2012

Construction of Polynomials and Linear Weight Calculation for WENO Schemes

Jacob Alan Nelson
University of Alabama at Birmingham

CONSTRUCTION OF POLYNOMIALS AND LINEAR WEIGHT CALCULATION FOR WENO SCHEMES

by

Jacob Nelson

Dr. Roy Koomullil, CHAIR
Dr. Gary Cheng
Dr. David McDaniel

A THESIS

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Master of Science

BIRMINGHAM, ALABAMA

2012

CONSTRUCTION OF POLYNOMIALS AND LINEAR WEIGHT CALCULATION FOR WENO SCHEMES

Jacob Nelson

MECHANICAL ENGINEERING

ABSTRACT

Many applications of computational fluid dynamics (CFD) require higher order spatial accuracy to resolve the flow field. A few examples of these complex flows include heat transfer problems associated with hypersonic flows, flapping wing simulations, turbo-machineries, rotorcrafts, aero-acoustic problems, and transition to turbulence. Methods being used within the CFD community to achieve higher order spatial accuracy include discontinuous Galerkin schemes, weighted essentially non-oscillatory schemes (WENO), and spectral volume (SV) schemes. Complex flow problems involving complex geometries may necessitate the use of generalized grids for the discretization of the domain. Generalized grids contain elements of multiple types and allow for a structured boundary layer near surfaces of interest and an unstructured mesh in the flow domain. This paper details the development of a third order accurate WENO scheme. In addition to providing higher order accurate spatial resolution, the scheme can utilize three-dimensional generalized meshes consisting of four to six sided elements. The WENO scheme has been developed for use with an existing HYB3D flow solver. With the higher order spatial accuracy of the numerical scheme along with the use of generalized meshes, this paper details the groundwork for providing a means for the creation of accurate simulations of complex flow fields involving complex geometries.

Keywords: WENO, Higher Order, CFD, HYB3D, Generalized Grid

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The CFD process of simulating fluid flow involves modeling the flow domain, discretizing the domain into a computational mesh, providing boundary and initial conditions, and iterating a numerical solution based on the governing equations which are provided by the physics of fluid flow. As the use of the CFD and the abilities of high performance computing have increased, so has the complexity of flow problems being studied. These problems involve increasingly complex flow fields as well as geometries. This necessitates the use of spatially accurate flow solvers as well as grid generation techniques that allow for the creation of computational meshes around complex geometries.

Numerous grid generation techniques have been studied for the use of discretizing the domain around complex geometries. Structured grids composed of hexahedral elements show good accuracy in the boundary layer region of a flow domain and are generated with built in connectivity information between neighboring cells [15]. However, they can be hard to adapt to complex geometries and require substantial computational time during the grid generation process. Unstructured grids composed of tetrahedrons are well suited for complex geometries but require an excessive amount of cells to be located near the surface in order for proper resolution of the boundary layer to be obtained [15]. Generalized or hybrid grids combine the advantages of both unstructured and structured meshes by utilizing structured elements around bodies while

using unstructured elements in the flow domain and in transitioning between two structured regions. This allows for accurate boundary layer resolution while maintaining the ability of the mesh to be applied to complex geometries.

A complex flow field can involve large changes in the flow variables over small distances. Resolution of such complex flows requires that the grid generation create a mesh of adequate resolution, as well as for the flow solver to be spatially accurate enough to account for the changes in the flow variables within each cell. A flow solver's spatial order of accuracy is governed by its ability to approximate the numerical flux through each cell-face. A cell-centered finite volume scheme stores the cell average values of the flow variables at the centroid of the cell. The spatial order of accuracy is defined by the way in which these flow variables are extrapolated to the cell-faces for flux calculations. For a first order spatially accurate scheme, the cell average values stored at the centroids of the cells on either side of a face are used to evaluate the numerical flux through that face. A second order spatially accurate scheme uses a linear reconstruction of the flow variables for extrapolation to the cell-face centroids. Similarly, third order spatial accuracy can be achieved by using a quadratic polynomial to reconstruct the flow variables for extrapolation to the cell-face. The degree of the polynomial used to reconstruct the flow variables determines the spatial accuracy of the scheme. However, the process of reconstructing a higher degree polynomial can cause a great increase in computational cost.

A few methods being employed by the CFD community to achieve higher order accuracy include Spectral Volume (SV), Discontinuous Galerkin (DG), Essentially Non-Oscillatory (ENO), and Weighted Essentially Non-Oscillatory (WENO). This paper

details the development of a third order accurate WENO scheme. Harten, et al. [1,17] introduced the higher order essentially non-oscillating (ENO) scheme for structured meshes. Their finite volume method uses a local stencil based on the reference cell and the neighbors of the reference cell. The ENO schemes construct stencils over which a polynomial of the desired order of accuracy is reconstructed. The ENO scheme then selects the stencil with the smoothest flow variable distribution for the final reconstruction [3]. Through this method, discontinuities can be avoided. WENO schemes use a weighted combination of the ENO stencils to create a reconstruction that is of higher order accuracy than the original polynomials in the stencils. The WENO schemes maintain the non-oscillatory nature of ENO by giving more weight to the stencils with smoother flow variable distributions. WENO schemes have shown the capability for higher order discretization of spatial derivatives in the governing equations for simulations using structured meshes [3]. A research group led by Shu [4-6] has developed WENO schemes for triangular and tetrahedral meshes, which have been shown to provide up to fourth order accurate solutions. However, these schemes have restrictions on the types of meshes that can be utilized. Initial results have been reported by Ollivier-Gooch et al. [7] using a k-exact reconstruction of flow variables for higher order spatial accuracy using an unstructured mesh. The WENO schemes in [2] introduced a mapped weighting system in order to increase the accuracy of the numerical scheme. This weighting system has been shown to cause smoother numerical fluxes, better steady state convergence, and greater accuracy over the same stencils.

The work presented in this paper is to extend the third order WENO scheme of Shu et al. [4-6] for use with generalized grids containing 3-dimensional elements of 4 to 6

sides. This WENO scheme uses a weighted average of linear polynomials to achieve the accuracy of a quadratic polynomial. This allows for third order accuracy without the computational cost associated with quadratic reconstruction of the flow variables. The WENO stencil selection procedures and polynomial reconstructions have been completed. However, there is additional work that needs to be completed in order to use the WENO scheme during the flux calculations in the existing HYB3D flow solver. Chapter 2 covers the details of the WENO formulation. Chapter 3 presents validations for various pieces of the implementation needed for the WENO scheme. Chapter 4 goes over results while Chapter 5 provides conclusions and future work.

CHAPTER 2

WENO SCHEME

Governing Equations

The integral forms of the Navier-Stokes equations are taken as the equations that govern the fluid flow. This is written for any control volume Ω as,

$$\int_{\Omega} \frac{\partial Q}{\partial t} dV + \oint_{\partial\Omega} F(Q) \cdot \vec{n}\, dA = \int_{\partial\Omega} F^{v}(Q) \cdot \vec{n}\, dA \tag{1}$$

where $\vec{n}$ is the unit normal vector pointing outward from the control surface $\partial\Omega$ enclosing the control volume, *dA* is the elemental control surface area, *dV* is the elemental volume, *Q* is the conserved variable vector, and *F(Q)* and $F^{v}(Q)$ are the convective and viscous fluxes, respectively. For brevity, the details of the conservative variables and the fluxes vectors are skipped here and it can be found in [15].

Spatial Discretization

This WENO scheme is developed for utilization of generalized grids for which finite-volume schemes are well suited [15]. A cell-centered finite-volume discretization of the governing equation is written as equation (2).

$$\frac{\Delta Q}{\Delta t} V_i + \sum_{j=1}^{k} F_{ij} \cdot \vec{n}\, ds_j = \sum_{j=1}^{k} F_{ij}^{v}(Q) \cdot \vec{n}\, ds_j \tag{2}$$

where V_i is the volume of the cell i, j represents the neighbor of the cell i, Q is the conserved variable vector, and F_{ij} and F_{ij}^{v} are the numerical values of the inviscid and viscous numerical fluxes, respectively. The overall accuracy of a CFD simulation is dictated by the approach used in the estimation of the numerical fluxes passing through a control surface. For a traditional algorithm, the numerical fluxes are evaluated at the surface centroid and are multiplied by the control surface area for the estimation of the total flux. However, for a WENO scheme, the numerical fluxes are calculated at different quadrature points on the surface and a weighted average of the fluxes at those points is used to estimate the total flux passing through the control surface. The inviscid numerical flux passing through a cell-face can be evaluated using equation (3).

$$F_{ij} \cdot \vec{n}\, ds_j = ds_j \sum_{p=1}^{q} w_p\, F_{ij}\left(Q_i\left(G_p,t\right), Q_j\left(G_p,t\right)\right) \cdot \vec{n} \tag{3}$$

where the index p represents the Gaussian point of a q-point scheme, w_p is the Gaussian weight for quadrature point p, and $Q_i\left(G_p,t\right)$ and $Q_j\left(G_p,t\right)$ are the extrapolated values of the conserved variables on either side of the cell-face. The numerical values of the flux can be calculated using any Riemann solver. For a traditional, spatially first order scheme, cell average values from the cells on either side of the cell-face are used for the numerical flux evaluation. In order for a simulation to achieve second order spatial accuracy, a linear reconstruction of the flow variables within the cell is used to extrapolate the flow variables to the cell-face centroids for flux evaluation. The degree of the polynomial used for reconstructing the flow variables can be increased to get a higher spatial accuracy. However, the increase in the degree of the reconstruction polynomials will increase computational time, stencil size, and data storage. The work being

presented in this paper utilizes a weighted average of linear polynomials to achieve the accuracy of a quadratic polynomial.

WENO Formulation

An overall flow diagram of the WENO scheme is given in Figure 1. The first step in the formulation is the reconstruction of flow variables within a cell using a quadratic polynomial and a set of linear polynomials. For a 3-dimensional computational domain, the quadratic polynomial contains 10 coefficients; a minimum of 10 cells are required to estimate these coefficients. Similarly, the linear polynomial contains four coefficients and four cells are required for this reconstruction. The stencil containing the cells used for the quadratic reconstruction is called a big stencil, while the stencils used for linear reconstructions are called small stencils and are subsets of the big stencil. In order to avoid the reconstruction of quadratic polynomials during every time step, each quadratic polynomial is represented as a linear combination of the linear polynomials. This, along with constraints on the exact reconstruction of quadratic distributions of local variables by the linear combination of linear polynomials at the quadrature points and summation of the weights for the linear combination as one, results in an over-determined system of equations for these weights. This over-determined system is solved in a least-square sense to get the linear weights shown in Figure 1. The above procedure is a function of the geometry only, and is carried out as a pre-processing step. The quadratic polynomial is discarded after the calculation of linear weights and third order accuracy is achieved without adding the computational cost associated with a quadratic reconstruction of the flow variables throughout the simulation. A special treatment is given for the presence of

negative linear weights found in the least-square solution. During the time iteration process, a smoothness indicator for the distribution of flow variables is calculated for every linear polynomial. Using the smoothness indicator, the linear weights are modified to avoid any spurious oscillations during the reconstruction, and to improve the stability of the numerical simulation [6]. The modified linear weights are denoted as non-linear weights and are used to combine the linear polynomials for the extrapolation of the flow variables to the quadrature points for flux estimation.

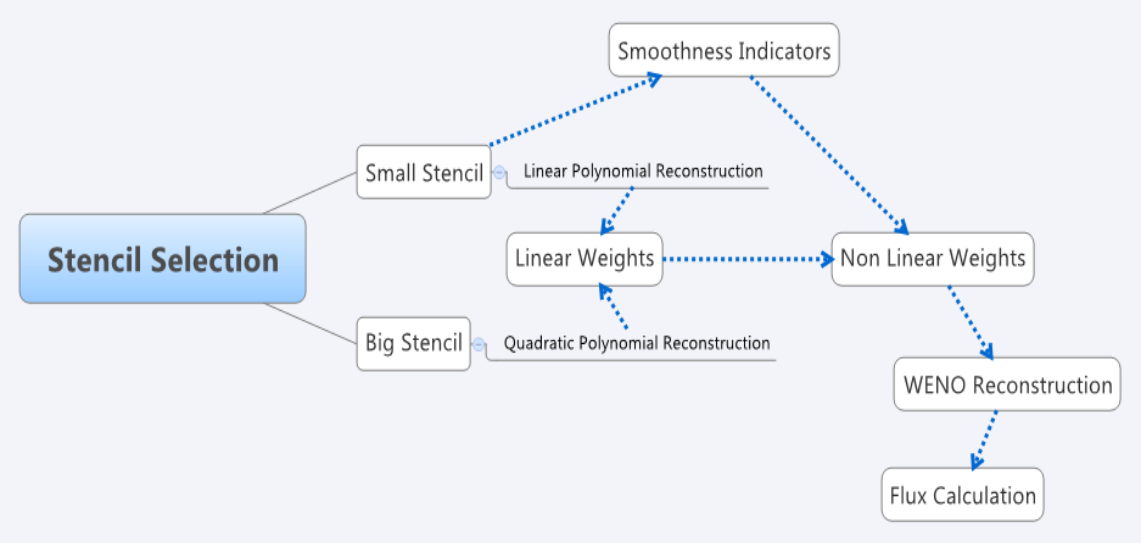

Figure 1. Overview of WENO Scheme

Quadratic Polynomial Reconstruction

A quadratic polynomial representing the flow variable distribution is constructed over a collection of cells called a big stencil. This section details how the quadratic polynomial is manipulated so that it can be expressed in terms of geometry dependent constants, called big stencil constants, and cell average values. This manipulation is done so that the big stencil constants can be used in the linear weight calculation to make the

linear combination of the linear polynomials achieve the same accuracy as the quadratic polynomial. The quadratic polynomial is given as equation (4).

$$p(x,y,z) = a_0 + a_1\xi + a_2\eta + a_3\zeta + a_4\xi^2 + a_5\eta^2 + a_6\zeta^2 + a_7\xi\eta + a_8\zeta\xi + a_9\eta\zeta \tag{4}$$

The local variables appearing in equation (4) are defined by equation (5) [6].

$$\begin{aligned} \xi &= \frac{(x - x_0)}{h} \\ \eta &= \frac{(y - y_0)}{h} \\ \zeta &= \frac{(z - z_0)}{h} \end{aligned} \tag{5}$$

where (x_o, y_o, z_o) are the coordinates of the centroid of the reference cell V_o, and $h = (V_o)^{1/3}$ [6]. There are 10 coefficients in equation (4), giving the requirement of a big stencil population of 10 or more cells. The quadratic polynomial is determined by requiring that for a given variable, u, the polynomial has the same cell average as the reference cell while matching the cell averages of the other big stencil cells in a least-squares sense [6]. For cell i, the cell average of variable u is defined by equation (6).

$$\bar{u}_i = \frac{1}{V_i}\int_{V_i} u(x,y,z)\,dV_i \tag{6}$$

where V_i is the volume of cell i. Each reconstruction is done at a fixed time level, so the time variable can be left out of equation (6) [6]. The cell average values obtained from this integration for the local variables are converted into surface integrals using the Gauss theorem. The surface integrals are then approximated using a Gaussian quadrature scheme. For more information on the Gaussian quadrature schemes employed, see Appendix A. Taking the integral of equation (4) over cell i results in equation (7).

$$\bar{u}_i = a_0 + a_1 \frac{1}{V_i}\int_{V_i} \xi dV_i + a_2 \frac{1}{V_i}\int_{V_i} \eta dV_i + a_3 \frac{1}{V_i}\int_{V_i} \zeta dV_i + a_4 \frac{1}{V_i}\int_{V_i} \xi^2 dV_i + a_5 \frac{1}{V_i}\int_{V_i} \eta^2 dV_i + a_6 \frac{1}{V_i}\int_{V_i} \zeta^2 dV_i + a_7 \frac{1}{V_i}\int_{V_i} \xi\eta dV_i + a_8 \frac{1}{V_i}\int_{V_i} \xi\zeta dV_i + a_9 \frac{1}{V_i}\int_{V_i} \eta\zeta dV_i \tag{7}$$

Applying the definition of cell average values, equation (6), to the local variables, equation (5), produces equation (8).

$$\begin{aligned}
\overline{\xi_i} &= \frac{1}{V_i}\int_{V_i} \frac{(x - x_0)}{h} dV_i \\
\overline{\eta_i} &= \frac{1}{V_i}\int_{V_i} \frac{(y - y_0)}{h} dV_i \\
\overline{\zeta_i} &= \frac{1}{V_i}\int_{V_i} \frac{(z - z_0)}{h} dV_i \\
\overline{\xi_i^2} &= \frac{1}{V_i}\int_{V_i} \frac{(x - x_0)^2}{h^2} dV_i \\
\overline{\eta_i^2} &= \frac{1}{V_i}\int_{V_i} \frac{(y - y_0)^2}{h^2} dV_i \\
\overline{\zeta_i^2} &= \frac{1}{V_i}\int_{V_i} \frac{(z - z_0)^2}{h^2} dV_i \\
\overline{(\xi\eta)_i} &= \frac{1}{V_i}\int_{V_i} \frac{(x - x_0)(y - y_0)}{h^2} dV_i \\
\overline{(\xi\zeta)_i} &= \frac{1}{V_i}\int_{V_i} \frac{(x - x_0)(z - z_0)}{h^2} dV_i \\
\overline{(\eta\zeta)_i} &= \frac{1}{V_i}\int_{V_i} \frac{(y - y_0)(z - z_0)}{h^2} dV_i
\end{aligned} \tag{8}$$

Substitution of the expressions given by equation (8) into equation (7) produces equation (9).

$$\bar{u}_i = a_0 + a_1\overline{\xi_i} + a_2\overline{\eta_i} + a_3\overline{\zeta_i} + a_4\overline{\xi_i^2} + a_5\overline{\eta_i^2} + a_6\overline{\zeta_i^2} + a_7\overline{(\xi\eta)_i} + a_8\overline{(\xi\zeta)_i} + a_9\overline{(\eta\zeta)_i} \tag{9}$$

Applying equation (9) to the reference cell (i.e. $i = 0$) and solving for the coefficient a_o, equation (9) can written in the form of equation (10).

$$a_0 = \bar{u}_0 - \left(a_1\bar{\xi}_0 + a_2\bar{\eta}_0 + a_3\bar{\zeta}_0 + a_4\overline{\xi_0^2} + a_5\overline{\eta_0^2} + a_6\overline{\zeta_0^2} + a_7\overline{(\xi\eta)}_0 + a_8\overline{(\xi\zeta)}_0 + a_9\overline{(\eta\zeta)}_0\right) \quad (10)$$

Equation (11) provides a definition of Δ for any arbitrary variable, given as *f*.

$$\Delta f = f - \bar{f}_o \quad (11)$$

where $\bar{f}_0$ is the reference cell's cell average value of the variable *f*. Some examples of how equation (11) will be used to simplify the notation are provided by $\Delta\xi = \xi - \bar{\xi}_0$ and $\Delta\bar{\xi}_\iota = \bar{\xi}_\iota - \bar{\xi}_0$. Substituting equation (10) into equation (9) and applying equation (11) for all of the local variables produces equation (12).

$$\begin{aligned} p(x,y,z) = \bar{u}_0 + a_1\Delta\xi + a_2\Delta\eta + a_3\Delta\zeta + a_4\Delta\xi^2 + a_5\Delta\eta^2 + \\ a_6\Delta\zeta^2 + a_7\Delta(\xi\eta) + a_8\Delta(\xi\zeta) + a_9\Delta(\eta\zeta) \end{aligned} \quad (12)$$

By taking the integral of equation (12) over cell *i*, equation (13) is generated.

$$\begin{aligned} \bar{u}_i - \bar{u}_0 = a_1\Delta\bar{\xi}_i + a_2\Delta\bar{\eta}_i + a_3\Delta\bar{\zeta}_i + a_4\Delta\overline{\xi_i^2} + a_5\Delta\overline{\eta_i^2} + \\ a_6\Delta\overline{\zeta_i^2} + a_7\Delta\overline{(\xi\eta)}_i + a_8\Delta\overline{(\xi\zeta)}_i + a_9\Delta\overline{(\eta\zeta)}_i \end{aligned} \quad (13)$$

The index *i* is referred to as the big stencil cell number and represents a big stencil cell other than the reference cell, $i = 1, 2, \ldots m\text{-}1$, where *m* is the total big stencil cells. The index *i* can only reach a maximum of *m*-1 because the reference cell is a big stencil cell but is denoted as big stencil cell number 0. Equation (13) can be written in matrix form by equation (14) on the following page.

$$\begin{bmatrix} \Delta\overline{\xi}_1 & \Delta\overline{\eta}_1 & \Delta\overline{\zeta}_1 & \Delta\overline{\xi_1^2} & \Delta\overline{\eta_1^2} & \Delta\overline{\zeta_1^2} & \Delta\left(\overline{\xi\eta}\right)_1 & \Delta\left(\overline{\xi\zeta}\right)_1 & \Delta\left(\overline{\eta\zeta}\right)_1 \\ \Delta\overline{\xi}_2 & \Delta\overline{\eta}_2 & \Delta\overline{\zeta}_2 & \Delta\overline{\xi_2^2} & \Delta\overline{\eta_2^2} & \Delta\overline{\zeta_1^2} & \Delta\left(\overline{\xi\eta}\right)_2 & \Delta\left(\overline{\xi\zeta}\right)_2 & \Delta\left(\overline{\eta\zeta}\right)_2 \\ \Delta\overline{\xi}_3 & \Delta\overline{\eta}_3 & \Delta\overline{\zeta}_3 & \Delta\overline{\xi_3^2} & \Delta\overline{\eta_3^2} & \Delta\overline{\zeta_2^2} & \Delta\left(\overline{\xi\eta}\right)_3 & \Delta\left(\overline{\xi\zeta}\right)_3 & \Delta\left(\overline{\eta\zeta}\right)_3 \\ \Delta\overline{\xi}_4 & \Delta\overline{\eta}_4 & \Delta\overline{\zeta}_4 & \Delta\overline{\xi_4^2} & \Delta\overline{\eta_4^2} & \Delta\overline{\zeta_4^2} & \Delta\left(\overline{\xi\eta}\right)_4 & \Delta\left(\overline{\xi\zeta}\right)_4 & \Delta\left(\overline{\eta\zeta}\right)_4 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \Delta\overline{\xi}_{m-1} & \Delta\overline{\eta}_{m-1} & \Delta\overline{\zeta}_{m-1} & \Delta\overline{\xi_{m-1}^2} & \Delta\overline{\eta_{m-1}^2} & \Delta\overline{\zeta_{m-1}^2} & \Delta\left(\overline{\xi\eta}\right)_{m-1} & \Delta\left(\overline{\xi\zeta}\right)_{m-1} & \Delta\left(\overline{\eta\zeta}\right)_{m-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{bmatrix} = \begin{bmatrix} \Delta\overline{u}_1 \\ \Delta\overline{u}_2 \\ \Delta\overline{u}_3 \\ \Delta\overline{u}_4 \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \Delta\overline{u}_{m-1} \end{bmatrix} \tag{14}$$

The big stencil cell number indicates the order in which the big stencil cells are placed within the big stencil. Each row of the left hand side matrix in equation (14) stores the cell average local variables for one big stencil cell. The row corresponding to a particular big stencil cell will be equal to the big stencil cell number of that cell. For example the first, first level neighbor to the reference cell, as provided by the existing HYB3D data structure, will be big stencil cell number 1 (i = 1), and the cell average local variables for this cell will be located on the first row of the matrix on the left hand side of equation (14). The second, first level neighbor to the reference cell will be big stencil cell number 2 (i = 2), and the cell average local variables for this cell will be located on the second row of the matrix on the left hand side of equation (14). The numbering continues using the HYB3D data structure until the first level of neighbors is exhausted. Then, the numbering of big stencil cells continues according to the ascending order of the second and third level neighbors with respect to their cell number within the grid. These cells are placed in ascending order within the data structure in which they are stored. Equation (14) can be written as equation (15).

