Application of the Discontinuous Galerkin Method to Wake Vortex Flows

by

Robert M Watson III

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 9, 2015

Keywords: Discontinuous Galerkin, Vortex Burst, Euler, Artificial Viscosity

Approved by

Andrew Shelton, Adjunct Professor of Aerospace Engineering
Roy Hartfield, Professor of Aerospace Engineering
Brian Thurow, Associate Professor of Aerospace Engineering

Abstract

A code was developed that utilizes the discontinuous Galerkin method to solve the Euler equations while utilizing a modal artificial viscosity sensor developed by Klöckner [12]. The sensor was augmented for the purpose of this research so that it could be run more quickly as well as having a more robust adaptation to different problems and specifically for this research the vortex burst problem. The sensor had a number of flops reduced from its calculation through the use of a different approach for creation of the modes. This new approach was based off of multiplying through by the Vandermonde matrix built from a one dimensional system instead of a three dimensional system. This was done by taking the cube of nodes for a cell and multiplying it by slices of data. The sensor was then made more robust by changing the value by which baseline decay model was added to the modes. The baseline decay model is needed to remove white noise from being sensed upon. Both these changes were reflected in examinations of routine test problems involving the Sod shock tube and the Kelvin-Helmholtz phenomena. With test cases validating the improvements to the sensor, the approach was then used against an element of wake vortex flow to show the sensor's robustness. Through plotting the pressure, vorticity, and total kinetic energy the test case was validated against previous research.

Acknowledgments

I owe so much to my support system, without them I would not be where I am today. My wife married me knowing that I was coming to graduate school in a place where we knew no one and she has supported me through everything. I do not know how I would have accomplished this without her by my side. Her continued support for me as I further my education in the pursuit of my PhD is one of the greatest examples of love and patience I know.

My son has been a great centering force and helped me understand life so much better. Holding him in my arms was the one of the single happiest moments of my entire life. Through my continued education I hope to raise a good man who understands the value of knowledge.

Dr. Andrew Shelton has been very influential in nurturing and helping me develop my abilities as a programmer as well as greatly enhancing my knowledge of fluid mechanics. Whenever I have needed help or guidance while I was pursuing my degree, he has been there as my mentor and my leader. I would not be half the engineer I am today without his molding hand and guidance. He believed in a graduate from the University of Florida that he didn't even know and I will be forever grateful for that.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

γ	Specific heat ratio
$\hat{u}_j^k$	Nodal state
$\mathbb{P}_N$	Function space of order N
Ω	True problem domain
Ω_h	Domain after it has been discretized into cells
$\tilde{u}_j^k$	Modal state
L	Lifting Matrix
M	Mass matrix
N	Order of the polynomial
N_p	Number of points in a cell
S	Stiffness Matrix
$\mathbf{u}$	State vector
a	Speed of sound
DG	Discontinuous Galerkin method
DNS	Direct numerical simulation
LDG	Local Discontinuous Galerkin
LES	Large eddy simulation

TVD	Total variation diminishing
V	Vandermonde matrix

Chapter 1

Introduction

1.1 Motivation

The world is full of highly difficult engineering and scientific problems that cannot be solved by conventional means such as the Navier-Stokes equations. This is why experiments are developed to replicate the physics, analyze it and unwrap the physics occurring in these problems. There is a problem with a pure experimental approach though in that the cost surrounding the development and running of these experiments can be very high. This problem has fueled the development of different numerical techniques to supplement these experiments in solving the partial differential equations that govern physics. This is not to say that any numerical simulation can replace an experiment. Their usefulness lies in their ability to provide guided experiments to reduce the cost of running multiple live experiments. This development has led to a major focus of modern numerical techniques to solve the partial differential equations that govern the physics of these problems.

The desire to always further the study of numerical techniques has led to many vast improvements in both their implementation and the types of solvers available. A major development was moving from the single point finite difference (FD) method into the finite volume (FV) and finite elements (FE) methods which utilize elements. The goal of all these methods was centered on being accurate to the physics of the problem while also making itself faster than previous iterations allowed. To further improve upon these methods accuracy a 4th method was developed, the discontinuous Galerkin (DG) method . This method has been introduced relatively recently compared to the other methods. This causes a necessity to improve the scheme and make it more robust and efficient in order to remove any qualms

about the method. This is very common with numerical techniques whenever they are developed.

