1 case, 5.4 case study.

Figure A-2. Iso-volume of mean streamwise velocity $U < 0.0$ for VC2 case, 5.4 case study with $\varepsilon = 1.6\nu$. 
Figure A-3. Iso-volume of mean streamwise velocity $U < 0.0$ for VC2 case, 5.4 case study with $\varepsilon = 2.0\nu$.

Figure A-4. Iso-volume of mean streamwise velocity $U < 0.0$ for VC2 case, 5.4 case study with $\varepsilon = 2.5\nu$. 
Figure A-5. TKE spectrum versus wavenumber for VC1 solution, 5.4 case study.
Figure A-6. TKE spectrum versus wavenumber for VC2 $\epsilon = 2.0\nu$ solution, 5.4 case study.
Figure A-7. Mesh from high resolution RANS study for 5.5 case study containing 30 million polyhedral cells; zoomed images show the local refinement around the car (top left) and streamwise refinement along the centerline (top right). The section of the mesh (centerline slice) corresponding to the VC2 simulation domain dimensions is pictured in the bottom image.
VITA

Austin Barrett Kimbrell was born in Hendersonville, TN on August 26th, 1985, to the parents of Donald and Deanna Kimbrell. He was homeschooled through primary school and later attended Hendersonville High School, graduating salutatorian in 2003.

After graduation, Austin studied both engineering and art at the University of Tennessee in Knoxville. He completed a year in a German language study abroad program as an exchange student at Phillipps Universität in 2005 where he met his wife, Oxana. Austin returned to the US and graduated summa cum laude from the University of Tennessee in December 2008 with a Bachelor of Science in Mechanical Engineering and a minor in materials science.

Austin accepted a position at Jacobs Technology as a computational fluids engineer and aerodynamicist after graduation. In the fall of 2009, he began studying at the University of Tennessee Space Institute as a part-time graduate student. He graduated with a Masters of Science in Mechanical Engineering in May 2012.