$$Ax = b \tag{15}$$

where $A \in \mathrm{R}^{(m-1)\mathrm{x}9}$, $x \in \mathrm{R}^{9}$, and $b \in \mathrm{R}^{m-1}$. The vector x contains the coefficients to the quadratic polynomial, and must be isolated in order to express these coefficients in terms of geometric constants and cell average values. The matrix A will only be square if there are exactly 10 big stencil cells, so the inverse cannot always be taken directly. The vector x must be isolated, so equation (15) is multiplied by the transpose of A, denoted as A^T, which is shown by equation (16).

$$A^T Ax = A^T b \tag{16}$$

Defining a matrix B as $B = A^T A$, equation (16) takes the form of equation (17).

$$Bx = A^T b \tag{17}$$

where B is a square matrix, $B \in \mathrm{R}^{9x9}$. The inverse of the matrix B is calculated, denoted as B^{-1}, and both sides of equation (17) are multiplied by this inverse, resulting in equation (18).

$$x = B^{-1} A^T b \tag{18}$$

Defining a matrix D as, $D = B^{-1}A^T = (A^T A)^{-1}A^T$ and is of the order $D \in \mathrm{R}^{9x(m-1)}$. The substitution of matrix D into equation (18) results in equation (19).

$$x = Db \tag{19}$$

By solving this system of equations for the elements in x, the coefficients to the quadratic polynomial can be expressed by equation (20).

$$a_j = \sum_{i=1}^{m-1} d_{ji} \Delta \bar{u}_i \tag{20}$$

The values of a_j are now only in terms of cell averaged local variables and general cell average values. These a_j variables are the original coefficients to the quadratic polynomial, and substituting equation (20) into equation (12) produces equation (21).

$$\begin{aligned} p(x,y,z) = \bar{u}_0 &+ \left(\sum_{i=1}^{m-1} d_{1i} \Delta \bar{u}_i\right)\Delta\xi + \left(\sum_{i=1}^{m-1} d_{2i} \Delta \bar{u}_i\right)\Delta\eta + \left(\sum_{i=1}^{m-1} d_{3i} \Delta \bar{u}_i\right)\Delta\zeta + \\ &\left(\sum_{i=1}^{m-1} d_{4i} \Delta \bar{u}_i\right)\Delta\xi^2 + \left(\sum_{i=1}^{m-1} d_{5i} \Delta \bar{u}_i\right)\Delta\eta^2 + \left(\sum_{i=1}^{m-1} d_{6i} \Delta \bar{u}_i\right)\Delta\zeta^2 + \\ &\left(\sum_{i=1}^{m-1} d_{7i} \Delta \bar{u}_i\right)\Delta(\xi\eta) + \left(\sum_{i=1}^{m-1} d_{8i} \Delta \bar{u}_i\right)\Delta(\xi\zeta) + \left(\sum_{i=1}^{m-1} d_{9i} \Delta \bar{u}_i\right)\Delta(\eta\zeta) \end{aligned} \tag{21}$$

Rearranging equation (21) by collecting all of the $\bar{u}_i$ terms for every big stencil cell, allows equation (22) to be found.

$$\begin{aligned} p(x,y,z) = \Bigg\{1 - \Bigg[&\left(\sum_{i=1}^{m-1} d_{1i}\right)\Delta\xi + \left(\sum_{i=1}^{m-1} d_{2i}\right)\Delta\eta + \left(\sum_{i=1}^{m-1} d_{3i}\right)\Delta\zeta + \\ &\left(\sum_{i=1}^{m-1} d_{4i}\right)\Delta\xi^2 + \left(\sum_{i=1}^{m-1} d_{5i}\right)\Delta\eta^2 + \left(\sum_{i=1}^{m-1} d_{6i}\right)\Delta\zeta^2 + \\ &\left(\sum_{i=1}^{m-1} d_{7i}\right)\Delta(\xi\eta) + \left(\sum_{i=1}^{m-1} d_{8i}\right)\Delta(\xi\zeta) + \left(\sum_{i=1}^{m-1} d_{9i}\right)\Delta(\eta\zeta) \Bigg]\Bigg\}\bar{u}_0 + \\ &(d_{11}\Delta\xi + d_{21}\Delta\eta + d_{31}\Delta\zeta + d_{41}\Delta\xi^2 + d_{51}\Delta\eta^2 + \\ &d_{61}\Delta\zeta^2 + d_{71}\Delta\xi\eta + d_{81}\Delta\xi\zeta + d_{91}\Delta\eta\zeta)\bar{u}_1 + ... \\ &..... \\ &(d_{1(m-1)}\Delta\xi + d_{2(m-1)}\Delta\eta + d_{3(m-1)}\Delta\zeta + d_{4(m-1)}\Delta\xi^2 + d_{5(m-1)}\Delta\eta^2 + \\ &d_{6(m-1)}\Delta\zeta^2 + d_{7(m-1)}\Delta\xi\eta + d_{8(m-1)}\Delta\xi\zeta + d_{9(m-1)}\Delta\eta\zeta)\bar{u}_{m-1} \end{aligned} \tag{22}$$

At each quadrature point, (x_G, y_G, z_G), the big stencil's quadratic polynomial is written in terms of a series of constants multiplied by cell average values. This takes the form of equation (23) [6].

$$p(x_G, y_G, z_G) = \sum_{i=0}^{m-1} C_i \bar{u}_i \tag{23}$$

The big stencil constants in equation (23) are only dependent on the geometry of the cells making up the big stencil and the coordinates of the quadrature point. The big stencil constants for every big stencil cell in equation (23) are equal to the terms in equation (22) located in front of that cell's average value. The big stencil constant for the reference cell, C_o, can be calculated by equation (24), while the big stencil constants for the rest of the big stencil cells (i = 1, 2, ... m-1) can be calculated with equation (25).

$$C_0 = \begin{aligned} & 1 - \sum_{i=1}^{m-1} d_{1i}\Delta\xi_G - \sum_{i=1}^{m-1} d_{2i}\Delta\eta_G - \sum_{i=1}^{m-1} d_{3i}\Delta\zeta_G - \\ & \sum_{i=1}^{m-1} d_{4i}\Delta\xi_G^2 - \sum_{i=1}^{m-1} d_{5i}\Delta\eta_G^2 - \sum_{i=1}^{m-1} d_{6i}\Delta\zeta_G^2 - \\ & \sum_{i=1}^{m-1} d_{7i}\Delta(\xi\eta)_G - \sum_{i=1}^{m-1} d_{8i}\Delta(\xi\zeta)_G - \sum_{i=1}^{m-1} d_{9i}\Delta(\eta\zeta)_G \end{aligned} \tag{24}$$

$$C_i = \begin{aligned} & d_{1i}\Delta\xi_G + d_{2i}\Delta\eta_G + d_{3i}\Delta\zeta_G + d_{4i}\Delta\xi_G^2 + d_{5i}\Delta\eta_G^2 + \\ & d_{6i}\Delta\zeta_G^2 + d_{7i}\Delta(\xi\eta)_G + d_{8i}\Delta(\xi\zeta)_G + d_{9i}\Delta(\eta\zeta)_G \end{aligned} \tag{25}$$

The subscript G indicates that the local variables are evaluated using the quadrature point's coordinates (x_G, y_G, z_G). The elements of the matrix D are solely dependent on geometry and can be calculated through the use of a geometric moment function and the matrix operations discussed above (equations (15)-(19)). For more information on the geometric moment function and the validation of the cell average local variable calculation, see Appendix B. Since the big stencil constants are solely dependent on geometry, they are calculated as a pre-processing step.

Linear Polynomial Reconstruction

Linear polynomials representing the flow variable distributions are constructed in subsets of the big stencil. This section details how these linear polynomials are manipulated so that they can be expressed as a series of geometrically dependent constants, called small stencil constants, and cell average values. This manipulation is performed so that each linear polynomial is able to reconstruct a flow variable distribution when given the flow variable's average values for the cells contained in the small stencil. Also, linear combinations of the small stencil constants are equated to the big stencil constants during the linear weights calculation. This allows the application of

the linear weights to the combinations of linear polynomials to mimic the accuracy of the quadratic polynomial. The linear polynomial is given by equation (26).

$$p_s(x,y,z) = a_0^s + a_1^s\xi + a_2^s\eta + a_3^s\zeta \tag{26}$$

where the index s = 1, 2, ..., q represents the small stencil number for a total of q small stencils. There are 4 coefficients in the linear polynomial given by equation (26); therefore, each small stencil must be made of 4 cells to create 4 conditions. Each small stencil must include the reference cell. Similar to the quadratic polynomial, the linear polynomial is created in agreement with the cell averages of the local variables of the 4 cells that make up the small stencil [6]. Equation (27) is produced by taking the integral of equation (26) with respect to cell i, and using the definition of cell average values from equation (6).

$$\bar{u}_i = a_0^s + a_1^s \frac{1}{V_i}\int_{V_i}\xi dV_i + a_2^s \frac{1}{V_i}\int_{V_i}\eta dV_i + a_3^s \frac{1}{V_i}\int_{V_i}\zeta dV_i \tag{27}$$

Using the definitions given by equation (8), equation (27) can be written as equation (28).

$$\bar{u}_i = a_0^s + a_1^s\bar{\xi}_i + a_2^s\bar{\eta}_i + a_3^s\bar{\zeta}_i \tag{28}$$

Denoting the reference cell as small stencil member i = 0, and the other 3 members as i = 1,2, and 3, equation (28) can be expressed for all 4 small stencil members by equations (29)-(32).

$$\bar{u}_0 = a_0^s + a_1^s\bar{\xi}_0 + a_2^s\bar{\eta}_0 + a_3^s\bar{\zeta}_0 \tag{29}$$

$$\bar{u}_1 = a_0^s + a_1^s\bar{\xi}_1 + a_2^s\bar{\eta}_1 + a_3^s\bar{\zeta}_1 \tag{30}$$

$$\bar{u}_2 = a_0^s + a_1^s\bar{\xi}_2 + a_2^s\bar{\eta}_2 + a_3^s\bar{\zeta}_2 \tag{31}$$

$$\bar{u}_3 = a_0^s + a_1^s \bar{\xi}_3 + a_2^s \bar{\eta}_3 + a_3^s \bar{\zeta}_3 \tag{32}$$

Equation (29) can be used to solve for the linear polynomial coefficient a_0^s in terms of the reference cell's average local variables. The resulting expression is written as equation (33).

$$a_0^s = \bar{u}_0 - \left(a_1^s \bar{\xi}_0 + a_2^s \bar{\eta}_0 + a_3^s \bar{\zeta}_0 \right) \tag{33}$$

Substituting equation (33) into the original linear polynomial, equation (26), and using the notation provided by equation (11), equation (34) is obtained.

$$p_s(x, y, z) = \bar{u}_0 + a_1^s \Delta\xi + a_2^s \Delta\eta + a_3^s \Delta\zeta \tag{34}$$

Substituting equation (33) into equations (30)-(32), equations (35)-(37) are produced.

$$\Delta\bar{u}_1 = a_1^s \Delta\bar{\xi}_1 + a_2^s \Delta\bar{\eta}_1 + a_3^s \Delta\bar{\zeta}_1 \tag{35}$$

$$\Delta\bar{u}_2 = a_1^s \Delta\bar{\xi}_2 + a_2^s \Delta\bar{\eta}_2 + a_3^s \Delta\bar{\zeta}_2 \tag{36}$$

$$\Delta\bar{u}_3 = a_1^s \Delta\bar{\xi}_3 + a_2^s \Delta\bar{\eta}_3 + a_3^s \Delta\bar{\zeta}_3 \tag{37}$$

Equations (35)-(37) can be written in a general form of equation (38) for i = 1,2,3.

$$\Delta\bar{u}_i = a_1^s \Delta\bar{\xi}_i + a_2^s \Delta\bar{\eta}_i + a_3^s \Delta\bar{\zeta}_i \tag{38}$$

The system of equations given by equation (38) can be expressed in matrix form as equation (39).

$$\begin{bmatrix} \Delta\bar{\xi}_1 & \Delta\bar{\eta}_1 & \Delta\bar{\zeta}_1 \\ \Delta\bar{\xi}_2 & \Delta\bar{\eta}_2 & \Delta\bar{\zeta}_2 \\ \Delta\bar{\xi}_3 & \Delta\bar{\eta}_3 & \Delta\bar{\zeta}_3 \end{bmatrix} \begin{bmatrix} a_1^s \\ a_2^s \\ a_3^s \end{bmatrix} = \begin{bmatrix} \Delta\bar{u}_1 \\ \Delta\bar{u}_2 \\ \Delta\bar{u}_3 \end{bmatrix} \tag{39}$$

Equation (39) takes the form of equation (15), $Ax=b$, with $A \in R^{3x3}$, $x \in R^3$, and $b \in R^3$. The vector x contains the coefficients to the linear polynomial, and it must be isolated. The matrix A will always be a square matrix. Denoting matrix D as the inverse of A, $D=A^{-1}$, and multiplying both sides of equation (39) by D leads to an isolation of the coefficients of the linear polynomial and can be expressed as equation (40).

$$\begin{bmatrix} a_1^s \\ a_2^s \\ a_3^s \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \begin{bmatrix} \Delta\bar{u}_1 \\ \Delta\bar{u}_2 \\ \Delta\bar{u}_3 \end{bmatrix} \tag{40}$$

Expressions for each coefficient are written as equations (41)-(43).

$$a_1^s = d_{11}\Delta\bar{u}_1 + d_{12}\Delta\bar{u}_2 + d_{13}\Delta\bar{u}_3 \tag{41}$$

$$a_2^s = d_{21}\Delta\bar{u}_1 + d_{22}\Delta\bar{u}_2 + d_{23}\Delta\bar{u}_3 \tag{42}$$

$$a_3^s = d_{31}\Delta\bar{u}_1 + d_{32}\Delta\bar{u}_2 + d_{33}\Delta\bar{u}_3 \tag{43}$$

The expressions in equations (41)-(43) are evaluated during every iteration using the cell average values of the flow variables at the given time level. This is done in order to calculate the linear polynomial reconstruction and the smoothness indicator, to be discussed in following sections. The geometry dependent matrix D is stored for every small stencil as a pre-processing step. The linear weights calculation needs the linear polynomial to be expressed as a series of geometry dependent small stencil constants and cell average values. By substituting the expressions for the linear polynomial coefficients from equation (40) into equation (34), equation (44) is produced.

$$\begin{aligned} p_s(x,y,z) = \bar{u}_0 &+ \left(d_{11}\Delta\bar{u}_1 + d_{12}\Delta\bar{u}_{12} + d_{13}\Delta\bar{u}_3\right)\Delta\xi + \\ &\left(d_{21}\Delta\bar{u}_1 + d_{22}\Delta\bar{u}_{12} + d_{23}\Delta\bar{u}_3\right)\Delta\eta + \\ &\left(d_{31}\Delta\bar{u}_1 + d_{32}\Delta\bar{u}_{12} + d_{33}\Delta\bar{u}_3\right)\Delta\zeta \end{aligned} \tag{44}$$

Rearranging equation (44) by collecting all of the $\bar{u}_i$ terms for every small stencil cell, allows equation (44) to be written as equation (45).

$$\begin{aligned} p_s(x,y,z) = \left\{1-\left[\left(\sum_{i=1}^{3} d_{1i}\right)\Delta\xi + \left(\sum_{i=1}^{3} d_{2i}\right)\Delta\eta + \left(\sum_{i=1}^{3} d_{3i}\right)\Delta\zeta\right]\right\}\bar{u}_0 + \\ (d_{11}\Delta\xi + d_{21}\Delta\eta + d_{31}\Delta\zeta)\bar{u}_1 + \\ (d_{12}\Delta\xi + d_{22}\Delta\eta + d_{32}\Delta\zeta)\bar{u}_2 + \\ (d_{13}\Delta\xi + d_{23}\Delta\eta + d_{33}\Delta\zeta)\bar{u}_3 \end{aligned} \tag{45}$$

At each quadrature point, (x_G, y_G, z_G), the linear polynomial in equation (45) is expressed as a summation of the small stencil constants multiplied by the cell average values. This expression is written as equation (46).

$$p_s(x_G, y_G, z_G) = \sum_{i=0}^{3} c_i^s \bar{u}_i \tag{46}$$

The small stencil constants in equation (46) are solely dependent on the geometry of the small stencil members and the coordinates of the quadrature point. At a given quadrature point, the small stencil constants for the reference cell (i = 0) and the other small stencil members (i = 1, 2, 3) are calculated using equations (47)-(50).

$$c_0^s = 1 - \left(\sum_{j=1}^{3} d_{1j}\right)\Delta\xi_G - \left(\sum_{j=1}^{3} d_{2j}\right)\Delta\eta_G - \left(\sum_{j=1}^{3} d_{3j}\right)\Delta\zeta_G \tag{47}$$

$$c_1^s = d_{11}\Delta\xi_G + d_{21}\Delta\eta_G + d_{31}\Delta\zeta_G \tag{48}$$

$$c_2^s = d_{12}\Delta\xi_G + d_{22}\Delta\eta_G + d_{32}\Delta\zeta_G \tag{49}$$

$$c_3^s = d_{13}\Delta\xi_G + d_{23}\Delta\eta_G + d_{33}\Delta\zeta_G \tag{50}$$

The subscript G indicates that the local variables are evaluated using the quadrature point's coordinates (x_G, y_G, z_G). The elements of the matrix D are solely dependent on geometry and can be calculated through the use of a geometric moment function and the matrix operations discussed above (equations (39)-(40)). For more information on the geometric moment function and the validation of the cell average local variable calculation, see Appendix B. The small stencil constant corresponding to the reference cell must be found using equation (47). However, a general equation can be produced for the other small stencil constants from equations (48)-(50) for i = 1, 2, 3, and can be written as equation (51).

$$c_i^s = d_{1i}\Delta\xi_G + d_{2i}\Delta\eta_G + d_{3i}\Delta\zeta_G \tag{51}$$

The linear reconstructions are made using equation (46). These reconstructions use the small stencil constant corresponding to the reference cell from equation (47), and the other 3 small stencil constants from equation (51). Linear reconstructions using equation (46) are performed during the linear weights calculations for the quadratic local variable terms. Reconstructions can be done for any variable of u, provided that the cell average values of u are available for all small stencil members.

Big Stencil Selection

The quadratic polynomial reconstruction for the big stencil contains 10 unknowns. This necessitates 10 conditions, so there must be at least 10 cells populating the big stencil. The first level neighbors are the cells that share a face with the reference cell. The HYB3D flow solver already creates a data structure containing the first level of neighbors for each cell. These are the first cells to populate the big stencil. While only

considering 4 to 6 sided elements, these first level neighbors will not be enough to reach the population requirement of at least 10 big stencil members. The second level of neighbors consists of the cells that share a face with the reference cell's first level of neighbors. Initially, 2 second level neighbors are added to the big stencil for each of the first level neighbors. These second neighbors are the first level of neighbors to the first level neighbors already added. These cells can easily be found by using the existing data structures while checking to ensure that the added cells are not the reference cell or a first level neighbor to the reference cell. These checks reveal the true second level neighbors to the reference cell; however, another requirement is set upon the second level neighbors added to the big stencil because of singularity in the linear local variable matrix during the small stencil constant calculation. The linear local variable matrix is the left hand side matrix of equation (39). In order for the small stencil constants to be calculated, the inverse of this matrix must be found. It is a requirement that all 4 members within a small stencil share at least one face with another member. For this reason, every small stencil that includes a second level neighbor must include the corresponding first level neighbor. Therefore, it is important that the second level neighbors are added to the big stencil in a way that does not allow the values of the local variables produced by their geometry and their corresponding first level neighbor's geometry to cause singularity in the linear local variable matrix. If any of the first level neighbor's centroid coordinates (x_i, y_i, or z_i) equal the corresponding reference cell's coordinates (x_c, y_c, or z_c) then the first level neighbor’s cell average local variable of that coordinate will be equal to zero. Problems with singularity arise if a first level neighbor and the second level neighbor have centroids that share the same 2 coordinates with the reference cell (i.e. both

centroids are in the same plane as the reference cell's centroid). This is a particular problem in the cell geometry produced by a structured boundary layer. Equation (52) shows the linear local variable matrix that is produced when both the first and second level neighbors share the same y and z centroid coordinates with the reference cell.

$$\text{Linear local variable matrix} = \begin{bmatrix} \Delta\overline{\xi}_1 & 0 & 0 \\ \Delta\overline{\xi}_2 & 0 & 0 \\ \Delta\overline{\xi}_3 & \Delta\overline{\eta}_3 & \Delta\overline{\zeta}_3 \end{bmatrix} \tag{52}$$

The determinant of this matrix is always zero, regardless of the cell average values produced by the third member of the small stencil. Figure 2 presents an illustration of a situation in which this can happen.

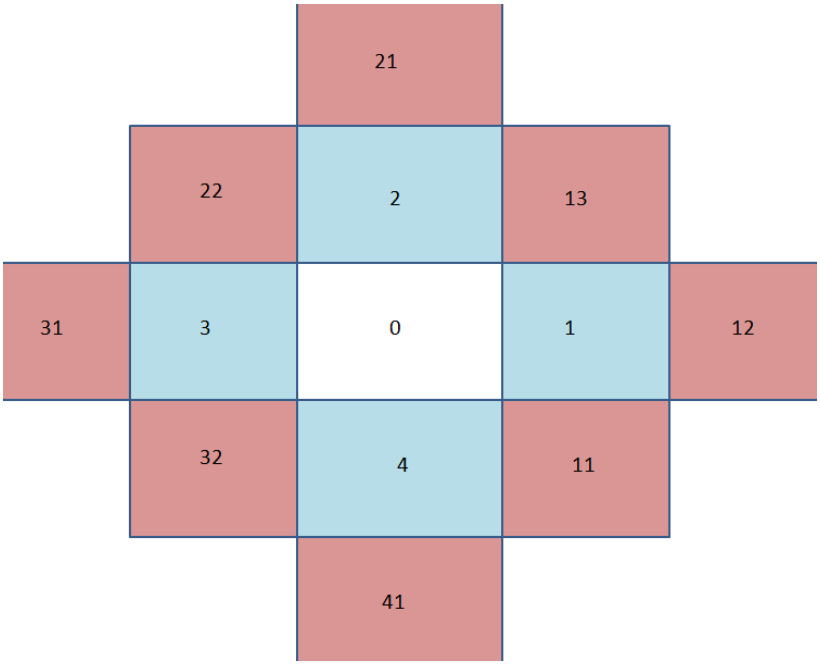

Figure 2: Boundary Layer Illustration

Assuming x-direction is horizontal, y-direction is vertical, and z-direction is into the page, the situation discussed in which both the first and second level neighbors share the same y and z coordinates with the reference cell can be seen in reference cell 0, cells 1

and 12, or cells 3 and 31. To control this situation, a check is made to reveal whether the first level neighbor shares 2 of its centroid coordinates with the reference cell. If this condition is true, then only second level neighbors that do not share those same 2 centroid coordinates will be added for this first level neighbor. In the example shown in Figure 2, this would result in the cell 12 not being added as a second level neighbor to cell 1, cell 21 not being added as a second level neighbor to cell 2, cell 31 not being added as a second level neighbor to cell 3, and cell 41 not being added as a second level neighbor to cell 4. This is the only condition that has been found in which a second level neighbor cannot be added into any possible small stencil candidate without causing a singularity.