This thesis utilizes this DG method to model features of wake flow and improves upon an existing sensor to make it more robust for a multitude of problems. The eventual goal is to eventually develop a fully realized wake modeling code utilizing these techniques. The simulation of a full wake is highly important for all types of engineering problems spanning from airport takeoffs and landings to the development of car drafting techniques. There is a reason these problems are so crucial and interesting though and that is the problems they present in order to be correctly modeled. A very large problem is wrapped in the scale of the problem. It is very difficult to try and capture all of the physics of the problem through a correctly sized domain because it will always lead to a preference being placed on either the small or large scale features. This is why not only the DG method can be a vast improvement but the adaptive DG allows for an even better solution. For the scope of this thesis certain key features will be examined.

A common feature in the evolution of a wake is the sudden change in a vortex core with no change in its circulation due to a vortex bursting. This is a well established phenomena and the problem has been undertaken by many modern day codes utilizing both the large eddy simulation (LES) and direct numerical simulation (DNS) . These techniques are very useful but also are very intricate in how they work and thus require a large amount of time to be run. By studying and comparing the data developed from this current configuration to the ones already on file the goal is established to prove this method's effectiveness.

1.2 Background

Since this work is based on a key aspect of wake evolution a brief history of the development of wake analysis theory and its applications follows. This is then followed by a general overview of the history of the DG method.

1.2.1 A Brief History of Wake Theory

The concept of aircraft wakes has been an important aspect of aviation for a long time. The wake has strong impacts on a multitude of situations such as aircraft taxiing and deployment of rotocraft on aircraft carriers. The decay of vortices throughout the wake plays one of the most important roles in this research especially at airports. On a grander scale there is a hope that with the development of a better understanding of vortices, the same will be said of turbulence. Turbulence remains one of the greatest mysteries of the modern era while also being one of the most divisive topics to discuss. The exact definition of turbulence is not something that is even unilaterally agreed with.

The analysis of aircraft wakes has been undertaken for as long as aircraft have existed. Scientists such as Crow [6] spent a lot of time developing analytical theories and models in the 1970's. Crow's work was about an understanding of the vortices contained in the wake of an aircraft being dispersed through natural means. Crow in fact developed the theory behind a common instability that has been well validated by todays experiments and simulations. This instability today is proclaimed as the Crow instability. In the time between 1970 and the mid 1990's wake modeling had advanced from a dependence on different analytical models to accepting the use of computational fluid dynamics models to supplement some of the research. The scientists Lewellen and Lewellen [13] in 1996 developed one of the first extensive three dimensional models of vortex wakes using a custom version of LES where they focused on different aspects of the gas dynamics and chemistry. They specifically focused on the chemistry of the engine wake from the aircraft and it's interaction with the atmosphere.

The work of Crow and others was then brought together by Spalart [21] in 1998 who provided an encompassing survey of the knowledge about wakes to that point. He discusses a multitude of issues concerning the propagation of wakes from aircraft and different aspects of analyzing them through experimental setups at airports. He makes special note of the struggles in developing a model due to the limitations of the assumptions needed in CFD of that era. He also helped develop some of the base theory concerning the vortex bursting,

challenging a preconceived notion that bursting was the destruction of vortices, which holds no weight when examining the argument from the standpoint of circulation.

These issues are still being examined and resolved in today's climate especially concerning CFD and experimental. Multiple programs have been established such as the Aircraft Vortex Spacing System (AVOSS) used by NASA to study these vortices and establish more scientifically based parameters especially with the influence of the atmosphere. Gerz et. al. [8] utilizes the data from AVOSS along with other similar agencies discusses different ways to discuss further change that can be applied to airports. Their goal was to propose ways to increase the efficiency of airports by utilizing known data about wakes in order to increase the number of commercial aircraft that can take off and land over the course of a day.

A common element of all the research is the breakdown of vortices which is explicitly examined by Holzäpfel[11] et. al. This group examined the effects of primary and secondary vortices, which are created through moderate turbulence from the ambient flow, have on each other in the case of decay through numerical simulations. It is also examined by Moet et. al.[16]. Their research is used as the basis for the vortex bursting validation undertaken in this research.

1.2.2 Development of Numerical Methods

In engineering, and science in general, the development of suitable numerical techniques has been a staple of research for many years. Common methods have been used with the FD, FV, and FE methods. In terms of modern CFD the majority of codes use second order accurate FV [23] methods or FD methods. While these methods are useful they do not fulfill the long term goals of the numerical sciences. They are difficult to increase the order of accuracy as the mesh complexity is increased moving from pure Cartiesian to non-uniform meshes with triangular elements. They are also expensive computationally for the spatial discretization of problems. This is not a problem with the Finite Element method but it has major flaws in that for each time step the entire matrix is solved at once creating

inconsistencies when applied to the physics of a fluid mechanics problem where information changes with the flow.