The previous procedure can result in duplicate second level neighbors being added for different first level neighbors. After 2 second level neighbors are added for every first level neighbor, a check is made determine whether there are at least 10 unique members making up the big stencil (including the reference cell), while accounting for any duplicate cells that could be in the second level of neighbors. If the number of unique big stencil cells is less than 10, an additional second level neighbor is added for each first level neighbor. If there are still less than 10 unique big stencil cells, then this process is repeated until there are at least 10 unique big stencil cells, or the entire second level of neighbors has been added. It is desirable for the big stencil to be evenly distributed about the reference cell so that the flow variable reconstructions are not pulling information too heavily from one side or the other. This is why the number of unique big stencil cells is only checked after an attempt has been made to add 1 second level neighbor for every member of the first level.

In the event that the reference cell, the first level of neighbors, and entire second level of neighbors do not fully populate the big stencil, then third level neighbors must be added. This situation can occur when the reference cell is located on a boundary and has a reduced amount of first level neighbors. Similar to the process of adding an additional second level neighbor for every first level, a single third level neighbor is added for every second level neighbor. This process is repeated for every second level neighbor until the number of unique big stencil cells is greater than 10. The issue of singularity in the linear coefficient matrix is again addressed in the addition of third level neighbors. If the first level neighbor or the corresponding second level neighbor for a given third level neighbor shares 2 centroid coordinates with the reference cell, then the third level neighbor must not share those 2 centroid coordinates. This would produce a linear local variable matrix with a determinant of zero, similar to the matrix shown by equation (52). Singularity in the linear local variable matrix can also be caused if the first, second, and third level neighbors all share 2 centroid coordinates. The cells' centroids are not in the same plane as the reference cell's centroid (producing values of zero for cell average of the linear local variables), but their centroids are all in the same plane with respect to each other. This makes 2 of the columns of the linear coefficient matrix contain the same values within those columns, producing a very small determinant (order of 1.0E-16). If the first and second level neighbors share 2 centroid coordinates, then a corresponding third level neighbor is only added if its centroid does not also share these coordinates. An illustration of a situation in which this can happen can be seen in Figure 3.

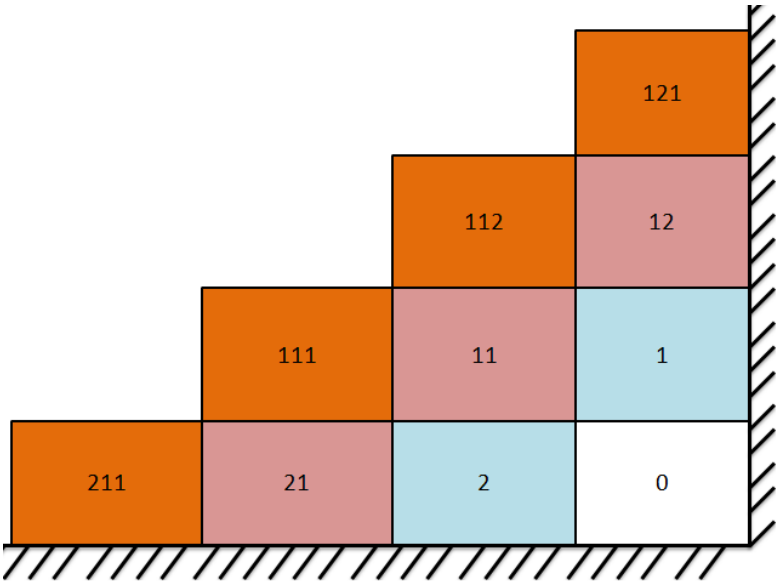

Figure 3: Boundary Illustration

The first level neighbor 2, second level neighbor 11, and third level neighbor 112 are all in the same plane and produce singularity. For this reason, cell 112 is not added to the big stencil for these first and second level neighbors, but cell 111 will be added instead. Similarly, for first level neighbor 1 and second level neighbor 11, third level neighbor cell 111 will produce singularity. Cell 111 is not be selected as a third level neighbor, instead, 112 will be added as the third level neighbor corresponding to first level neighbor 1 and second level neighbor 11.

When the small stencils are formed, all 4 members must share at least one face with another member. This means that for a stencil to include a third level neighbor, it must also include the corresponding second and first level neighbors. This gives only one possible small stencil candidate that includes this third level neighbor. Even with the previous exclusions of third level neighbor additions, it is possible for geometric factors to produce singularity in the linear local variable matrix. For this reason, a determinant check of the linear local variable matrix is performed when adding third level neighbors.

This does not add very large computational cost because third level neighbors are not needed for most cells, and the determinant check for this small stencil is omitted during the small stencil formation process.

Small Stencil Selection

The small stencils are subsets of the big stencil that are used to reconstruct a linear polynomial representing the flow variable distribution. There are 4 unknowns to the linear equation, so each small stencil is made up of 4 cells including the reference cell itself. Additionally, there must be a total of 7 or more small stencils [6]. Every cell in the big stencil must be included in at least one small stencil. This is necessary in order for the small and big stencil constants to be correctly equated by the system of equations when solving for the linear weights. Each of the 4 members of a small stencil must share a face with at least one of the other cells in that small stencil. These requirements are the reason for considering the small stencil formations when forming the big stencil cells. If a given second level neighbor has geometric issues (causing singularities within the linear local variable matrix whose inverse must be found) with the corresponding first level neighbor, it would be impossible to form a small stencil that contained both of those cells with any other applicable candidate cell. The requirements that were put into place in the big stencil selection do not ensure the removal of all singularities found in all possible linear local variable matrices. They do, however, ensure that no cells will be added to the big stencil that cannot be placed in a small stencil. A check has been created that calculates the determinant of the linear local variable matrix before a small stencil is

formed. If the determinant is below a certain threshold, then that small stencil candidate is skipped and other candidates are examined.

The small stencil candidates that can be added are dependent on the cells that were added to the big stencil. For a reference cell with 3 first level neighbors, the first small stencil to be added contains all 3 first level neighbors. If there are 4 first level neighbors, then all 4 possible combinations of the first level neighbors are added. If there are 5 first level neighbors, there are a total of ten possible combinations. Only 5 of these possible combinations are added, and these 5 are selected in such a way that every first level neighbor is in 3 of the small stencils. If there are 6 first level neighbors, then there are a total of twenty possible combinations. Only 6 of these are added and each of the first level neighbors is included in 3 of these small stencils. Adding all possible combinations for these cases causes computational cost problems and can lead the linear weights system of equations to be underdetermined. The determinants are checked when adding these stencil candidates. However, the varying geometric locations of the first level neighbors with respect to the reference cell generally provides manageable values within the linear local variable matrix.

Small stencils involving the second level of neighbors must be added. The procedure of adding these stencils is dictated by how many second level neighbors are present in the big stencil for a given first level neighbor. In the ideal case, the big stencil selection will add 2 second level neighbors for each of the first level neighbors. In this ideal case, the reference cell, the first level neighbor, and both of the corresponding second level neighbors to that first level neighbor are considered as a small stencil candidate. If the determinant check denies this candidate, then alternative small stencil

candidates must be examined that include the 2 second level neighbors. These alternative candidates are made up of 2 of the first level neighbors (1 of which is corresponding to the 2 second level neighbors in question) and 1 of the second level neighbors. The additional first level neighbor used in the previous process can be varied until suitable candidates for both of the second level neighbors are found. If more than 2 second level neighbors have been added for a given first level neighbor, then combinations of those second level neighbors are used to form stencils in a manner that ensures all of the second level neighbors are included in a stencil, without creating excessive stencils. The determinants are checked in each case with varying combinations ready to be added in the event of singularity. If there is only 1 second level neighbor added for a first level neighbor, then a small stencil is created containing the reference cell, the first level neighbor, the corresponding second level neighbor, and an additional first level neighbor. Again, the additional first level neighbor can be varied until a suitable small stencil candidate is found. If there are zero second level neighbors for a first level neighbor then the only additional stencils that can exist for this cell are stencils containing the reference cell, the first level neighbor in question, an additional first level neighbor, and a second level neighbor that corresponds to the additional first level neighbor. However, if there are 3 or more first level neighbors for the reference cell, then the first level neighbor with zero second level neighbors has already been included in at least one small stencil. It is improbable, but not impossible, that a first level neighbor will have zero second level neighbors. If there are third level neighbors present in the big stencil, then small stencils that include the reference cell, the third level neighbor, the corresponding second level neighbor, and the corresponding first level neighbor are created. The determinant check

was already performed on these candidate stencils during the big stencil selection, and can be omitted here.

Initially, this entire procedure was performed in such a way that no duplicate stencils were possible. However, with the singularity issues experienced in the linear weight calculations, changes had to be made that resulted in the possible creation of duplicate small stencils. For this reason, a procedure has been implemented that removes duplicate small stencils after they are generated. Once the duplicates are removed, the total number of small stencils is tallied, and if that total is greater than or equal to 7, then the small stencil selection is complete. If this condition is not met, then additional small stencil candidates are added in such a way that small stencils on any side of the reference cell are not overloaded. It is not desirable for the cell to be weighing information from one side or the other too heavily when reconstructing the flow variable distributions. This procedure shows that the first level of neighbors will be present in more small stencils than the second or third level. They are the closest cells to the reference cell, and their presence in numerous small stencils will give these neighbors more weight when performing the flow variable reconstruction for the reference cell.

When the small stencil selection has been completed, the big stencil cell information for every cell in the grid can be consolidated into one array. Previously, the second level neighbor information was stored in such a way that the corresponding first level neighbor could be identified. Also, if third level neighbors were added, their cell information was stored along with the corresponding first and second level neighbors. This connectivity information is required for proper small stencil selection. Now that the small stencils have been formed, this information can be discarded and any duplicate

cells from the big stencil can be removed. There are possible duplicates in the second and third level of neighbors, but these were accounted for when tallying up the total unique big stencil cells. However, the duplicates were not removed from the data structures. This maintained the connectivity information for proper small stencil formation. The second and third level neighbors are now arranged in ascending order with duplicates removed and added to an array. These cells combined with the first level neighbors in the HYB3D data structure make up the entire big stencil.

Linear Weight Calculation

The purpose of the linear weights is to make a linear combination of the linear polynomials that mimics the accuracy of the quadratic polynomial. If two methods produce the same values at the quadrature points, then equivalent numerical fluxes are calculated. This makes the methods equal in terms of spatial accuracy. The WENO linear weights essentially create third order spatial accuracy by combining linear polynomial reconstructions, which involves less computational cost than a quadratic polynomial reconstruction.

The Linear weights are calculated by simultaneously solving two systems of equations, along with the requirement that the summation of linear weights is equal to one [6]. This requirement is written as equation (53).

$$\sum_{s=1}^{q} \gamma_s = 1 \tag{53}$$

where γ_s is the linear weight for small stencil s. The first system of equations is written as equation (54).

$$\sum_{s=1}^{q} \gamma_s p_s(x_G, y_G, z_G) = u(x_G, y_G, z_G) \tag{54}$$

where p_s is the linear reconstruction of small stencil s for the variable u, which is evaluated at the coordinates of the quadrature point, (x_G, y_G, z_G). The linear reconstruction is performed at the quadrature point and is found using equation (46). This system will be for the local variables of $u = \xi^2, \eta^2, \zeta^2, \xi\eta, \xi\zeta, \eta\zeta$ forming a total of 6 equations. The linear polynomial has 6 degrees of freedom less than the quadratic polynomial, namely for the variables of non-linear terms of u previously mentioned [6]. This system of equations allows for the linear weights, when applied to the linear polynomials, to accurately approximate these quadratic variables of u.

The second system of equations is produced by equating the quadratic polynomial with a linear combination of the linear polynomials at the Gaussian quadrature points. This system of equations is written as equation (55).

$$\sum_{s=1}^{q} \gamma_s p_s(x_G, y_G, z_G) = p(x_G, y_G, z_G) \tag{55}$$

Substituting the form of the quadratic polynomial given by equation (23) and the form of the linear polynomials given by equation (46) allows for equation (55) to be written as equation (56).

$$\sum_{s=1}^{q} \gamma_s \left(\sum_{i=0}^{3} c_i^s \bar{u}_i^s \right) = \sum_{i=0}^{m-1} C_i \bar{u}_i \tag{56}$$

where $\bar{u}_i^s$ is the cell average value of the i-th member of small stencil s. Equation (56) can be expanded in terms of the cell average value of every big stencil cell. Then, terms on either side of the equation being multiplied by the corresponding cell average value

can be equated. This forms a system of equations in terms of the big stencil constants, the small stencil constants, and the linear weights and takes the form of equation (57).

$$c\gamma = C \tag{57}$$

where $c \in R^{mxq}$ contains the small stencil constants, $\gamma \in R^{qx1}$ contains the linear weights, $C \in R^{mx1}$ contains the big stencil constants, m is equal to the number of big stencil cells, and q is equal to the number of small stencils.

For the linear local variables ($u = 1, \xi, \eta, \zeta$), both the quadratic polynomial and the linear polynomials should reproduce the exact values [6]. The summation requirement by equation (53) ensures that the linear combination of the linear polynomials will maintain the ability to reproduce these variables exactly. Combining equations (53), (54), and (57), results in the system of equations given in equation (58).

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & \bullet & \bullet & \bullet & 1 & 1 \\ P_1(\xi^2) & P_2(\xi^2) & P_3(\xi^2) & P_4(\xi^2) & P_5(\xi^2) & P_6(\xi^2) & \bullet & \bullet & \bullet & P_{q-1}(\xi^2) & P_q(\xi^2) \\ P_1(\eta^2) & P_2(\eta^2) & P_3(\eta^2) & P_4(\eta^2) & P_5(\eta^2) & P_6(\eta^2) & \bullet & \bullet & \bullet & P_{q-1}(\eta^2) & P_q(\eta^2) \\ P_1(\zeta^2) & P_2(\zeta^2) & P_3(\zeta^2) & P_4(\zeta^2) & P_5(\zeta^2) & P_6(\zeta^2) & \bullet & \bullet & \bullet & P_{q-1}(\zeta^2) & P_q(\zeta^2) \\ P_1(\xi\eta) & P_2(\xi\eta) & P_3(\xi\eta) & P_4(\xi\eta) & P_5(\xi\eta) & P_6(\xi\eta) & \bullet & \bullet & \bullet & P_{q-1}(\xi\eta) & P_q(\xi\eta) \\ P_1(\xi\zeta) & P_2(\xi\zeta) & P_3(\xi\zeta) & P_4(\xi\zeta) & P_5(\xi\zeta) & P_6(\xi\zeta) & \bullet & \bullet & \bullet & P_{q-1}(\xi\zeta) & P_q(\xi\zeta) \\ P_1(\eta\zeta) & P_2(\eta\zeta) & P_3(\eta\zeta) & P_4(\eta\zeta) & P_5(\eta\zeta) & P_6(\eta\zeta) & \bullet & \bullet & \bullet & P_{q-1}(\eta\zeta) & P_q(\eta\zeta) \\ c_0^1 & c_0^2 & c_0^3 & c_0^4 & c_0^5 & c_0^6 & \bullet & \bullet & \bullet & c_0^{q-1} & c_0^q \\ c_1^1 & c_1^2 & c_1^3 & c_1^4 & c_1^5 & c_1^6 & \bullet & \bullet & \bullet & c_1^{q-1} & c_1^q \\ c_2^1 & c_2^2 & c_2^3 & c_2^4 & c_2^5 & c_2^6 & \bullet & \bullet & \bullet & c_2^{q-1} & c_2^q \\ c_3^1 & c_3^2 & c_3^3 & c_3^4 & c_3^5 & c_3^6 & \bullet & \bullet & \bullet & c_3^{q-1} & c_3^q \\ c_4^1 & c_4^2 & c_4^3 & c_4^4 & c_4^5 & c_4^6 & \bullet & \bullet & \bullet & c_4^{q-1} & c_4^q \\ c_5^1 & c_5^2 & c_5^3 & c_5^4 & c_5^5 & c_5^6 & \bullet & \bullet & \bullet & c_5^{q-1} & c_5^q \\ c_6^1 & c_6^2 & c_6^3 & c_6^4 & c_6^5 & c_6^6 & \bullet & \bullet & \bullet & c_6^{q-1} & c_6^q \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ c_{m-1}^1 & c_{m-1}^2 & c_{m-1}^3 & c_{m-1}^4 & c_{m-1}^5 & c_{m-1}^6 & \bullet & \bullet & \bullet & c_{m-1}^{q-1} & c_{m-1}^q \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \\ \bullet \\ \bullet \\ \bullet \\ \gamma_{q-1} \\ \gamma_q \end{bmatrix} = \begin{bmatrix} 1 \\ (\xi^2)_G \\ (\eta^2)_G \\ (\zeta^2)_G \\ (\xi\eta)_G \\ (\xi\zeta)_G \\ (\eta\zeta)_G \\ C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ \bullet \\ \bullet \\ \bullet \\ C_{m-2} \\ C_{m-1} \end{bmatrix} \tag{58}$$

Let the left hand side matrix and right hand side vector of equation (58) be denoted as LHS and RHS, respectively. Each column of LHS will correspond to a small stencil s for s=1,2,...,q. The summation requirement from equation (53) is written across the first row of LHS and RHS. Rows 2 through 7 provide the system of equations from equation (54), with each row corresponding to a different quadratic local variable term. In the RHS vector, these local variables are directly evaluated using the quadrature point coordinates. The LHS matrix contains the linear polynomial extrapolations to the quadrature point of the local variable corresponding to the row number by the linear polynomial from the small stencil corresponding to the column number. The system of equations from (57) is provided by rows 8 to m+7 of LHS and RHS. Each row from 8 to m+7 corresponds to one big stencil cell. A given big stencil cell will correspond to the row of 8 + its big stencil cell number. The assigning of big stencil cell numbers to each big stencil cell is discussed in the Quadratic Polynomial Reconstruction section. The RHS contains the big stencil constants. This begins with the reference cell's constant in row 8 and continues for all of the other big stencil cells' corresponding constants. Each column of LHS contains the small stencil constants for the small stencil corresponding to that column. Row 8 of LHS side is fully populated by the reference cell's small stencil constants from every small stencil. The notation used in equation (58) changed from that of equation (56) in order to generate a general expression for equation (58). Rows 9 through m+7 of LHS contain only 3 non-zero elements in each column. These non-zero elements are the small stencil constants corresponding to the 3 small stencil members other than the reference cell. The placement of these constants within each column will be dependent on the big stencil numbers of the members making up the small stencil. For

example, small stencil number 3 could contain members with big stencil cell numbers of 0, 2, 4, and 8. For this case, column 3 of LHS contains non-zero elements in rows 8, 10, 12, and 16. However, for a different reference cell, small stencil number 3 could contain members with big stencil cell numbers of 0, 1, 6, and 7. This would result in non-zero elements in rows 8, 9, 14, and 15. This is the reason for the different notation need to make equation (58) a general expression for any reference cell. The solution vector, γ, contains the linear weights. There is one linear weight corresponding to every small stencil. This system of equations can be solved in a least-square sense. A LAPACK routine called DGELSS is used to solve equation (58) for the linear weights.

Smoothness Indicators

During each time iteration, the smoothness of the linear polynomial's flow variable distribution can be calculated by differentiation. It is desirable to give more weight to the linear polynomials with less variation in the flow variable distribution when combining the linear polynomials. This provides more stability to the scheme and avoids spurious oscillations during flow variable reconstruction. The equation for the smoothness indicator calculation is given by equation (59) [6].

$$\beta_s = \sum_{1 \le \alpha \le k} \int_{V_0} V_0^{\frac{2}{3}|\alpha|-1} (D^\alpha p_s(x,y,z))^2 \, dV_0 \tag{59}$$

where β_s is the smoothness indicator for polynomial s, α is a multi-index, D^α is the derivative operator, and k is the degree of the polynomial. The derivative operator, D^α, is defined by equation (60).

$$D^{\alpha} = \frac{\partial^{(\alpha_1+\alpha_3+\alpha_3)}}{\partial x^{\alpha_1}\partial y^{\alpha_2}\partial z^{\alpha_3}} \tag{60}$$

where $\alpha_1 + \alpha_2 + \alpha_3 = \alpha$. The degree of the polynomial in equation (59) is 1, so, $k = \alpha = 1$. Therefore, for 3-dimensions, the order of derivatives in *x*, *y*, and *z* directions $(\alpha_1, \alpha_2, \alpha_3)$ are defined as (1,0,0), (0,1,0), and (0,0,1). With $\alpha = 1$, equation (59) can be written as equation (61).

$$\beta_s = \sum_{1\le\alpha\le 1}\int_{V_0} V_0^{-\frac{1}{3}}(D^{\alpha} p_s(x,y,z))^2 dV_0 \tag{61}$$

where h was earlier defined as, $h = (V_o)^{1/3}$, and remains constant throughout the integration and the summation over the multi-index. This allows for equation (61) to take the form of equation (62).

$$\beta_s = \frac{1}{h}\sum_{1\le\alpha\le 1}\int_{V_0} (D^{\alpha} p_s(x,y,z))^2 dV_0 \tag{62}$$

The linear polynomial in (62) is provided by equation (26). Equation (63) is found by substituting the definitions for the local variables into equation (26).

$$p_s(x,y,z) = a_0^s + a_1^s\left(\frac{x-x_0}{h}\right) + a_2^s\left(\frac{y-y_0}{h}\right) + a_3^s\left(\frac{z-z_0}{h}\right) \tag{63}$$

The three differentiations done by the derivative operator are provided by equations (64)-(66).

$$\frac{\partial P_s}{\partial x} = \frac{a_1^s}{h} \tag{64}$$

$$\frac{\partial P_s}{\partial y} = \frac{a_2^s}{h} \tag{65}$$

$$\frac{\partial P_s}{\partial z} = \frac{a_3^s}{h} \tag{66}$$

The coefficients in equations (64)-(66) are given by equations (41)-(43). These coefficients are dependent solely on the geometry of the small stencil cells and the cell average values of the flow variables for those cells. The smoothness indicator is calculated at a given time level. Therefore, the cell average values of the flow variables will be constant. When the volume integral in equation (62) is applied to the summation of these constants, the result is simply the sum of the constants multiplied by the volume of the reference cell. Equation (62) can now be written as equation (67)

$$\beta_s = \frac{1}{h}\left[\left(\frac{a_1^s}{h}\right)^2 + \left(\frac{a_2^s}{h}\right)^2 + \left(\frac{a_3^s}{h}\right)^2\right]V_0 \tag{67}$$

The terms of *h* can be collected to form equation (68).

$$\beta_s = \frac{1}{h^3}\left[\left(a_1^s\right)^2 + \left(a_2^s\right)^2 + \left(a_3^s\right)^2\right]V_0 \tag{68}$$

Knowing that $\frac{1}{h^3} = V_0$, equation (68) simplifies to the form of equation (69).

$$\beta_s = \left(a_1^s\right)^2 + \left(a_2^s\right)^2 + \left(a_3^s\right)^2 \tag{69}$$

The linear polynomial coefficients are calculated using equations (41)-(43). The smoothness indicator can then be found using equation (69). Smaller values for the smoothness indicator are produced when the flow variable distribution has less variation over the small stencil than if the flow variable distribution shows large fluctuations. Ultimately, this will give larger weight to the small stencils with smoother variable distributions during the flow variable reconstructions.

Non-Linear Weight Calculation

The non-linear weights are modifications of the linear weights. These modifications can be used during the flow variable reconstructions in place of the linear weights in order to increase the stability of the scheme. The non-linear weights are calculated by equations (70) and (71).

$$\tilde{\omega}_s = \frac{\gamma_s}{(\varepsilon + \beta_s)^2} \tag{70}$$

$$\omega_s = \frac{\tilde{\omega}_s}{\sum_{j=1}^{q} \tilde{\omega}_j} \tag{71}$$

where ε is simply a small positive number that is used to avoid dividing by zero. The flow variables can now be extrapolated to the quadrature points using the non-linear weights and linear reconstructions in the small stencils. The values for the smoothness indicators are larger for linear polynomials with large slopes in the flow variable distribution than for a linear polynomial with a smaller slope. When these values are used in equation (70), the resulting non-linear weights are larger for the smoother polynomials and smaller for the polynomials with large fluctuations in the flow variable distribution.

Treatment for Negative Linear Weights

The linear weights are dependent on local geometry and may become negative. Negative weights cause instability during the application of the scheme. Previous work attempts to combat this issue by regrouping the stencils or reducing the order of accuracy. However, Shi et al. [8] have developed a technique to deal with the presence of negative

linear weights without having to modify the stencils or reduce accuracy. This technique involves splitting the linear weights into 2 groups using equations (72) and (73).

$$\tilde{\gamma}_s^{+} = \frac{1}{2}(\gamma_s + 3|\gamma_s|) \tag{72}$$

$$\tilde{\gamma}_s^{-} = \tilde{\gamma}_s^{+} - \gamma_s \tag{73}$$

This is done for every linear weight, s=1,2,.....,q. The 2 groups are then separately scaled by equations (74) and (75).

$$\sigma^{\pm} = \sum_{s=1}^{q} \tilde{\gamma}_s^{\pm} \tag{74}$$

$$\gamma_s^{\pm} = \frac{\tilde{\gamma}_s^{\pm}}{\sigma^{\pm}} \tag{75}$$

Next, 2 groupings of non-linear weights are calculated using the positive and negative groups of the linear weights, γ^+ and γ^-. Equations (70) and (71) now take the form of equations (76) and (77).

$$\tilde{\omega}_s^{\pm} = \frac{\gamma_s^{\pm}}{(\varepsilon + \beta_s)^2} \tag{76}$$

$$\omega_s^{\pm} = \frac{\tilde{\omega}_s^{\pm}}{\sum_{j=1}^{q} \tilde{\omega}_j^{\pm}} \tag{77}$$

The smoothness indicator, β_s, remains unchanged because it is calculated using the linear polynomial reconstructions before any weights are applied.

Flow Variable Extrapolation

After the non-linear weight calculation, any flow variable *u* can be extrapolated to each Gaussian quadrature point on a cell-face using equation (78).

$$u(x_G, y_G, z_G) = \sum_{s=1}^{q} \omega_s p_s(x_G, y_G, z_G) \tag{78}$$

When a given flow variable has been extrapolated to a quadrature point from both sides of the cell-face, the numerical flux can be calculated using a Riemann solver. If the linear weights, γ_s, are used in place of the non-linear weights, ω_s, then the flow variables are extrapolated using equation (79).

$$u(x_G, y_G, z_G) = \sum_{s=1}^{q} \gamma_s p_s(x_G, y_G, z_G) \tag{79}$$

If the splitting technique of Shi et. al. [8] is employed to treat the presence of negative linear weights, then the flow variable extrapolation has to be performed for both groups of the non-linear weights, ω^+ and ω^-. Equation (78) now takes the form given by equation (80).

$$u^{\pm}(x_G, y_G, z_G) = \sum_{s=1}^{q} \omega_s^{\pm} p_s(x_G, y_G, z_G) \tag{80}$$

Similarly, equation (79) now takes the form of equation (81).

$$u^{\pm}(x_G, y_G, z_G) = \sum_{s=1}^{q} \gamma_s^{\pm} p_s(x_G, y_G, z_G) \tag{81}$$

If the splitting technique has been employed, then the final WENO extrapolation to the quadrature points is written as equation (82).

$$u(x_G, y_G, z_G) = \sigma^+ u^+(x_G, y_G, z_G) - \sigma^- u^-(x_G, y_G, z_G) \tag{82}$$

The above equation represents extrapolation of any variable *u* to the quadrature points on the cell-face. The same approach is applied to the extrapolation of the conserved variables to the quadrature points from either side of the cell-face. The flux through each quadrature point is then calculated. The total flux through the face is found using a weighted average of the fluxes at the quadrature points on that face by applying equation (3).

The procedures detailed in this chapter describe the process of combining linear polynomials to extrapolate the flow variables to the quadrature points. These extrapolations are used for the numerical flux calculation, producing third order spatial accuracy without a quadratic polynomial reconstruction. The numerical flux at each of the quadrature points can be calculated using the extrapolated flow variables with any Riemann solvers. The implementation described by this paper utilizes an existing application of Roe's approximate Riemann solver [12] for the flux estimations.

CHAPTER 3

VALIDATION

Local Variable Calculation

The local variables in the WENO formulation are defined by equation (5). The cell average values of these local variables are defined by equation (8). The polynomial reconstructions are dependent on these cell average local variables and the volume integrals seen in equation (8) must be evaluated for every big stencil cell. In order for numerical evaluation to be accomplished, the volume integrals must first be converted to surface integrals through the use of the Gauss Theorem [14]. The surface integrals can then be approximated using a Gaussian quadrature scheme [13]. The details of the Gaussian quadrature schemes are given in Appendix A. The procedures of converting the volume integrals to surface integrals and applying the quadrature schemes are given in Appendix B. The specific application of the quadrature schemes for the numerical calculation of the cell average values is provided in Appendix B. A tetrahedron and a cube were used to test the error of the calculated cell average values when compared to the exact analytical values. Table 1 contains the analytical values and numerical approximations for the cube's average local variables when the cube itself is given as the reference cell.

Table 1. Local Variables for a Cube Reference Cell

Function	Analytical Value	Numerical Result	Error
$\overline{\xi_0}$	0.0	-6.9389E-017	6.9389E-17
$\overline{\eta_0}$	0.0	-6.9389E-017	6.9389E-17
$\overline{\zeta_0}$	0.0	-6.9389E-017	6.9389E-17
$\overline{\xi_0^2}$	0.08333	8.3333E-002	0.0000E+00
$\overline{\eta_0^2}$	0.08333	8.3333E-002	0.0000E+00
$\overline{\zeta_0^2}$	0.08333	8.3333E-002	0.0000E+00
$\overline{(\xi\eta)_0}$	0.0	-6.9389E-018	6.9389E-18
$\overline{(\xi\zeta)_0}$	0.0	-5.2041E-018	5.2042E-18
$\overline{(\eta\zeta)_0}$	0.0	-6.9389E-018	6.9389E-18

As seen in Table 1, all numerical approximations are within an acceptable range of accuracy. Table 2 presents the numerical approximations for the cell average values when the cube is defined as cell *i* and a reference cell, other than the cube itself, is prescribed.

Table 2. Local Variables for Cube Cell *i*

Function	Analytical Value	Numerical Result	Error
$\overline{\xi_i}$	-44.5	-44.5000	0.000
$\overline{\eta_i}$	-35.8	-35.8000	9.9476E-14
$\overline{\zeta_i}$	-24.2	-24.2000	0.0000
$\overline{\xi_i^2}$	1980.3333	1980.3333	0.0000
$\overline{\eta_i^2}$	1281.7233	1281.7233	0.0000
$\overline{\zeta_i^2}$	585.7233	585.7233	0.0000
$\overline{(\xi\eta)_i}$	1593.1	1593.1000	1.0004E-11
$\overline{(\xi\zeta)_i}$	1076.9	1076.9000	0.0000
$\overline{(\eta\zeta)_i}$	866.36	866.3600	3.9790E-12

Table 3 and Table 4 show numerical approximations for the cell average values when the cube is given as the reference cell. Because the cube undergoes rotation, these tables differ from the results presented in Table 1.

Table 3. Local Variables for Cube Rotated 45° About x-axis

Function	Analytical Value	Numerical Result	Error
$\overline{\xi_0}$	0.0	-1.8041E-016	1.8041E-16
$\overline{\eta_0}$	0.0	-7.6328E-017	7.6328E-17
$\overline{\zeta_0}$	0.0	4.8572E-017	4.8572E-17
$\overline{\xi_0^2}$	0.08333	8.3333E-002	2.000E-17
$\overline{\eta_0^2}$	0.08333	8.3333E-002	0.000
$\overline{\zeta_0^2}$	0.08333	8.3333E-002	3.000E-17
$\overline{(\xi\eta)_0}$	0.0	2.3111E-033	2.3111E-33
$\overline{(\xi\zeta)_0}$	0.0	-1.0408E-017	1.0408E-17
$\overline{(\eta\zeta)_0}$	0.0	6.9389E-018	6.9389E-18

Table 4. Local Variables for Cube Rotated 86° About y-axis

Function	Analytical Value	Numerical Result	Error
$\overline{\xi_i}$	0.0	1.3878E-016	1.3878E-16
$\overline{\eta_i}$	0.0	-1.8041E-016	1.8041E-16
$\overline{\zeta_i}$	0.0	-7.1991E-017	7.1991E-17
$\overline{\xi_i^2}$	0.08333	8.3333E-002	0.0000
$\overline{\eta_i^2}$	0.08333	8.3333E-002	0.0000
$\overline{\zeta_i^2}$	0.08333	8.3333E-002	0.0000
$\overline{(\xi\eta)_i}$	0.0	-3.4005E-018	3.4005E-18
$\overline{(\xi\zeta)_i}$	0.0	1.6697E-017	1.6697E-17
$\overline{(\eta\zeta)_i}$	0.0	-3.4694E-018	3.4694E-18

Further validation was done for varying locations of the reference cell (i.e. varying the x_o, y_o, and z_o coordinates in equation (8)), various rotations and translations of the cells, and varying lengths of the cell's sides.

Linear Polynomial Extrapolation

The linear polynomials from each small stencil can be used to extrapolate any linear function to a quadrature point if given the linear function's cell average values of the 4 cells in the small stencil. The cell average value of a linear function is equivalent to the function value at the centroid of the cell. By substituting the cell average values into equations (41)-(43), the coefficients to the linear polynomial can be obtained. Further substitution of the values of each coefficient into equation (34) allows for the extrapolation to be calculated by evaluating the resulting expression at the quadrature point. A tetrahedral grid of 114 cells is used to test the linear polynomials' extrapolation accuracy for linear functions. An image of this grid is seen in Figure 4.

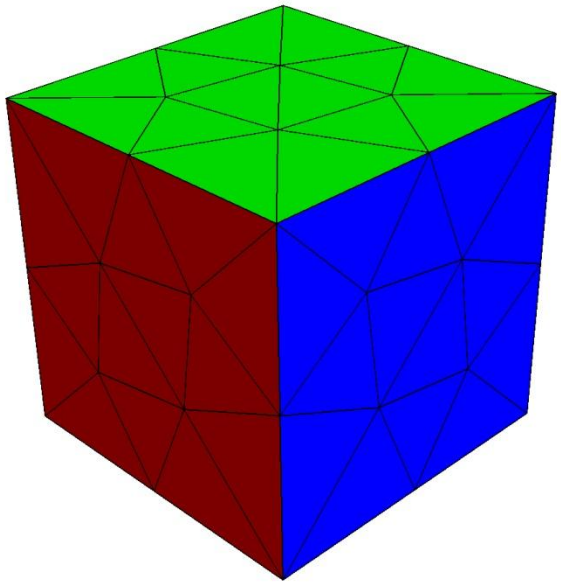

Figure 4. Tetrahedral Mesh Used for Validation

Extrapolations were performed for every cell's quadrature points using every small stencil created for that cell. The exact value used to calculate the error of the extrapolation is the function value at the quadrature point. Table 5 shows the maximum error seen out of every reconstruction in the grid for various linear functions.

Table 5. Linear Polynomial Extrapolation Error for Linear Functions

Function	Maximum Error	RMS Error
ξ	7.1342E-12	1.8464E-13
η	1.1604E-11	3.3140E-13
ζ	1.6721E-11	4.2796E-13
$0.5x + 9.7y - 2.5z + 0.9$	3.1331E-11	7.5262E-13
$10.2x - 2.9y + 5.1z + 3.6$	1.8956E-11	5.6494E-13
$-16.5x + 0.2y - 0.5z + 15.6$	2.0889E-11	6.0053E-13

The root mean square (RMS) error is defined by equation (83).

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} E_i^2}{N}} \tag{83}$$

where E_i is the error for extrapolation i, and N is the total number of extrapolations. From Table 5 it can be seen that all linear polynomials within the tetrahedral grid can extrapolate linear functions to the quadrature points with an acceptable degree of accuracy.

Quadratic Polynomial Extrapolation

The quadratic polynomials from each cell's big stencil can be used to extrapolate any quadratic function to a quadrature point when given the cell average values of the m cells in the big stencil for that function. The coefficients of the quadratic polynomial are

obtained by substituting the cell average values into equation (20). The extrapolation is calculated by substituting the values of each coefficient into equation (12) and evaluating the resulting expression at the quadrature point. Obtaining the cell average values for the quadratic functions is more difficult than for the linear functions. Simply evaluating the quadratic expression at the cell centroid will not result in the cell average value. However, a modified form of the function used for the calculation of the cell average local variables can be used to approximate the cell average value of any analytical quadratic function. The cell average value is defined by equation (6). The modified function approximates the volume integral of the quadratic expression and divides the result by the volume of the cell. This results in the approximate cell average value. This function allows the cell average values of a quadratic function to be found for each cell in the big stencil. Using the 114 cell tetrahedral grid, the quadratic polynomial from every big stencil was used to extrapolate various functions to every quadrature point of the cell corresponding to the particular big stencil. Table 6 shows the maximum deviance from the exact function value at the quadrature point for every extrapolation performed on the grid as well as the RMS error.

Table 6. Quadratic Polynomial Extrapolation Error

Function	Maximum Error	RMS Error
ξ	1.2490E-14	2.0067E-15
η	1.4655E-14	2.3126E-15
ζ	1.6098E-14	2.3358E-15
ξ^2	2.7534E-14	3.0965E-15
η^2	1.3156E-14	3.0743E-15
ζ^2	2.2510E-14	3.3317E-15
$\xi\eta$	6.3838E-15	1.1399E-15
$\xi\zeta$	1.0353E-14	1.3074E-15
$\eta\zeta$	1.0159E-14	1.4932E-15
$1.2x^2 + 4.5y^2 - 9.1z^2 + 7.2x - 1.3y + 0.9z - 1.2$	6.3061E-14	9.1829E-15
$4.1x^2 + 3.2y^2 + 6.7z^2 + 9.1x + 9.3y - 2.9z - 0.3$	1.1013E-13	1.3020E-14
$-2.1x^2 + 1.7y^2 - 6.9z^2 + 3.2x - 6.2y - 8.9z - 10.5$	9.2371E-14	1.1513E-14

The results in Table 6 show that the quadratic polynomials can perform accurate second order extrapolations to the quadrature points.

Extrapolation Using the Linear Weights

The previous sections showed that the linear polynomials and quadratic polynomials can be used for extrapolations of first and second order accuracy, respectively. The validation in the previous sections gives assurance that the inputs into the linear weight calculation are correct because they depend on the linear polynomial extrapolations, the small stencil constants, and the big stencil constants. The linear weights can now be tested in order to validate their replication of the quadratic polynomial's accuracy when used to combine the linear polynomials. The linear polynomials are used to extrapolate quadratic functions to the quadrature points. The linear weights are then applied to these extrapolations using equation (79) to get the final

value of the quadrature point extrapolation. Table 7 provides the error values of the extrapolations obtained from the linear weight application.

Table 7. Linear Weight Extrapolation Error

Function	Maximum Error	RMS Error
ξ	1.2297E-12	3.2632E-14
η	1.8682E-12	4.7618E-14
ζ	1.2037E-12	4.4819E-14
ξ^2	2.9554E-13	1.3115E-14
η^2	2.1139E-13	1.0754E-14
ζ^2	5.2261E-13	1.7451E-14
$\xi\eta$	2.7192E-13	8.6011E-15
$\xi\zeta$	5.1348E-13	1.4156E-14
$\eta\zeta$	1.4921E-13	7.7182E-15
$1.2x^2 + 4.5y^2 - 9.1z^2 + 7.2x - 1.3y + 0.9z - 1.2$	1.7024E-12	9.0109E-14
$4.1x^2 + 3.2y^2 + 6.7z^2 + 9.1.2x + 9.3y - 2.9z - 0.3$	1.1141E-11	3.1623E-13
$-2.1x^2 + 1.7y^2 - 6.9z^2 + 3.2x - 6.2y - 8.9z - 10.5$	7.3008E-12	2.2564E-13

The results in Table 7 show the linear weights' ability to combine linear polynomials and mimic the accuracy of quadratic polynomials. This is key to the WENO scheme, achieving third order spatial accuracy while only having to deal with the computational cost of linear polynomials. The results shown in Table 7 were obtained using extrapolations calculated from equation (79). This equation uses the linear weights before the splitting technique, given by equations (72)-(75), has been applied. The splitting technique is used to treat the presence of negative linear weights which cause instability during the application of the WENO scheme to CFD problems [8]. Each set of linear weights is checked for the presence of negative values. If negative linear weights are present, then the splitting technique is applied. The extrapolation to the quadrature points can then be calculated by equations (81) and (82). Table 8 shows the error values

obtained in the same fashion as Table 7, however, the linear weights were split if the presence of negative weighting was detected.

Table 8. Split Linear Weight Extrapolation Error

Function	Maximum Error	RMS Error
ξ	1.2202E-12	3.4481E-14
η	1.8631E-12	4.7598E-14
ζ	1.2047E-12	4.9693E-14
ξ^2	3.4583E-13	1.4364E-14
η^2	2.0831E-13	1.2745E-14
ζ^2	5.2083E-13	1.6831E-14
$\xi\eta$	2.9324E-13	1.1091E-14
$\xi\zeta$	7.4662E-13	1.9719E-14
$\eta\zeta$	5.1700E-13	1.6454E-14
$1.2x^2 + 4.5y^2 - 9.1z^2 + 7.2x - 1.3y + 0.9z - 1.2$	1.7039E-12	9.6437E-14
$4.1x^2 + 3.2y^2 + 6.7z^2 + 9.1.2x + 9.3y - 2.9z - 0.3$	1.0637E-11	4.4357E-13
$-2.1x^2 + 1.7y^2 - 6.9z^2 + 3.2x - 6.2y - 8.9z - 10.5$	9.4076E-12	4.0714E-13

Table 8 shows that the splitting technique of [8] maintains the same level of accuracy as the non-split weights that were originally calculated. This allows for the accuracy of the scheme to be maintained while also maintaining stability during application. The error analysis from the quadratic function extrapolations obtained using the quadratic polynomials from the big stencils, the linear weights combining the linear polynomials, and the split linear weights combining the linear polynomials is consolidated in Table 9.

.

Table 9. Comparison of Extrapolation Methods

Function	Max Error			RMS Error		
	Quadratic Polynomial	Linear Weight Combinations	Split Linear Weights	Quadratic	Linear Weight Combinations	Split Weight Linear
ξ	1.2490E-14	1.2297E-12	1.2202E-12	2.0067E-15	3.2632E-14	3.4481E-14
η	1.4655E-14	1.8682E-12	1.8631E-12	2.3126E-15	4.7618E-14	4.7598E-14
ζ	1.6098E-14	1.2037E-12	1.2047E-12	2.3358E-15	4.4819E-14	4.9693E-14
ξ^2	2.7534E-14	2.9554E-13	3.4583E-13	3.0965E-15	1.3115E-14	1.4364E-14
η^2	1.3156E-14	2.1139E-13	2.0831E-13	3.0743E-15	1.0754E-14	1.2745E-14
ζ^2	2.2510E-14	5.2261E-13	5.2083E-13	3.3317E-15	1.7451E-14	1.6831E-14
$\xi\eta$	6.3838E-15	2.7192E-13	2.9324E-13	1.1399E-15	8.6011E-15	1.1091E-14
$\xi\zeta$	1.0353E-14	5.1348E-13	7.4662E-13	1.3074E-15	1.4156E-14	1.9719E-14
$\eta\zeta$	1.0159E-14	1.4921E-13	5.1700E-13	1.4932E-15	7.7182E-15	1.6454E-14
$1.2x^2 + 4.5y^2 - 9.1z^2 + 7.2x - 1.3y + 0.9z - 1.2$	6.3061E-14	1.7024E-12	1.7039E-12	9.1829E-15	9.0109E-14	9.6437E-14
$4.1x^2 + 3.2y^2 + 6.7z^2 + 9.1.2x + 9.3y - 2.9z - 0.3$	1.1013E-13	1.1141E-11	1.0637E-11	1.3020E-14	3.1623E-13	4.4357E-13
$-2.1x^2 + 1.7y^2 - 6.9z^2 + 3.2x - 6.2y - 8.9z - 10.5$	9.2371E-14	7.3008E-12	9.4076E-12	1.1513E-14	2.2564E-13	4.0714E-13

Additional Extrapolation Results

This section provides the quadrature point extrapolation results obtained for a tetrahedral unstructured mesh consisting of 553,409 cells, 138,584 nodes, and 1,198,096 faces. This is the same mesh that is used in the computational cost analysis and the vortex convection case that are presented in Chapter 4. Figure 5 and Figure 6 show this tetrahedral mesh.

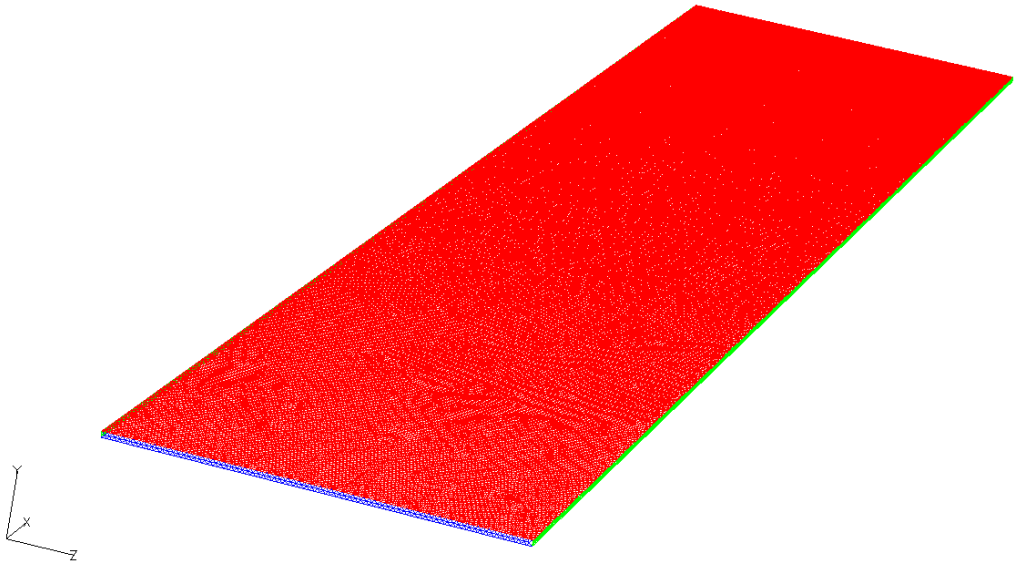

Figure 5. Unstructured Tetrahedral Mesh

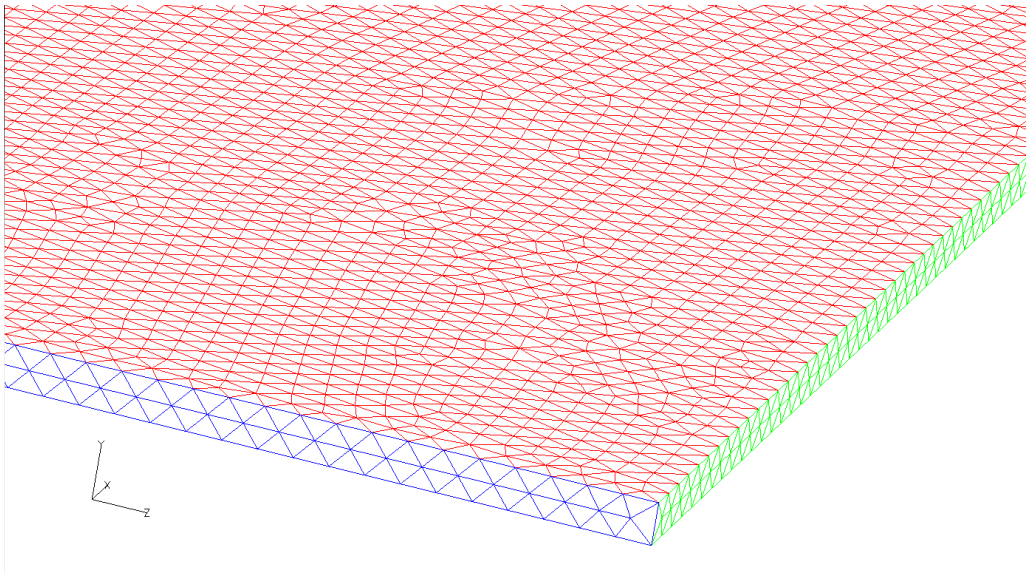

Figure 6. Close Up of Unstructured Tetrahedral Mesh

This mesh is different than the mesh shown in Figure 4 in terms of number of cells, nodes, and faces, as well as average cell volume. The same local variables and analytical functions from the previous sections are used to analyze the extrapolation error. The maximum error obtained for the extrapolated value and the RMS error, as defined by equation (83), are used to evaluate the accuracy. Table 10 presents the extrapolation error for the linear local variables and linear analytical functions when the linear polynomial reconstructions from the small stencils are used.