These desires led to the development of a new method of solving PDE's that gave the flexibility required for solving two dimensional and three dimensional problems while also allowing for a higher order of accuracy akin to FE codes. This new method was called the DG method. This method was first proposed as a new way to solve the neutron transport equation by Reed, etc. [18]. The method slowly grew in its different applications to different fields but it was not till the mid 1990's that this method exploded into applications to CFD. Advances were made in the development of Runge-Kutta DG methods that provided an explicit and cheap method to use in the solution of problems. In 1996 [2] and 1997 [3] Bassi and Rebay published two papers creating the local DG (LDG) method and applying it to the Euler and Navier-Stokes equations respectively. This method further improved the previous results because in the previous DG methods only the time discretization was discontinuous but now both space and time were.

The LDG method quickly became one of the most useful adaptations of the method. Cockburn and Shu [4] analyzed this method and showed the great promise that comes from its inherent flexibility. It's discontinuous nature throughout allows for easy adaptations to parallel processing which would allow for the code to become much faster. They would later compare this method against versions of the DG method[5]. Here they found the LDG method had inherent advantages in adapting to not only elliptical PDE's but also hyperbolic and parabolic PDE's allowing for a more general use of this method to problems even outside of the realm of CFD. This was further explored by Yan and Osher [24] in the solving of the Hamilton-Jacobi problems. This was then taken further by Kevin Albarado when he purposed the DG method to solve the Hamiltonian for the burning of a two dimensional start solid rocket grain[1].

1.3 Objective

The purpose of this thesis is to further the development of a robust DG scheme based on the Euler equations and implementing an artificial viscosity sensor that has been augmented to make it more applicable to different this problem. Artificial viscosity is a necessity in that even though the Euler equations are inherently invisicid, the numerical equations solved have a viscous term when backed into their conservative forms. This is normally taken care of in the inherent dissipation in the numerical scheme but with the high order of the DG method this needs to be readdressed. This artificial viscosity sensor will be altered and improved upon in order to allow for more versatility and robustness on a range of problems from shocks to vortex interactions while also increasing the speed which is always important when working with numerical simulations and especially when these problems take place in three dimensional space. Different test problems will be examined in order to accurately show the effects of these two changes to these important factors.

After the sensor has been tuned for optimal use on three dimensional problems the vortex bursting problem will be solved. This will be done by using a complex method for developing the initial condition that utilizes a pseudo-steady state problem. The mesh size and polynomial order will be varied to show the effects on the full problem. The helical instability will be tripped as well to show a more physically accurate problem as well as to show that the sensor can account for artificial viscosity where needed but not overload the problem and "wash out" useful and accurate physics from the problem.

Chapter 2

Discontinuous Galerkin Method

What follows is the development of the DG method and a process to apply it to a simple one dimensional problem. This analysis follows a procedure done by Hesthaven et. al. [10]. The DG method is a hybrid of sorts of the FV and FE methods in order to gain the distinct advantages of each method. This allows the new method to have the geometric flexibility of a FV code while allowing for much more flexible method of creating a higher order of accuracy spatially. This is a major problem in many current CFD codes running FV solvers. The simple one dimensional transport equation will be used as a tool for the application of this method where u represents the velocity of the wave.

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \tag{2.1}$$

The first part of any numerical scheme is to take the domain of the problem (Ω) and discretize it into K number of cells for a new domain (Ω_h) . The point of discretization is to break a domain into something that can be solved computationally. Each of these cells attaches to surrounding cells at an interface that overlaps. This is an important feature for the creation of a flux that occurs between the two cells which will be discussed later in this section. An example of how the discretized domain is represented is shown in Figure 2.1.

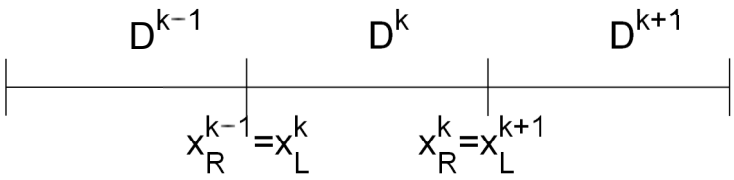


Figure 2.1: 1D discretization of a domain into three elements

The global true solution u is then first equated to the computational solution and then broken up into each cell such that

$$u(x,t) \simeq u_h(x,t) = \bigoplus_{k=1}^{K} u_h^k(x,t) \tag{2.2}$$

where there is now a state for each cell that combines into an approximation with the global state. A high order local basis function is then applied to each cell's state in order to create a nodal state

$$u_h^k(x,t) = \sum_{j=0}^{N} \tilde{u}_j^k(t) b_j^k(x) \tag{2.3}$$

The residual $\mathcal{R}_h$ is then created and with this method it has to vanish for each cell in a Galerkin sense where the basis function forms the weighting function. This is how this method differs from a traditional Galerkin FE method where the residual vanishes globally. This is also what allows the method to be more open to a parallelization which will be discussed later.