Table 10. Linear Extrapolation Errors for Unstructured Grid

Function	Maximum Error	RMS Error
ξ	4.1303E-10	5.3712E-13
η	1.08E-09	2.4111E-13
ζ	4.39E-09	7.9505E-13
$0.5x + 9.7y - 2.5z + 0.9$	7.4177E-10	1.8099E-13
$10.2x - 2.9y + 5.1z + 3.6$	1.0408E-09	4.6183E-13
$-16.5x + 0.2y - 0.5z + 15.6$	3.44E-10	6.4861E-13

The results of the linear extrapolations presented in Table 10 are less accurate than the results presented in Table 5. This could be due to the increased complexity and size of the unstructured grid used, shown in Figure 5, as compared to the grid shown in Figure 4. Table 11 provides the extrapolation results for all of the local variables as well as quadratic analytical functions. The extrapolations in Table 11 are performed using the quadratic polynomial from the big stencil, the linear weights, and the linear weights using the splitting technique.

Table 11. Quadratic Extrapolation Errors for Unstructured Grid

Function	Max Error			RMS Error		
	Quadratic Polynomial	Linear Weight Combinations	Split Linear Weights	Quadratic	Linear Weight Combinations	Split Weight Linear
ξ	3.5980E-12	1.7733E-07	1.7689E-07	5.1141E-15	6.0110E-11	6.1130E-11
η	4.6252E-12	1.3539E-07	1.4310E-07	5.0644E-16	4.5709E-11	4.8242E-11
ζ	6.0056E-12	1.3611E-07	1.3551E-07	6.0412E-15	4.5790E-11	4.5548E-11
ξ^2	1.2958E-11	8.9952E-09	1.1363E-08	1.4689E-14	4.3311E-12	5.1843E-12
η^2	2.2032E-12	6.4059E-08	6.4735E-08	3.1127E-15	2.1834E-11	2.2155E-11
ζ^2	1.4736E-11	6.0930E-09	7.3451E-09	1.5557E-14	2.8671E-12	3.2615E-12
$\xi\eta$	1.1326E-12	2.5569E-08	2.4835E-08	1.4306E-15	8.6741E-12	8.3796E-12
$\xi\zeta$	3.3315E-12	5.3126E-08	5.4569E-08	3.9212E-15	1.7955E-11	1.8512E-11
$\eta\zeta$	1.1416E-12	2.5484E-08	2.2515E-08	1.6292E-15	8.6876E-12	7.5947E-12
$1.2x^2 + 4.5y^2 - 9.1z^2 + 7.2x - 1.3y + 0.9z - 1.2$	2.9787E-08	7.7582E-07	3.9846E-06	2.5435E-10	4.9888E-10	1.4693E-09
$4.1x^2 + 3.2y^2 + 6.7z^2 + 9.1.2x + 9.3y - 2.9z - 0.3$	1.0503E-07	8.1708E-06	1.0429E-05	9.1710E-10	3.0819E-09	4.0381E-09
$-2.1x^2 + 1.7y^2 - 6.9z^2 + 3.2x - 6.2y - 8.9z - 10.5$	5.4098E-08	1.0566E-06	4.7015E-06	4.7533E-10	7.4366E-10	2.0725E-09

More error is seen in results of Table 11 than the corresponding results for the less complex grid that are given in Table 9. The WENO scheme is geometry dependent, so great care must be taken when a new mesh is being used to ensure that the following components have been correctly formed: small stencil polynomials, big stencil polynomials, and linear weights. Test routines have been written for all of these elements making up the WENO scheme, and they can be utilized before a CFD simulation is attempted on a new computational domain.

CHAPTER 4

RESULTS AND DISCUSSION

Computational Cost Analysis

The additional computational expense caused by increasing the spatial order of accuracy through the use of the WENO scheme is examined in this section. This analysis was performed using a vortex convection case on two different grids. One grid used is the unstructured tetrahedral mesh seen in Figure 5 and Figure 6 while the other grid is a structured hexahedral mesh containing 64,000 cells, 97,443 nodes, and 224,960 faces. Figure 7 and Figure 8 show the hexahedral structured mesh.

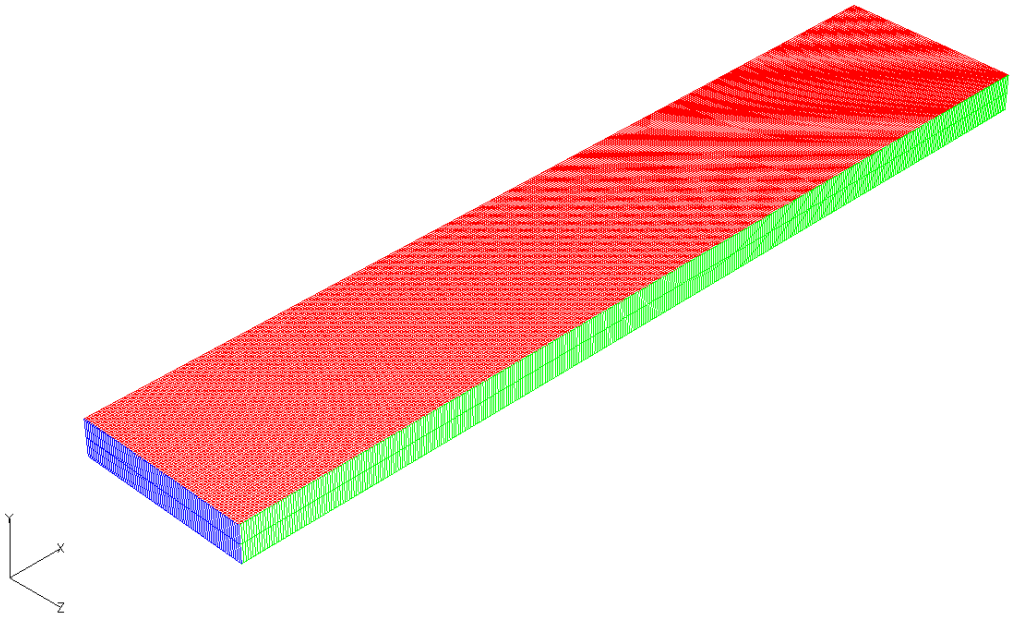

Figure 7. Structured Hexahedral Mesh

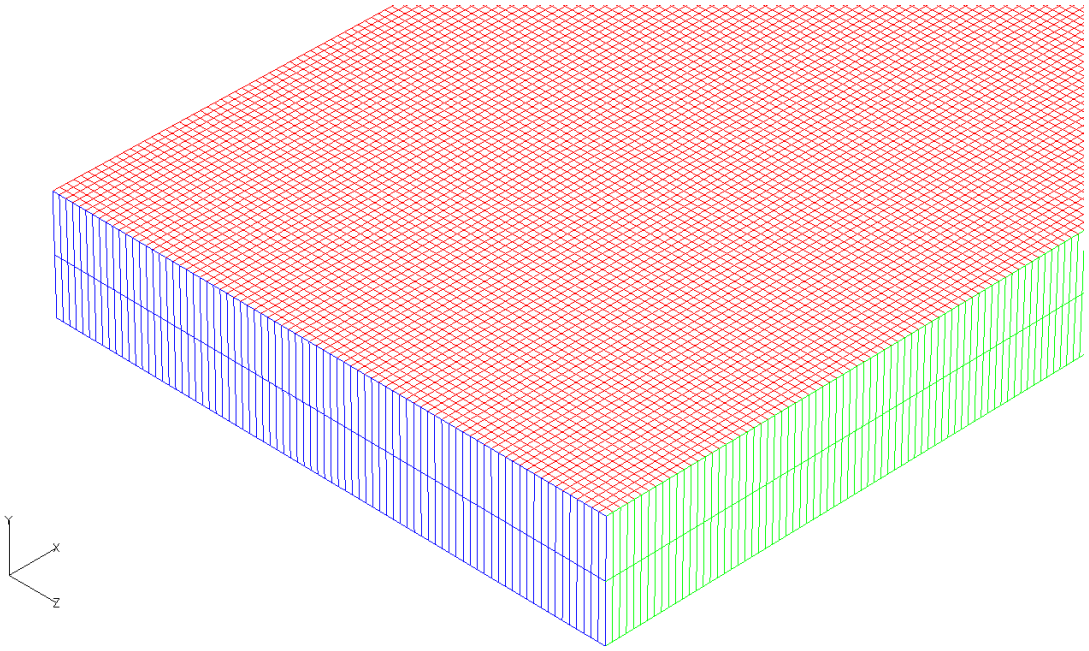

Figure 8. Close Up of Structured Hexahedral Mesh

The computational cost analysis will include the cost associated with first and second order schemes. In addition, a second order accurate scheme utilizing the Gaussian quadrature schemes and linear extrapolations of the flow variables will be analyzed. The unstructured mesh will also contain a cost comparison for the third order accurate WENO scheme. Difficulties are encountered in calculating the linear weights on the structured mesh and a third order scheme will not be included in the cost analysis. Due to the inability to use the non-linear weights needed for the third order scheme, the solution of this vortex convection case diverges after 18 iterations when using the unmodified linear weights' extrapolations alone for the flux calculations. Therefore, the full CFD simulation results cannot be obtained for an accuracy comparison of the WENO scheme, but the increase in computational cost can be measured. The difference in these 4 schemes is the way in which the flow variables are extrapolated to the face for flux calculation. The first order scheme assumes a constant flow variable distribution across

each cell and takes the flow variable value at the centroids of the cells on either side of the face for the flux calculations. The second order scheme uses linear extrapolations of the flow variables from the cell centroids from either side of the face to the centroid of the face for the flux calculation. The second order scheme utilizing the quadrature points uses linear extrapolations of the flow variables from the cell centroids on either side of the face to each of the 4 quadrature points on that face. The flux is calculated at each of these quadrature points based on the linearly extrapolated values. The total flux through the face is then calculated as a weighted sum of the four fluxes as seen in equation (3). The third order scheme is the WENO scheme described in this paper, and it obtains the accuracy equivalent to that obtained by a quadratic polynomial extrapolation through the use of a weighted sum of linear extrapolations as defined by equation (81) and equation (82). The results presented in this section compare the computational time required for the flux calculation for each method based on the average time of the first 9 times the flux subroutine was called (well before the solution diverged in the third order case). Table 12 provides data obtained for the average flux calculation time when the simulation was run on the unstructured mesh.

Table 12. Flux Calculation Time for Unstructured Grid

Spatial Order of Accuracy	Time per Flux Calculation per cell (s)	Time per Iteration per cell (s)
First Order	4.96E-07	5.32E-05
Second Order	1.02E-06	5.70E-05
Second Order with Quadrature Scheme	2.00E-06	5.71E-05
Third Order WENO Scheme	2.67112E-05	1.26E-04

The increase of the computational time required with increasing order of accuracy shown in Table 12 reflects the increased number of operations in the extrapolation of the flow

variables when the order of accuracy is increased. The relatively large increase in computational time of the third order WENO scheme when compared to the other schemes can be attributed to the procedure of obtaining the flow variable extrapolations from numerous linear extrapolations that must be combined using the linear weights as governed by equation (81) and equation (82). Table 13 provides the computational cost data that was obtained using the structured grid.

Table 13. Flux Calculation Time for Structured Grid

Spatial Order of Accuracy	Time per Flux Calculation per cell (s)	Time per Iteration per cell (s)
First Order	4.34E-07	3.64E-05
Second Order	7.12E-07	3.92E-05
Second Order with Quadrature Scheme	2.07E-06	4.28E-05

Similar patterns of increasing computational cost with increasing order of accuracy can be seen in Table 13.

Vortex Convection Case

The structured and unstructured meshes and the flux calculation schemes described in the computational cost study were used to form vortex convection simulations. This case was intended to be used for the validation of the third order accurate WENO scheme by showcasing its ability to accurately propagate the vortex along the flow domain while exhibiting less dissipation when compared to the second and first order schemes. However, the current implementation cannot produce the non-linear weight modifications to the linear weights that can maintain the third order accuracy while providing numerical stability to the simulation. The solution diverges when the unmodified linear weights are used for the flow variable extrapolations during the flux

calculation. Therefore, the full WENO scheme cannot be currently validated with this vortex convection case; however, the Gaussian quadrature schemes used by the WENO scheme in the flux calculation can be examined. These Gaussian quadrature schemes have been used to create a second order scheme by using linear extrapolations from the cell centroids on either side of a face to the 4 quadrature points on that face, as described in the previous section. The vortex was prescribed to propagate in the positive x-direction for both the tetrahedral unstructured mesh and the hexahedral structured mesh. The center of the vortex was determined to be the point at which the minimum values of pressure and density were located within the domain. All of the simulations had extremely close results when comparing the propagation of the vortex throughout the flow domain with respect to location and time. However, the dissipation of the vortex proved to be greater for the first order scheme when compared to the two second order schemes. Figure 9 shows the value of the minimum pressure within the vortex as the simulation progressed, while Figure 10 shows the value of the minimum density within the vortex.

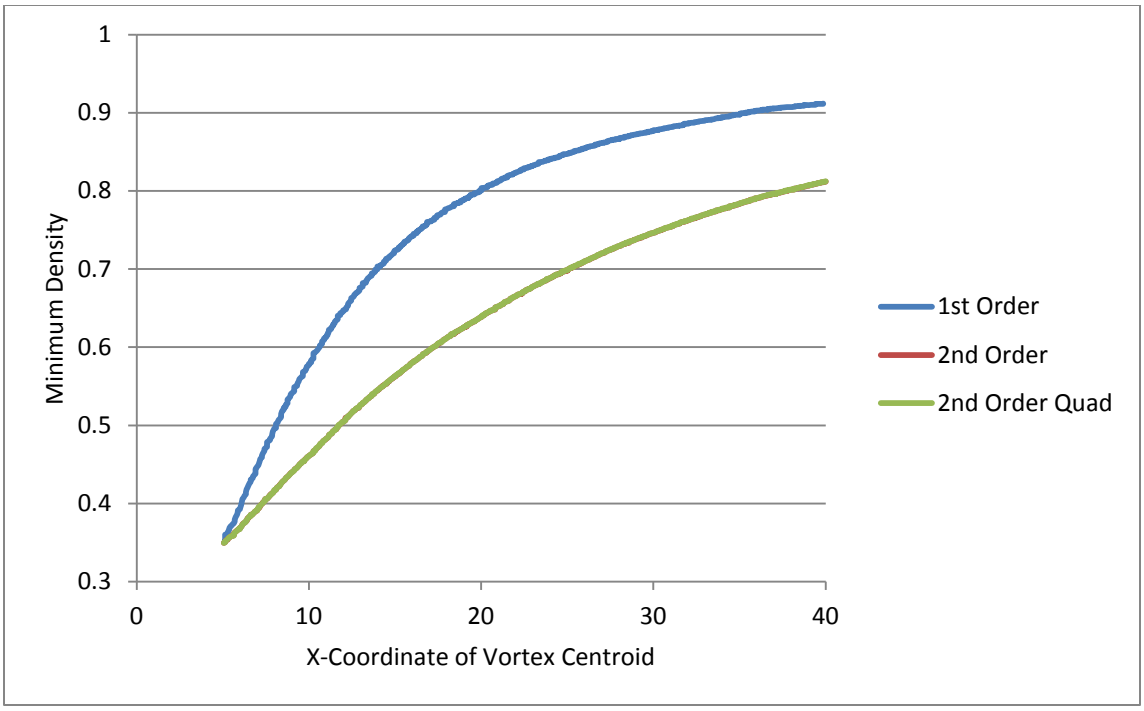

Figure 9. Minimum Pressure of Unstructured Grid

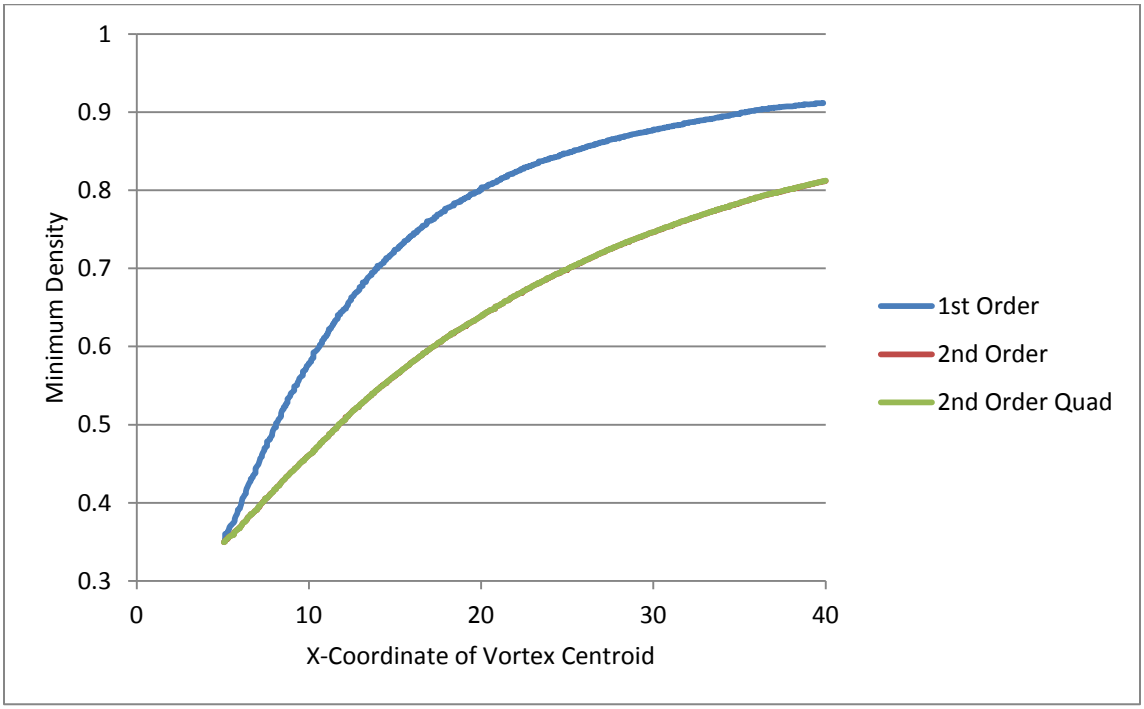

Figure 10. Minimum Density of Unstructured Grid

The results presented in the above figures show that both second order accurate schemes show significantly less dissipation than the first order accurate scheme for the simulations

obtained using the unstructured domain. The second order scheme that utilizes the quadrature point flux calculation showed negligible difference from the solution obtained using the existing HYB3D second order scheme. Figure 11 and Figure 12 provide the data for minimum pressure and density values, respectively, for the simulations obtained using structured mesh.

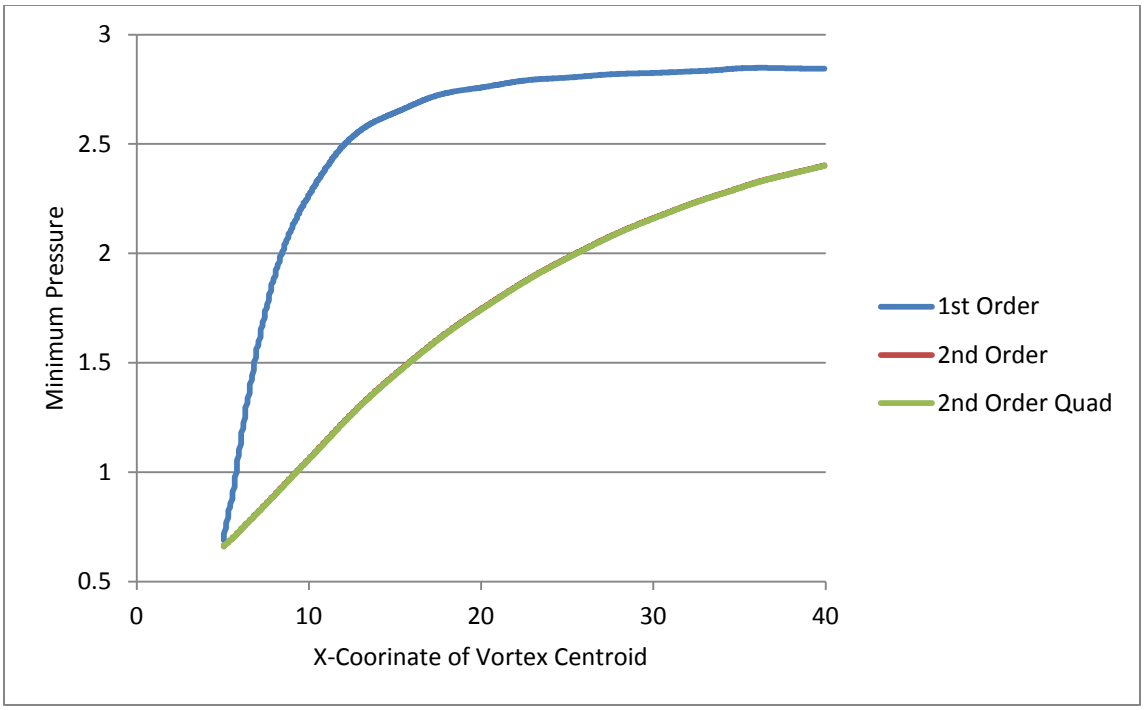

Figure 11. Minimum Pressure of Structured Grid

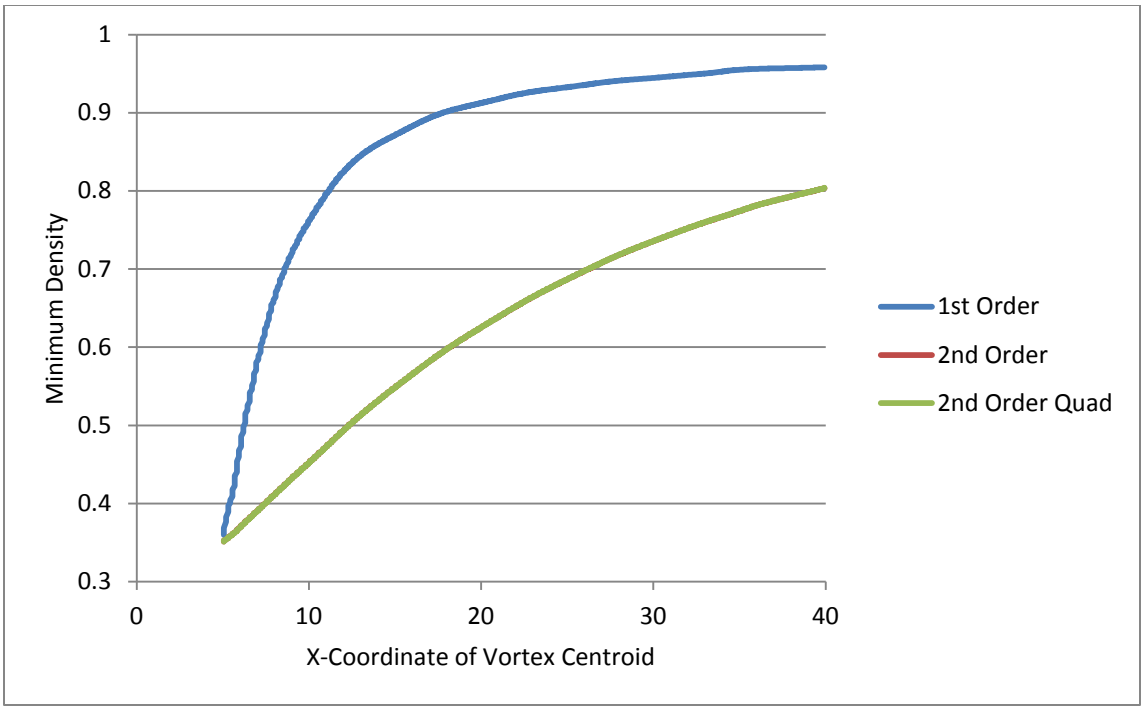

Figure 12. Minimum Density of Structured Grid

The results seen in the two figures above differ from the results of the unstructured mesh's simulations because of the larger dissipation exhibited by the first order scheme. The structured mesh contained less elements compared to the unstructured mesh. This meant that the cells of the structured mesh were larger in terms of volume when compared to the unstructured elements. Therefore, the assumption of a constant flow variable distribution over each cell used by the first order scheme is more erroneous for the structured grid than for the unstructured grid. This is why more dissipation was obtained in the structured mesh's simulation when compared to the unstructured mesh's simulation when using the first order scheme. When compared to the first order scheme, less dissipation was seen in the second order schemes' simulations due to their higher order of accuracy, which was obtained by assuming a linear flow variable distribution over each cell.

The results presented in the following graphs were obtained by finding the cell containing the center of the vortex at t = 17.5. This is the time level corresponding to half of the total time for which the simulation was run. The density and pressure values were recorded at every time step of the simulation for the cell. This allows for the before, during, and after effects of the vortex propagation through this cell to be analyzed. Figure 13 and Figure 14 show the results obtained for the unstructured mesh.

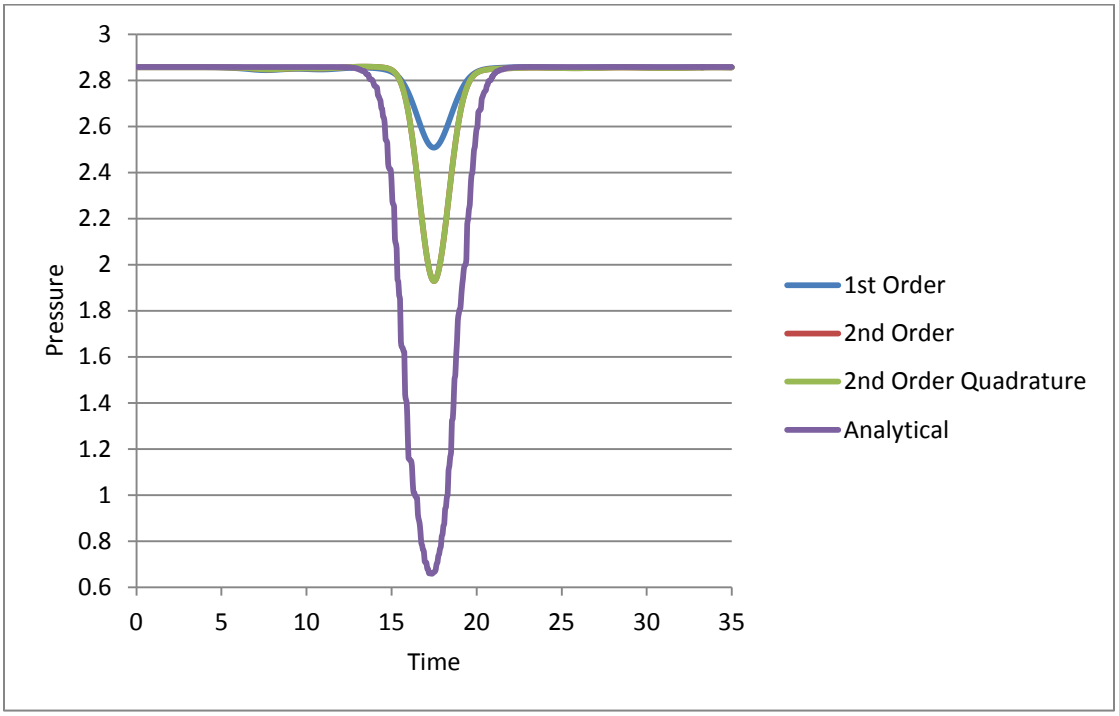

Figure 13. Pressure at Middle Cell of Unstructured Grid

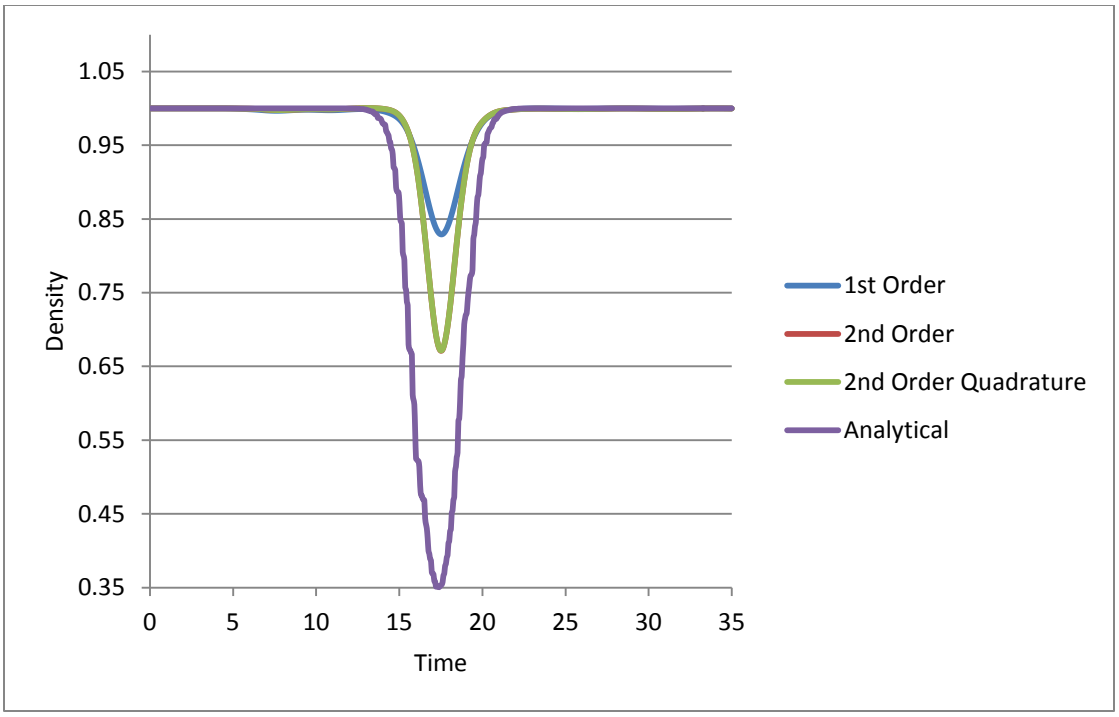

Figure 14. Density at Middle Cell of Unstructured Grid

The results from the previous figures show larger dissipation for the first order scheme when compared to both of the second order schemes. The simulations obtained by the second order schemes are so close in the predicted values for pressure and density that no differences can be observed in the previous figures. However, while the difference between the second order scheme and the second order scheme utilizing Gaussian Quadrature cannot be visually seen in the plots, a direct comparison of the numerical values at each time level does produce very small differences for the predicted values of pressure and density between the two schemes. Figure 15 and Figure 16 present the results obtained for the structured grid.

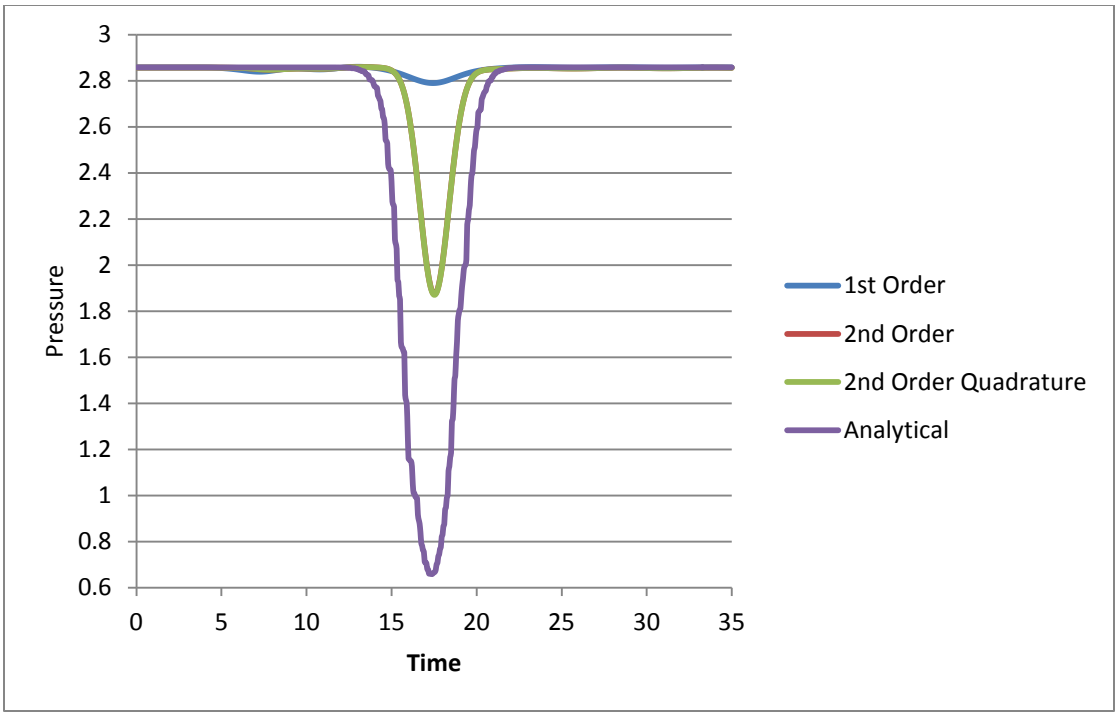

Figure 15. Pressure at Middle Cell of Structured Grid

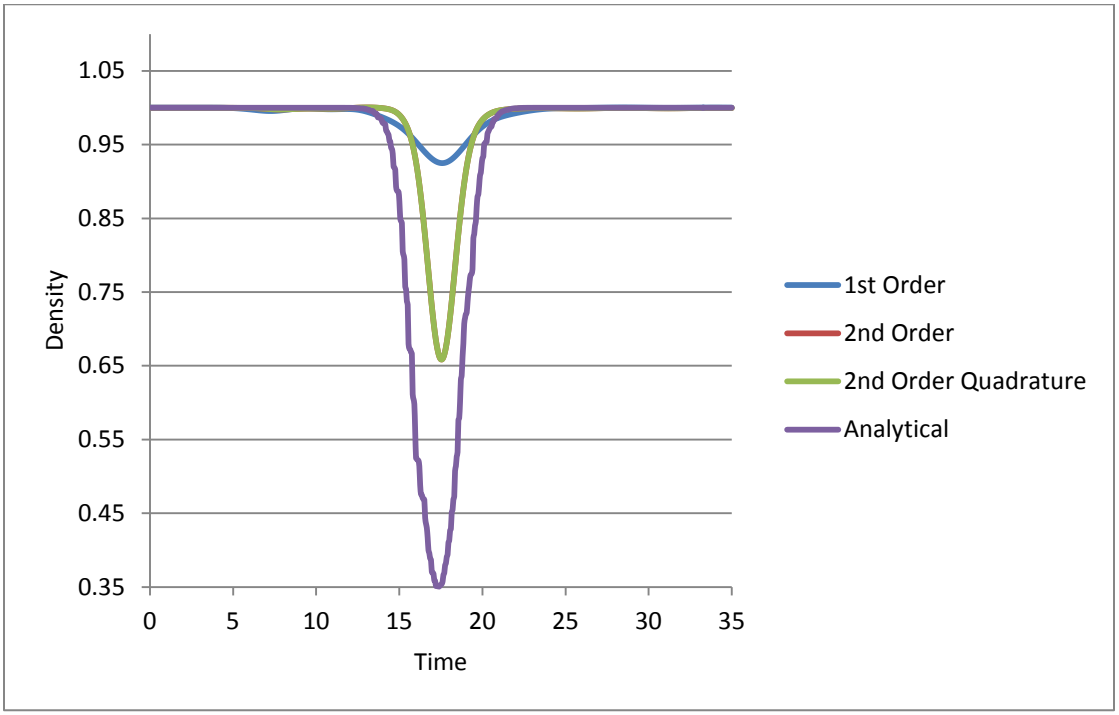

Figure 16. Density at Middle Cell of Structured Grid

Similar to the results obtained for minimum pressure and density values, more dissipation is seen in the simulation using the first order scheme for the structured grid than corresponding scheme for the unstructured grid. The structured grid simulations for the second order schemes are so close in predicted values for pressure and density that no clear deviation can be observed in the plots.

Visualizations of these simulations were created by plotting the distributions of density, pressure, and velocity magnitude at the initialization of the flow variables, t = 0, and at t = 35, after the vortex had propagated most of the length of the flow domain. These visualizations were done for all 6 of the vortex convection simulations. Figure 17 provides the initial density distribution of the vortex, Figure 18 provides the initial pressure distribution of the vortex, and Figure 19 provides the initial velocity magnitude of the vortex. These initial conditions are true of all 6 vortex convection simulations studied in this section.

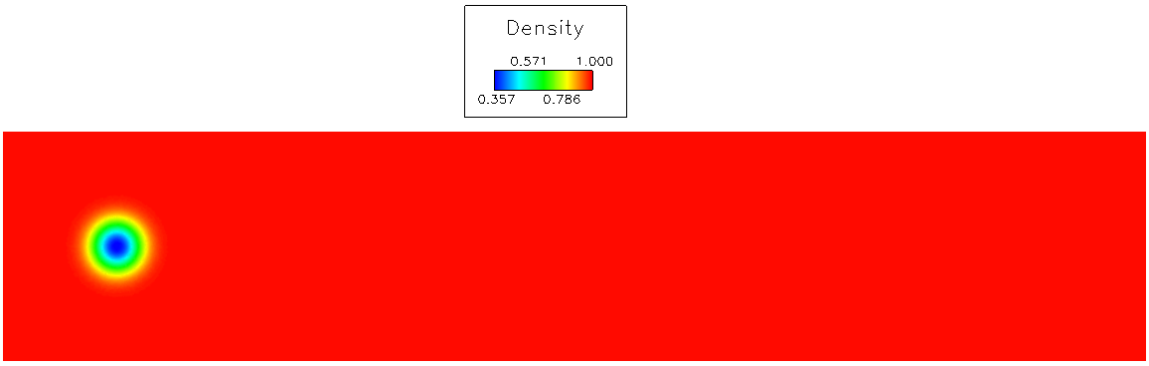

Figure 17. Density Distribution of Vortex at t = 0

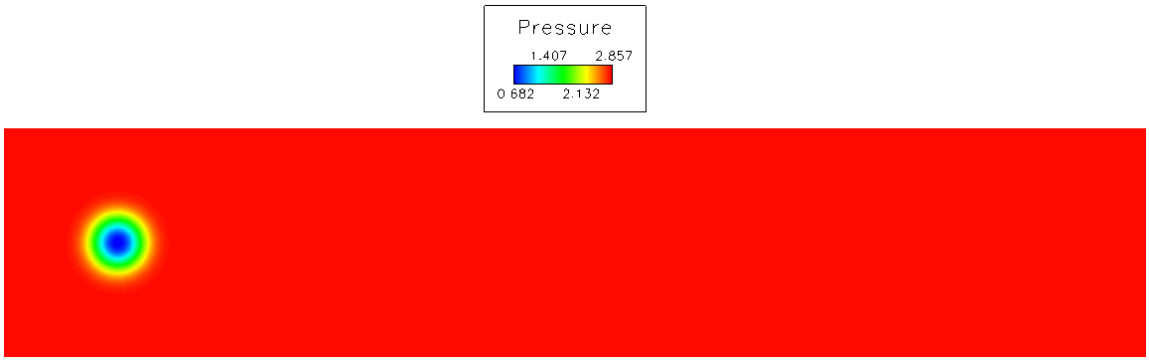

Figure 18. Pressure Distribution of Vortex at t = 0

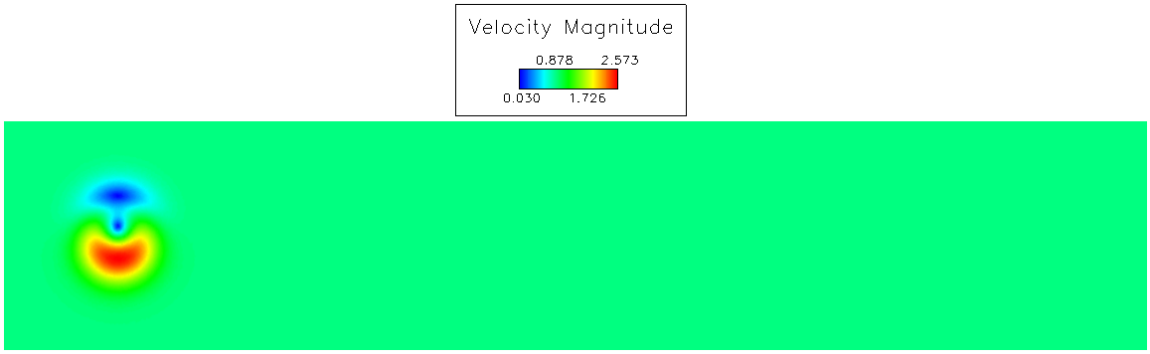

Figure 19. Velocity Magnitude Distribution of Vortex at t = 0

The vortex was allowed to propagate along the flow domain from the initial location of x = 5.0, at t = 0, to x = 40.0, at t = 35. The following figures show the distributions of density, pressure, and velocity magnitude at the time level t = 35. In each of the figures, the results for the first order scheme are presented as the top distribution, the results for the second order scheme are presented as the middle distribution, and the results from the second order Gaussian quadrature point scheme are presented as the bottom distribution. This was done for an easy side-by-side comparison, and all of the plots use the same color spectrum and legend values for each variable. Figure 20, Figure 21, and Figure 22 show the results for the unstructured grid, while Figure 23, Figure 24, and Figure 25 show the results for the structured grid.

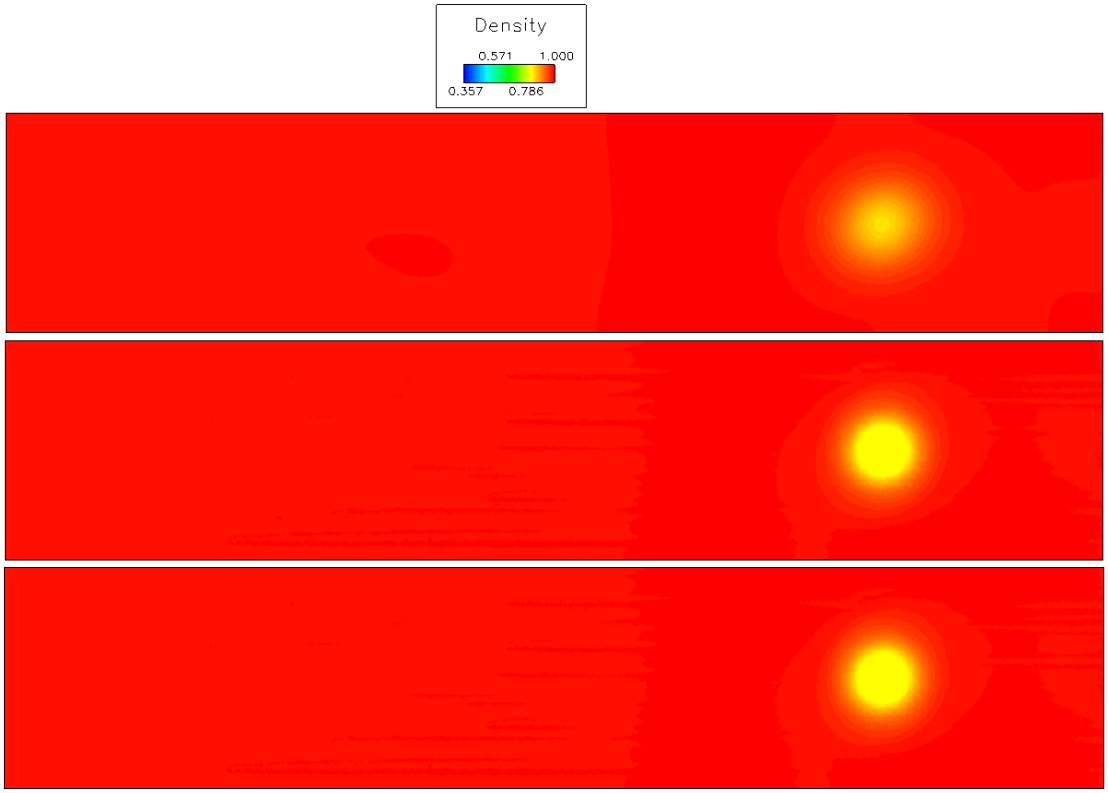

Figure 20. Density Distrubtions at t = 35 for Unstructured Grid Simulations

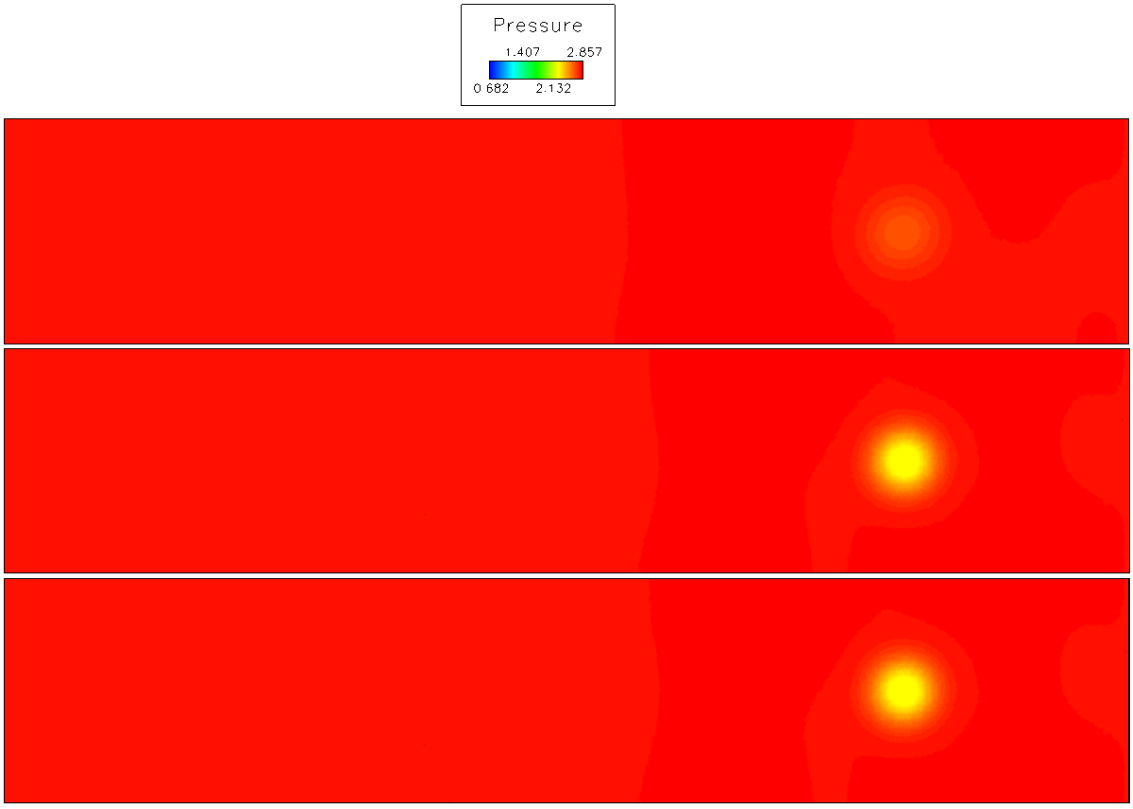

Figure 21. Pressure Distrubtions at t = 35 for Unstructured Grid Simulations

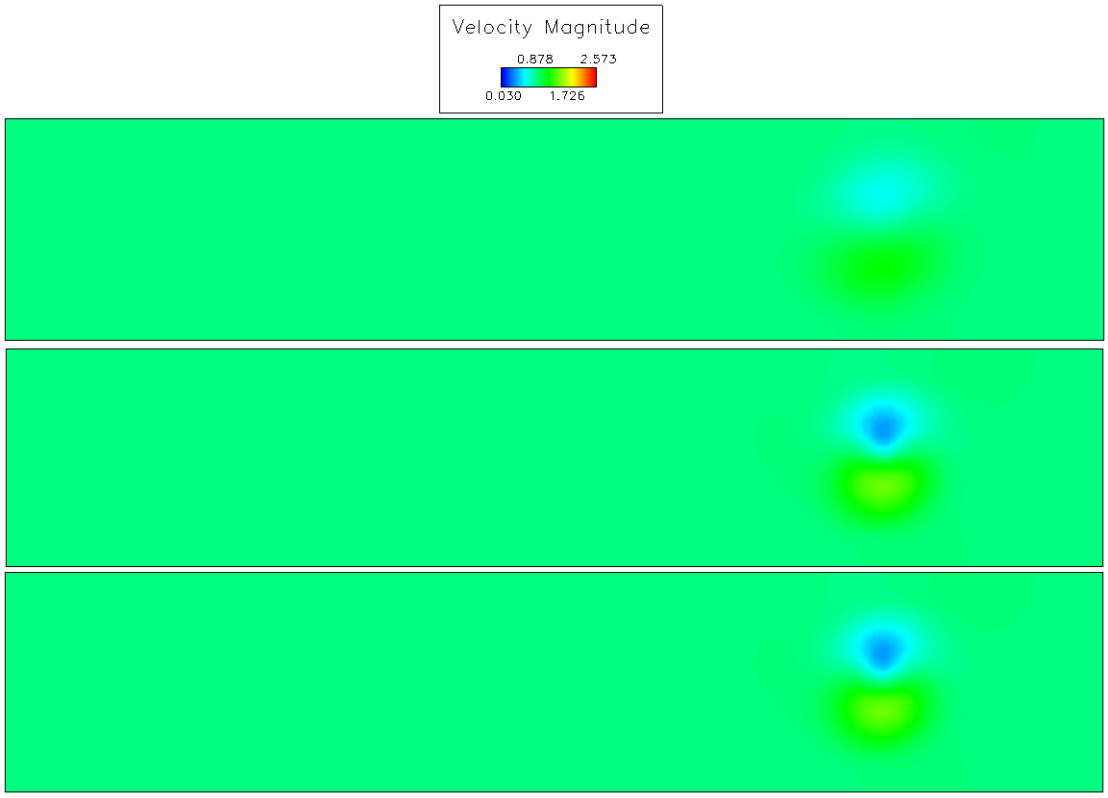

Figure 22. Velocity Magnitude Distrubtions at t = 35 for Unstructured Grid Simulations

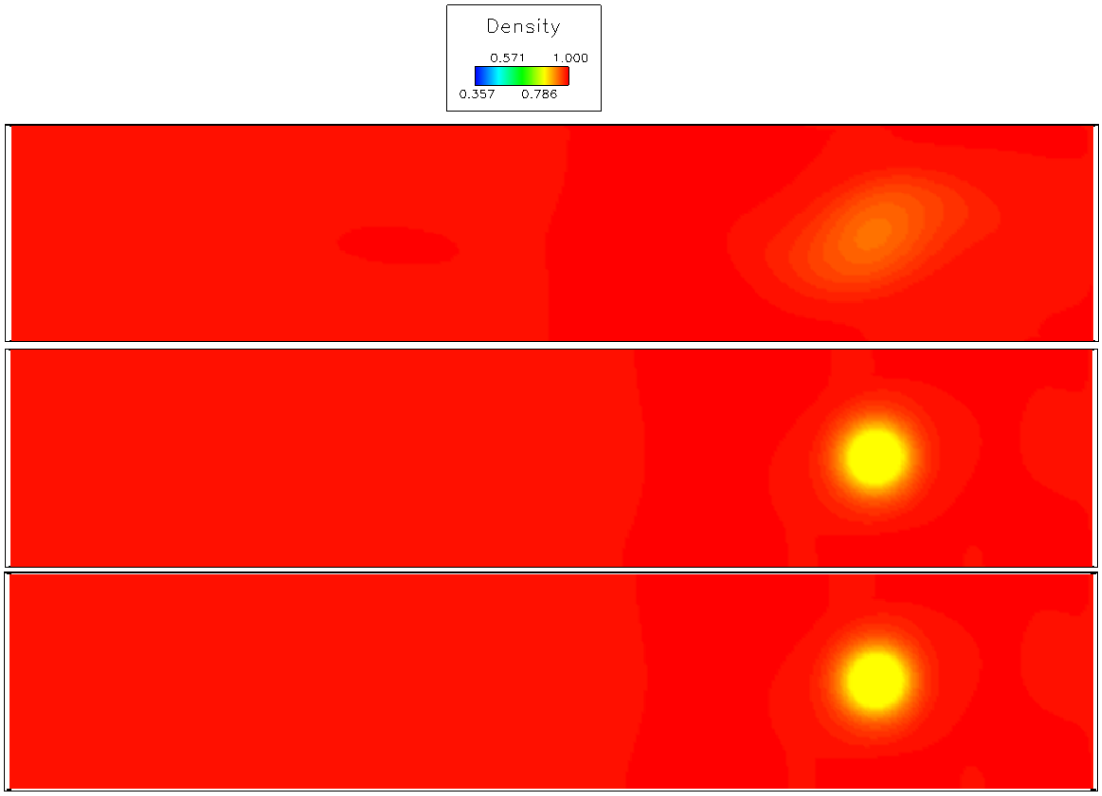

Figure 23. Density Distrubtions at t = 35 for Structured Grid Simulations

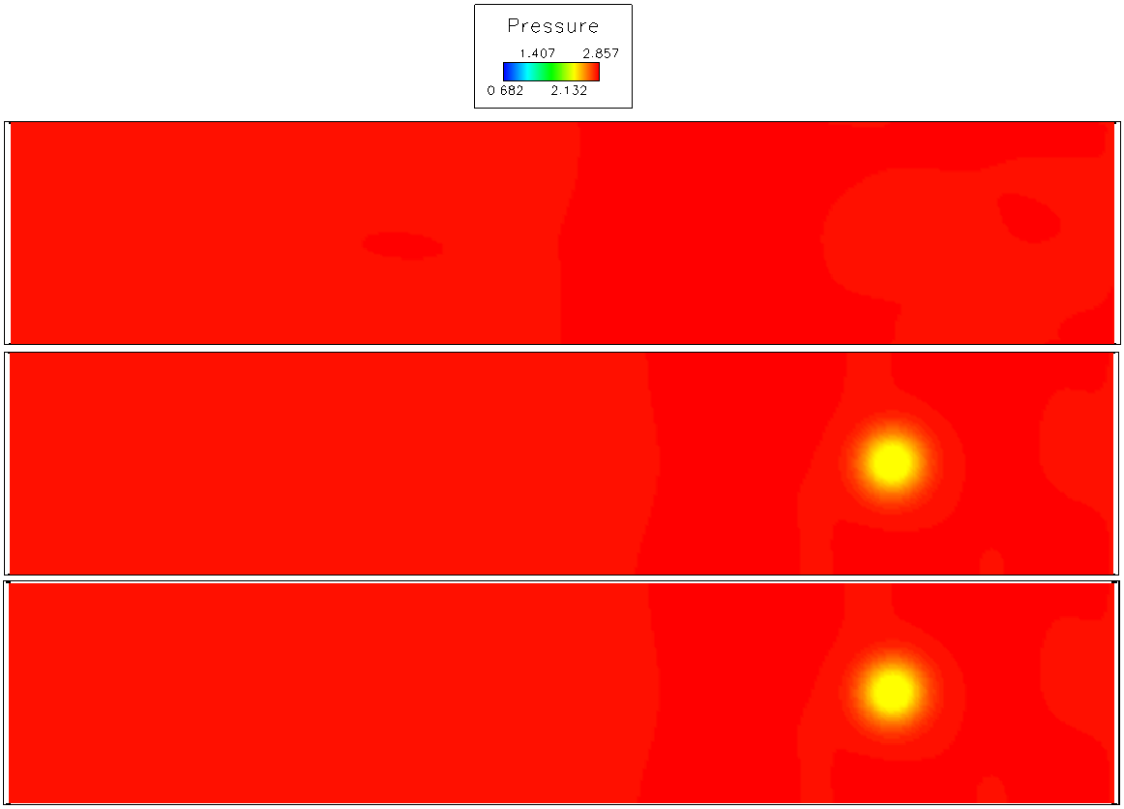

Figure 24. Pressure Distrubtions at t = 35 for Structured Grid Simulations

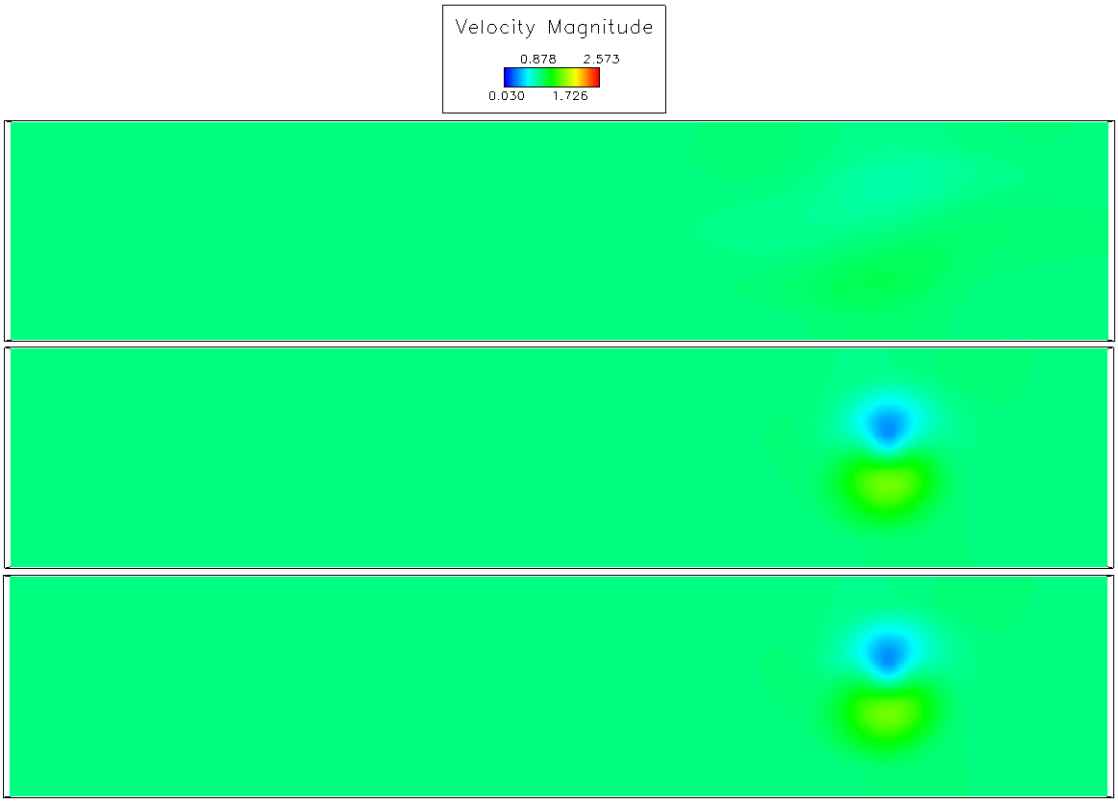

Figure 25. Velocity Magnitude Distrubtions at t = 35 for Structured Grid Simulations

The figures above indicate that second order schemes show less dissipation of the vortex when compared to the first order scheme. While the second order scheme that used Gaussian quadrature does not show a significant increase in accuracy, it does match the existing second order scheme with a negligible difference. This is true of both the Gaussian quadrature point schemes described in this paper as the structured mesh contained only quadrilateral faces while the unstructured mesh contained only triangular faces. The Gaussian quadrature points were successful in generating second order accurate results when used with linear extrapolations. Assuming that this would extend to third order accurate results obtained from quadratic extrapolations, these schemes are suitable for the WENO implementation.

Conclusions

This paper details the groundwork completed for generating a functional WENO scheme that can use generalized meshes of 4 to 6 sided elements. Implementations have been created for the stencil selections, linear polynomial reconstructions, quadratic polynomial reconstructions, and reconstructions using the linear weights. These implementations are validated using unstructured tetrahedral meshes. In addition, the linear weights splitting technique of [8] has been implemented and validated. Structured hexahedral mesh are more difficult for polynomial reconstruction and, consequently, linear weights calculation. This difficulty could be encountered because the structured mesh has little geometric variation when comparing neighboring cells' centroids. It is possible that a rotation of the entire grid with respect to the coordinate system would alleviate some of the numerical issues encountered. This rotation would eliminate values

of zero produced for the cell average local variables. The WENO scheme is dependent on the geometry of the mesh for which it is being applied. Test routines are in place to ensure proper stencil selection, linear polynomial formation, quadratic polynomial formation, and linear weight calculation. These test routines should be run anytime that a new mesh is being used with the WENO scheme. This will ensure that the mesh is suitable for numerical simulation.

CHAPTER 5

FUTURE WORK

Non-Linear Weight Implementation

The non-linear weights are currently producing large errors when used to modify the linear weights for extrapolations to the quadrature points. This modification is extremely important to the WENO scheme because it provides more weight to the small stencils that exhibit smaller variations in the flow variable distributions. This allows spurious oscillations to be avoided and provides stability to the scheme during the numerical flux calculations. For the vortex convection case, the solution diverges when the unmodified linear weights are used in extrapolating the flow variables to the quadrature points for the flux calculation. The non-linear weights provide the non-oscillatory nature to the WENO scheme, which is the basis of the originally developed ENO schemes. Before a full validation of the WENO scheme detailed in this paper can be obtained, the non-linear weights must be implemented in such a way that they correctly perform their function. The current implementation agrees with the theoretical formulation as given by [6]. The derivatives of each linear polynomial as defined by equations (64)-(66) are calculated with negligible error when each small stencil's polynomial is applied to a linear analytical function. These derivatives are used to calculate the smoothness indicators for each polynomial. The smoothness indicators are used to calculate the non-linear weights. While the derivatives have been validated for

linear functions, the overall non-linear weight modifications of the linear weights do not produce accurate extrapolations. These non-linear weights must be made to work before the WENO scheme can be used to create accurate CFD simulations. When it was discovered that the linear weights alone could not be used in the flux calculations without causing the solution to diverge, a different limiting function was used in place of the faulty non-linear weight implementation. This limiting function did not allow any extrapolated flow variable value for a quadrature point to lie outside the minimum or maximum value of that flow variable's cell average value for the reference cell and the first level neighbors of the reference cell. If an extrapolation was found to lie outside of these bounds, the extrapolated value was set to either the minimum or the maximum of the range, depending on which bound the extrapolation had breached. However, this limiting function proved unsuccessful in eliminating the divergence of the solution. In summary, the non-linear weights are a crucial element to the WENO scheme and must be made to function correctly before the scheme can be used for the study of CFD problems.

Ill-Conditioning of Linear Weights Problem

Problems are encountered when applying the linear weight calculation procedure to more geometrically complex meshes. It is observed that as the condition number of the left hand side matrix from the least squares problem, equation (58), increases, the magnitude of the calculated values of linear weights also drastically climbs. These linear weights then produced larger extrapolation errors when used to combine the linear polynomials. Therefore, it is of interest to see the extreme maximum and minimum values produced by the linear weights calculation. An examination of these linear weight

values and the corresponding error produced when they are used to obtain extrapolations has been performed for the two tetrahedral grids used in the extrapolation error analysis. The 114 cell grid can be seen in Figure 4while the larger grid that was also used in the vortex convection case can be seen in Figure 5. For every reference cell there is a set of linear weights calculated for every quadrature point on that cell. For both meshes examined, the maximum and minimum values of any linear weight were found to have come from the same set of linear weights.

For the 114 cell tetrahedral grid, the maximum value was seen to be 1876.0 while the minimum value was equal to -625.4. These extreme values came from a set of linear weights calculated for the same reference cell and quadrature point. The error seen in using these extreme values for extrapolations was then obtained by using this set of linear weights to combine the linear polynomials for analytical function extrapolations to that quadrature point. The linear weights without the use of the splitting technique produced an average error of 7.20E-13 for the three analytical functions. The linear weights used in conjunction with the splitting technique produced an average error of 8.47E-12.

The same procedures were then done for the larger tetrahedral mesh. The maximum and minimum values for calculated linear weights are equal to 16252539.3 and -15965053.5, respectively. Similar to the 114 cell grid, these extreme linear weight magnitudes were located within the same set of linear weights. Using this set of linear weights, an average error value was found to be 6.42E-7. When the splitting technique was introduced to the linear weights, the average extrapolation error was increased to 8.01E-7.

The extrapolations performed using the 114 cell grid's extreme linear weight set only produced the maximum extrapolation error for the whole grid in one of the six extrapolations performed. This was for the third analytical function as listed in Table 9 with the use of the splitting technique. Similarly, the extrapolations performed for the larger grid's extreme linear weights showed only one out of six extrapolations that were the most erroneous in the grid. However, this extrapolation was for the non-split linear weights when used to extrapolate the third analytical function. It could be that, in this case, the round-off errors introduced by the additional operations associated with the splitting technique actually negated some of the extrapolation error from the linear polynomials. The presence of such extreme values produced by the linear weights calculation is undesirable. This is especially true in the case of the larger grid where the magnitudes are very large. In order for WENO to be implemented accurately, this issue needs to be addressed. The condition number of the left hand side matrix during the linear weight calculation for the 114 cell grid was calculated using Matlab to be 2.99E16. This large of a condition number indicates that the system is ill-conditioned. Matlab was also used to compute the condition number of the corresponding matrix for the larger grid, and this was found to be 8.72E16. This increase in condition number could be the reason for the large increase in the extreme linear weight magnitudes and associated error magnitudes for the larger grid when compared to the 114 cell grid. Further study needs to be done on ways to generate a more stable system of equations when solving for the linear weights.

Each small stencil has 4 constants in each column of the LHS matrix. Each of these constants corresponds to one small stencil cell. In some cases, the linear

polynomial procedure produces very small values for one of the small stencil constants. This means that the polynomial reconstruction is effectively limited to the information pulled from 3 cells instead of 4. For a given quadrature point, this happens for 2 or more different small stencils, and the 3 cells that are being used for the effective reconstruction are the same cells. These small stencils are not exactly the same stencil because the fourth ineffective members of these small stencils are different cells, making the selecting of these stencils valid. However, the fourth member does not contribute, and the other small stencil constants are very close in value between the different small stencils. This means two different small stencils are being produced that have very small differences in their calculated constants, and, therefore, very small differences in their reconstructed values. This leads to two columns of the LHS matrix of the linear weights' least-squares problem being approximately equal in value. This causes the condition number of this LHS matrix to increase, which has been shown to cause the calculated values of the linear weights to increase. These larger magnitude linear weights have then been shown to cause an increase in error.

Mapping Function

A mapped weighting technique has been reported in WENO schemes for the purpose of increasing accuracy. The methodology presented in this section was developed by Henrick et al. [9]. The addition of this mapped weighting technique is referred to as WENOM by Nichols et al [3], where the M signifies the mapped weighting used in the WENO scheme. The weighting function is defined by equation (84).

$$g_s(\omega_s;\gamma_s) = \frac{\omega_s(\gamma_s + \gamma_s^2 - 3\gamma_s\omega_s + \omega_s{}^2)}{\gamma_s^2 + (1-2\gamma_s)\omega_s} \tag{84}$$

The non-linear weights used in the WENO reconstruction can now be replaced by the mapped weights, $\omega_s{}^M$, which are expressed as equation (85).

$$\omega_s^M = g_s(\omega_s;\gamma_s) \tag{85}$$

The mapped non-linear weights calculated by equation (85) are used in equation (78) for flow variable extrapolation to the quadrature points. If the splitting technique of Shi et al. [8] is used, then 2 groupings of mapped non-linear weights are calculated using equation (86).

$$\omega_s^{\pm,M} = g_s(\omega_s^{\pm};\gamma_s^{\pm}) \tag{86}$$

The groupings of mapped non-linear weights are then used to calculate the final WENO extrapolations to the quadrature points with equations (80) and (82). Introducing this mapped weighting function could prove to increase the accuracy of the current implementation.

Extension to Grids With Cells of Any Shape

The WENO scheme described in this paper was implemented for use with generalized grids containing elements with a maximum of 6 faces. This section will describe the considerations and changes that would have to be made to enable the scheme to utilize grids containing any three-dimensional element type.

The limitation of only 4 to 6 sided elements within the mesh directly limits the shapes of faces within the mesh to triangles or quadrilaterals. Each of the two Gaussian

quadrature schemes implemented by the WENO scheme (details in Appendix A) are designed for one of these type faces. Allowing for elements of any shape would necessitate the implementation of quadrature schemes that could be used for any cell-face shape that could be encountered in the grid. The quadrature schemes are a key component to the WENO formulation as they are responsible not only for the approximation of the cell average values of the local variables, but also the numerical fluxes through each cell-face.

The big stencil selection was formulated based on a maximum of 6 first level neighbors for any given reference cell. Given a reference cell with 9 or more first level neighbors, the big stencil selection could be completed by adding those cells: however, in the current implementation this is never encountered, and second level neighbors are always added. Extending the WENO scheme to element shapes with more faces would involve changing the big stencil selection so that an excess of cells would not be added. The current scheme adds all of the first level neighbors, and then attempts to add 2 second level neighbors for every first level neighbor. If a cell contained 8 first level neighbors, 16 corresponding second level neighbors could be added, resulting in a big stencil population of 25 (including the reference cell). The minimum big stencil population required for the quadratic polynomial reconstruction is 10, so unwarranted computational cost would be incurred by a big stencil population of 25 members. For the case of 8 first level neighbors, only 1 second level neighbor is required to be added to the big stencil for full population. Considering how the stencils are used for the flow variable extrapolation, it might prove best to add the second level neighbor that is in closest proximity to the reference cell with respect to their centroids. Another point to

consider is that when performing the flow variable extrapolations it is desirable to pull information from all directions around the control volume. With this in mind, it might prove best to determine which direction around the reference cell is least represented, in terms of big stencil population, and add second level neighbors from the underrepresented directions. This could be a topic of future research and is outside of the scope of this paper.

The small stencil selection would have to be formulated to handle elements with more first level neighbors. An element that has 7 first level neighbors would result in 35 possible small stencil candidates from combinations of those neighbors. Similarly, 8 first level neighbors would result in 56 possible small stencils. The minimum number of small stencils needed is only 7 [6], and very large computational time, as well as additional memory allocation, would be required by such a large number of small stencils. In the current implementation, there is already a limitation on the number of small stencils created using first level neighbor combinations for elements with 5 first level neighbors (10 possible combinations) or with 6 first level neighbors (20 possible combinations). Similar limitations would have to be made that could apply to a cell with any larger number of first level neighbors. For extension to an arbitrary cell shape, the changes made to the small stencil selection will be dependent on the changes made to the big stencil selection. Small stencil selection is restricted because every cell in the big stencil must be represented by at least one small stencil and every cell within a small stencil must share at least one face with another cell in that small stencil. These requirements make the small stencil selection dependent on the level of neighbors to which the big stencil extends.

The WENO scheme described in this paper has been developed for generalized grids containing element shapes of 4 to 6 sides. While these restrictions do allow for a structured boundary layer mesh and an unstructured tetrahedral mesh in the flow domain, it is desirable to make this scheme applicable to any type of generalized grid. In order to be applied to cells of an arbitrary shape, the foreseeable changes to the WENO scheme include the implementation of Gaussian quadrature schemes to handle an arbitrary face shape, and the big and small stencil selections.

Reference Material

1. Harten, A., Osher, S., Engquist, B., and Chakravarthy, S.R., "Some results on uniformly high-order accurate essentially nonoscillatory schemes," Applied Numerical Mathematics, vol. 2, Oct. 1986, pp. 347-377.

2. Liu, X., Osher, S., and Chan, T., "Weighted Essentially Non-oscillatory Schemes," Journal of Computational Physics, vol. 115, Nov. 1994, pp. 200-212.

3. Nichols, R.H., Tramel, R.W., and Buning, P.G., "Evaluation of Two High Order WENO Schemes," AIAA Paper 2007-3920, Presented at the 25th AIAA Applied Aerodynamics Conference, 25-28 June 2007, Miami, FL.

4. Shu, C., "High-order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD," International Journal of Computational Fluid Dynamics, vol. 17, 2003, pp. 107-118.

5. Hu, C., "Numerical Methods for Hyperbolic Equations on Unstructured Meshes," Ph.D. dissertation, Brown University, Providence, RI, 1999.

6. Zhang, Y., and Shu, C., "Third Order WENO Scheme on Three Dimensional Tetrahedral Meshes," Computers in Physics, vol. 5, 2009, pp. 836-848.

7. Ollivier-Gooch, C., Nejat, A., and Michalak, K., "On Obtaining Higher-Order Finite-Volume Solutions to the Euler Equations on Unstructured Meshes", AIAA Paper 2007-4464, June, 2007.

8. Shi, J., Hu, C., and Shu, C., "A Technique of Treating Negative Weights in WENO Schemes", Journal of Computational Physics, vol. 175, 2002, pp. 108-127.

9. Henrick, A., Aslam, T., and Powers, J., "Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points", Journal of Computational Physics, vol. 207, 2005, pp. 542-567.

10. Kini, D., "Application of Weighted Essentially Non-Oscillating Schemes to Hybrid Meshes", Project Report, University of Alabama at Birmingham, Birmingham, AL, 2009.

11. Fagan, M.J., "Finite Element Analysis: Theory and Practice," Harlow, Essex, England: Longman Scientific & Technical, 1992.

12. Roe, P.L., "Approximate Riemann Solvers, Parameter Vector, and Difference Schemes," Journal of Computational Physics, vol. 43, pp. 357-372, 1981.

13. Chapra, S., Canale R. "Numerical Methods for Engineers," 6th ed. Boston: McGraw-Hill, 2010.

14. GaussExamples. 2007. Maria Belk. http://math.bard.edu/~mbelk/math601/GaussExamples.pdf

15. Koomullil, R., "Flow Simulation System for Generalized Static and Dynamic Grids," Ph.D. dissertation, Mississippi State University, Starkville, MS, 1997.

16. Kannan, R., Wang, Z., "LDG2: Variant of the LDG Flux Formulation for the Spectral Volume Method," Journal Scientific Computing 46(2), 2010, pp. 314-328.

17. Harten, A., Engquiest, B., Osher, S. Chakravarthy, S., " Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III," Journal of Computational Physics, vol. 131, pp. 3-47, 1997.

APPENDIX A

GAUSSIAN QUADRATURE

The use of Gaussian quadrature is needed in this WENO scheme for the calculation of the cell average local variables. It is also used during the flux calculation in equation (3). The WENO scheme detailed in this paper is developed for 3-dimensional generalized grids that are restricted by containing faces with 3 or 4 nodes. With this restriction, all of the faces within the grid will either be a triangle or a quadrilateral. This necessitates the use of two different Gaussian quadrature schemes: one for a triangular face and another for a quadrilateral face. Both schemes that have been chosen are 4-point Gaussian quadrature schemes. For a triangular face, the scheme used by Zhang et al. [6] is employed. Given a triangular face with vertices of V_1, V_2, and V_3, the coordinates of the 4 quadrature points, G_1, G_2, G_3, and G_4, can be calculated by equation (87).

$$\begin{aligned} G_1 &= \lambda_1 V_1 + \lambda_2 V_2 + \lambda_3 V_3 \\ G_2 &= \lambda_2 V_1 + \lambda_1 V_2 + \lambda_3 V_3 \\ G_3 &= \beta_1 V_1 + \beta_2 V_2 + \beta_3 V_3 \\ G_4 &= \beta_2 V_1 + \beta_1 V_2 + \beta_3 V_3 \end{aligned} \tag{87}$$

where the weights for the quadrature point coordinate calculation are given by equation (88).

$$
\begin{aligned}
\lambda_1 &= \frac{6-\sqrt{6}}{10}\left(\frac{1}{2}-\frac{\sqrt{3}}{6}\right) \\
\lambda_2 &= \frac{6-\sqrt{6}}{10}\left(\frac{1}{2}+\frac{\sqrt{3}}{6}\right) \\
\lambda_3 &= \frac{4+\sqrt{6}}{10} \\
\beta_1 &= \frac{6+\sqrt{6}}{10}\left(\frac{1}{2}-\frac{\sqrt{3}}{6}\right) \\
\beta_2 &= \frac{6+\sqrt{6}}{10}\left(\frac{1}{2}+\frac{\sqrt{3}}{6}\right) \\
\beta_3 &= \frac{4-\sqrt{6}}{10}
\end{aligned}
\tag{88}
$$

The weights used in this Gaussian quadrature point scheme [6] to perform the integral approximations are given by equation (89).

$$
\begin{aligned}
w_1 = w_2 &= \frac{9-\sqrt{6}}{36} \\
w_3 = w_4 &= \frac{9+\sqrt{6}}{36}
\end{aligned}
\tag{89}
$$

where the subscript indicates which of the 4 quadrature points the weights will be applied to when calculating an integral approximation or flux calculation.

The Gaussian quadrature scheme used for the quadrilateral faces comes from Fagan et al. [11]. For each quadrilateral face, the quadrature points are expressed in the terms of the natural coordinate system of that face. The relation of the natural coordinate system to the global coordinate system is illustrated in Figure 26.

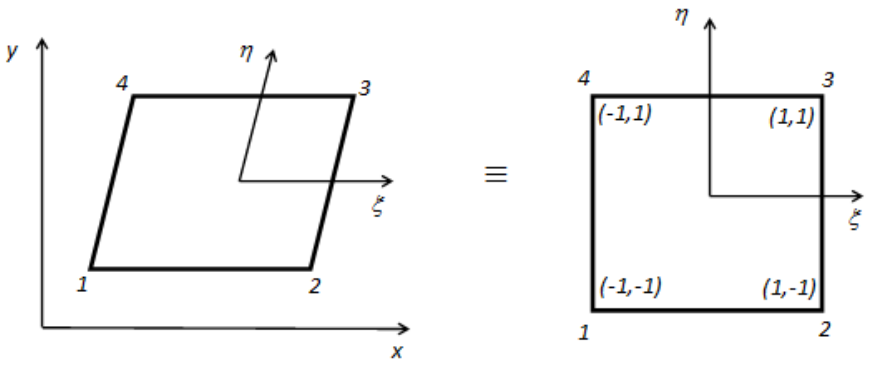

Figure 26. Natural Coordinate System

As long as all of the 4 sides are straight, the quadrilateral can be distorted in the global Cartesian coordinate system but still remain a square, with sides parallel to the axes in reference to the natural coordinate [11]. An interpolation function between global and natural coordinate systems is given by equation (90).

$$\phi = N_1 V_1 + N_2 V_2 + N_3 V_3 + N_4 V_4 \tag{90}$$

where V_1, V_2, V_3, *and* V_4 are the global coordinates of the vertices of the quadrilateral as seen in Figure 26. Each shape function in equation (90) are written as equation (91).

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta) \\ N_2 &= \frac{1}{4}(1+\xi)(1-\eta) \\ N_3 &= \frac{1}{4}(1+\xi)(1+\eta) \\ N_4 &= \frac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{91}$$

Table 14 provides the locations of each quadrature point for any given quadrilateral face with respect to the natural coordinate system [11].

Table 14. Quadrilateral Natural Coordinates of Quadrature Points

Coordinates		Weight
ξ	η	
$\frac{1}{\sqrt{3}}$	$\frac{1}{\sqrt{3}}$	1/4
$-\frac{1}{\sqrt{3}}$	$\frac{1}{\sqrt{3}}$	1/4
$-\frac{1}{\sqrt{3}}$	$-\frac{1}{\sqrt{3}}$	1/4
$\frac{1}{\sqrt{3}}$	$-\frac{1}{\sqrt{3}}$	1/4

The natural coordinates in Table 14 can be converted into global coordinates by use of the shape functions in equation (91) and the interpolation function in equation (90). Equation (92) provides the expressions in which the quadrature points' coordinates are calculated as a function of the vertices coordinates, V_1, V_2, V_3, and V_4, of the quadrilateral face.

$$\begin{aligned} G_1 &= \lambda_1 V_1 + \lambda_2 V_2 + \lambda_3 V_3 + \lambda_4 V_4 \\ G_2 &= \lambda_4 V_1 + \lambda_1 V_2 + \lambda_2 V_3 + \lambda_3 V_4 \\ G_3 &= \lambda_3 V_1 + \lambda_4 V_2 + \lambda_1 V_3 + \lambda_2 V_4 \\ G_4 &= \lambda_2 V_1 + \lambda_3 V_2 + \lambda_4 V_3 + \lambda_1 V_4 \end{aligned} \tag{92}$$

where the weights used in equation (92) are given by equation (93).

$$\lambda_1 = \frac{1}{3} - \frac{\sqrt{3}}{6}$$
$$\lambda_2 = \frac{1}{6}$$
$$\lambda_3 = \frac{1}{3} + \frac{\sqrt{3}}{6} \tag{93}$$
$$\lambda_4 = \frac{1}{6}$$

The weights used for the integral approximations according to this quadrature scheme are given in Table 14, where every quadrature point is weighted equally. A *n*-point Gaussian quadrature scheme can be used to approximate an integral with equation (94) [13].

$$\int f(x)dx = \sum_{G=1}^{n} w_G f(x_G) \tag{94}$$

A function was created in agreement with equation (94) using both quadrature point schemes presented to approximate volume integrals. This function was used to validate the schemes, and to ensure acceptable accuracy for the applications required by the WENO scheme. The following tables present data obtained from this function over a cube. Table 15 presents the values obtained for a unit cube with one vertex on the origin.

Table 15. Integral Approximation Tests for a Unit Cube

Function	Analytical Value	Numerical Result	Error
$\int_{\Omega_i} 1 d\Omega_i$	1.0	1.0000	0.0
$\int_{\Omega_i} x d\Omega_i$	0.5	0.5000	0.0
$\int_{\Omega_i} y d\Omega_i$	0.5	0.5000	0.0
$\int_{\Omega_i} z d\Omega_i$	0.5	0.50000	0.0
$\int_{\Omega_i} xy d\Omega_i$	0.25	0.2500	0.0
$\int_{\Omega_i} xz d\Omega_i$	0.25	0.2500	0.0
$\int_{\Omega_i} yz d\Omega_i$	0.25	0.2500	0.0
$\int_{\Omega_i} x^2 d\Omega_i$	0.3333	0.3333	0.0
$\int_{\Omega_i} y^2 d\Omega_i$	0.3333	0.3333	0.0
$\int_{\Omega_i} z^2 d\Omega_i$	0.3333	0.3333	0.0
$\int_{\Omega_i} x^3 d\Omega_i$	0.25	0.2500	0.0
$\int_{\Omega_i} x^5 d\Omega_i$	0.1667	0.1667	0.0

It can be seen from Table 15 that the accuracy for this ideal case is exceptional. The unit cube was then subjected to translations, rotations, and elongations of the sides in order to ensure the quadrature scheme's accuracy. Table 16 provides the results obtained from translating the unit cube away from the origin.

Table 16. Integration Tests for Translated Unit Cube

Function	Analytical Value	Numerical Result	Error
$\int_{\Omega_i} 1 d\Omega_i$	1.0	0.999999999999998	1.9984E-15
$\int_{\Omega_i} x d\Omega_i$	9.5	9.49999999999999	1.0658E-14
$\int_{\Omega_i} y d\Omega_i$	3.5	3.50000000000000	0.0000E+00
$\int_{\Omega_i} z d\Omega_i$	15.5	15.5000000000000	0.0000E+00
$\int_{\Omega_i} xy d\Omega_i$	33.25	33.2499999999999	9.9476E-14
$\int_{\Omega_i} xz d\Omega_i$	147.25	147.250000000000	0.0000E+00
$\int_{\Omega_i} yz d\Omega_i$	54.25	54.2500000000000	0.0000E+00
$\int_{\Omega_i} x^2 d\Omega_i$	90.33333333	90.3333333333331	1.9895E-13
$\int_{\Omega_i} y^2 d\Omega_i$	12.33333333	12.3333333333333	0.0000E+00
$\int_{\Omega_i} z^2 d\Omega_i$	240.3333333	240.333333333333	0.0000E+00
$\int_{\Omega_i} x^3 d\Omega_i$	859.75	859.749999999998	2.0464E-12
$\int_{\Omega_i} x^5 d\Omega_i$	78093.16666667	78093.1666666665	1.0186E-10

As seen in Table 16, the quadrature scheme is able to achieve acceptable accuracy. The unit cube is tripled in size and translated with respect to the origin. The results from integrating functions over this cell can be seen in Table 17.

Table 17. Integration Tests for Translated and Elongated Unit Cube

Function	Analytical Value	Numerical Result	Error
$\int_{\Omega_i} 1 d\Omega_i$	27.0	27.0000	0.0
$\int_{\Omega_i} x d\Omega_i$	283.5	283.5000	0.0
$\int_{\Omega_i} y d\Omega_i$	121.5	121.5000	0.0
$\int_{\Omega_i} z d\Omega_i$	445.5	445.500	0.0
$\int_{\Omega_i} xy d\Omega_i$	1275.75	1275.7500	0.0
$\int_{\Omega_i} xz d\Omega_i$	4677.75	4677.7500	0.0
$\int_{\Omega_i} yz d\Omega_i$	2004.75	2004.7500	0.0
$\int_{\Omega_i} x^2 d\Omega_i$	2997.0	2997.0000	0.0
$\int_{\Omega_i} y^2 d\Omega_i$	567.0	567.0000	0.0
$\int_{\Omega_i} z^2 d\Omega_i$	7371.0	7371.0000	0.0
$\int_{\Omega_i} x^3 d\Omega_i$	31893.75	31893.7500	0.0
$\int_{\Omega_i} x^5 d\Omega_i$	3681814.5	3681814.5000	0.0

APPENDIX B

LOCAL VARIABLE CALCULATION

The cell averages of the local variables must be calculated for every cell in the big stencil for the calculations of the big stencil constants, as well as the small stencil constants. The cell average values of the local variables for cell *i* were given by the triple integrals in equation (8). The Gauss theorem can be written as equation (95)[14].

$$\int_V (\nabla \cdot F) dV = \int_S \left(F \cdot \vec{n}\right) dS \tag{95}$$

where equation (95) is written for a volume, *V*, a closed surface, *S*, and the normal to the closed surface, $\vec{n}$. Using the Gauss theorem as expressed in equation (95), the triple integrals in equation (8) is converted to surface integrals seen in equation (96).

$$\begin{aligned}
\overline{\xi_i} &= \frac{1}{V_i}\int_{V_i}\frac{(x-x_0)}{h}dV_i = \frac{1}{2hV_i}\int_{S_i}(x-x_0)^2 n_x dS_i \\
\overline{\eta_i} &= \frac{1}{V_i}\int_{V_i}\frac{(y-y_0)}{h}dV_i = \frac{1}{2hV_i}\int_{S_i}(y-y_0)^2 n_y dS_i \\
\overline{\zeta_i} &= \frac{1}{V_i}\int_{V_i}\frac{(z-z_0)}{h}dV_i = \frac{1}{2hV_i}\int_{S_i}(z-z_0)^2 n_z dS_i \\
\overline{\xi_i^2} &= \frac{1}{V_i}\int_{V_i}\frac{(x-x_0)^2}{h^2}dV_i = \frac{1}{3h^2V_i}\int_{S_i}(x-x_0)^3 n_x dS_i \\
\overline{\eta_i^2} &= \frac{1}{V_i}\int_{V_i}\frac{(y-y_0)^2}{h^2}dV_i = \frac{1}{3h^2V_i}\int_{S_i}(y-y_0)^3 n_y dS_i \\
\overline{\zeta_i^2} &= \frac{1}{V_i}\int_{V_i}\frac{(z-z_0)^2}{h^2}dV_i \frac{1}{3h^2V_i}\int_{S_i}(z-z_0)^3 n_z dS_i \\
\overline{(\xi\eta)_i} &= \frac{1}{V_i}\int_{V_i}\frac{(x-x_0)(y-y_0)}{h^2}dV_i = \frac{1}{2h^2V_i}\int_{S_i}(x-x_0)^2(y-y_0) n_x dS_i \\
\overline{(\xi\zeta)_i} &= \frac{1}{V_i}\int_{V_i}\frac{(x-x_0)(z-z_0)}{h^2}dV_i = \frac{1}{2h^2V_i}\int_{S_i}(x-x_0)^2(z-z_0) n_x dS_i \\
\overline{(\eta\zeta)_i} &= \frac{1}{V_i}\int_{V_i}\frac{(y-y_0)(z-z_0)}{h^2}dV_i = \frac{1}{2h^2V_i}\int_{S_i}(y-y_0)^2(z-z_0) n_y dS_i
\end{aligned} \tag{96}$$

where n_x, n_y, and n_z are the surface normals for the x, y, and z directions, respectively. The surface integrals in equation (96) can be written into one generic equation for any local variable. This generic equation is written for each of the 3 directions in which the Gauss theorem can be applied and is seen in equation (97).

$$\begin{aligned}
\overline{(\xi^p \eta^q \zeta^r)_i} &= \frac{1}{(p+1)h^{p+q+r}V_i} \int_{S_i} (x-x_0)^{p+1}(y-y_0)^q (z-z_0)^r n_x dS_i \\
\overline{(\xi^p \eta^q \zeta^r)_i} &= \frac{1}{(q+1)h^{p+q+r}V_i} \int_{S_i} (x-x_0)^p (y-y_0)^{q+1}(z-z_0)^r n_y dS_i \\
\overline{(\xi^p \eta^q \zeta^r)_i} &= \frac{1}{(r+1)h^{p+q+r}V_i} \int_{S_i} (x-x_0)^p (y-y_0)^q (z-z_0)^{r+1} n_z dS_i
\end{aligned} \tag{97}$$

where p, q, and r are the exponents to the local variables, ξ, η, and ζ, respectively. The 4-point Gaussian quadrature schemes can be applied to equation (97) in order for those expressions to be evaluated. After the application of the quadrature scheme of the form of equation (94), equation (97) takes the form of equation (98).

$$\begin{aligned}
\overline{(\xi^p \eta^q \zeta^r)_i} &= \frac{1}{(p+1)h^{p+q+r}V_i} \sum_{j=1}^{faces} \left\{ \sum_{G=1}^{4} w_G \left[(x_G - x_0)^{p+1}(y_G - y_0)^q (z_G - z_0)^r \right] \right\} (n_x)_j A_j \\
\overline{(\xi^p \eta^q \zeta^r)_i} &= \frac{1}{(q+1)h^{p+q+r}V_i} \sum_{j=1}^{faces} \left\{ \sum_{G=1}^{4} w_G \left[(x_G - x_0)^p (y_G - y_0)^{q+1}(z_G - z_0)^r \right] \right\} (n_y)_j A_j \\
\overline{(\xi^p \eta^q \zeta^r)_i} &= \frac{1}{(r+1)h^{p+q+r}V_i} \sum_{j=1}^{faces} \left\{ \sum_{G=1}^{4} w_G \left[(x_G - x_0)^p (y_G - y_0)^q (z_G - z_0)^{r+1} \right] \right\} (n_z)_j A_j
\end{aligned} \tag{98}$$

where *faces* equals the number of faces that make up cell i, index G represents the quadrature point number for face j and has coordinates (x_G, y_G, z_G) which are used to evaluate the function, $(n_i)_j$ is the i-th normal component to face j, and A_j is the area of face j. A function has been created that can calculate equation (98) when given values for p, q, r, the reference cell (cell 0), and cell i. This allows for the calculation of all big stencil cells' cell average values, including the reference cell. The values produced for $\overline{\xi_o}, \overline{\eta_o}$,

and $\overline{\zeta_o}$ are theoretically zero, but, if there is some error in the calculation of the reference cell's centroid, (x_o, y_o, z_o), then a non-zero value could be produced. For tetrahedral elements, the HYB3D solver calculates the centroids exactly, but for elements of different shapes there could be some error in the centroid calculation. For this reason, the terms of $\overline{\xi_o}, \overline{\eta_o}$, and $\overline{\zeta_o}$ have been left in the formulation. The function that calculates these values is a key component to the WENO scheme, and validation was performed to ensure its accuracy. The data structures previously existing from the HYB3D code included an array with the information indicating which cells were on either side of every face. For the function calculating equation (98) it is necessary to know what faces make up a cell. It would be very costly to search through each face in the HYB3D data structures to see if it makes up the cell. For the purpose of efficiency, an array containing the faces that make up every cell was created. This allows for much more rapid cycling through the faces.