$$x \in D^k : \ \mathcal{R}_h^k(x,t) = \frac{\partial u_h^k}{\partial t} + \frac{\partial f_h^k}{\partial x} \tag{2.4}$$

$$\int_{D^k} b_j^k \mathcal{R}_h \mathrm{dx} = 0 \quad j = 0, ..., N \tag{2.5}$$

Here N is the order of the polynomial and for the one dimensional test case the number of points per cell is equal to one more than the order ($N_p = N + 1$) .

With the distinction that each cell is unique the boundaries of each cell can have multiple solutions depending on which cell is being examined which leads to Figure 2.2.

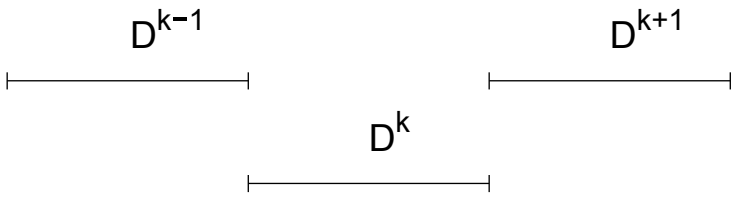


Figure 2.2: Cells having different boundary values

They can then communicate between cells through a numerical flux function which in terms of CFD is physics based with examples of Lax-Friedrichs or AUSM$^+$.

With the cells decoupled the original function was then solved by an integration by parts using the divergence theorem to obtain

$$\int_{D^k} b_i^k \frac{\partial u_h^k}{\partial t} \mathrm{d}x + \int_{D^k} b_i^k \frac{\partial f_h^k}{\partial x} \mathrm{d}x = 0 \tag{2.6}$$

$$\int_{D^k} b_i^k \frac{\partial u_h^k}{\partial t} \mathrm{d}x + \int_{\partial D^k} \frac{\partial}{\partial x} \left(b_i^k f_h^k \right) \mathrm{d}x - \int_{D^k} \frac{\partial b_i^k}{\partial x} f_h^k \mathrm{d}x = 0 \tag{2.7}$$

$$\int_{D^k} b_i^k \frac{\partial u_h^k}{\partial t} \mathrm{d}x + \oint_{\partial D^k} b_i^k f_h^k n_x^k \mathrm{d}s - \int_{D^k} \frac{\partial b_i^k}{\partial x} f_h^k \mathrm{d}x = 0 \tag{2.8}$$

where n_x^k is the element normal. Now even though the cells are independent of each other there is still a piecewise continuous element of the solution concerning the flux. As previously mentioned this is a problem because all the cells needs to be discontinuous. By using the previously mentioned flux function the previous equation now allows for the discontinuity at the cell edge.

$$x \in \partial D^k : \quad f_h^k n_x^k \rightarrow f^* \left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-} \right) \tag{2.9}$$

The flux is designed such that the flux coming from one cell is equal to the negative of the flux leaving the adjacent cell at the same boundary.

$$f^*\left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right) = -f^*\left(u_h^{k,+}, u_h^{k,-}; n_x^{k,+}\right) \tag{2.10}$$

$$\int_{D^k} b_i^k \frac{\partial u_h^k}{\partial t} \mathrm{d}x + \oint_{\partial D^k} b_i^k f^*\left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right) \mathrm{d}s - \int_{D^k} \frac{\partial b_i^k}{\partial x} f_h^k \mathrm{d}x = 0 \tag{2.11}$$

This gives $N_p * K$ unknowns and equations which can be solved cell by cell. Here lies the main question of the DG method which is to find $u_h^k \in \mathbb{P}_N$ such that $b_i \in \mathbb{P}_N$. Where $\mathbb{P}_N$ stands for a function space on the standard interval of order less than or equal to N and vanishes everywhere else. This leads to a need for a strong basis function.

2.1 Polynomials for Cells

2.1.1 Cell Mapping

An important aspect of a Galerkin model is a strong basis function to work with. Polynomials are widely used here because of their relative simplicity over Fourier modes. There is an exception in the case of spectral methods due to Fourier models providing a simpler model to work with. They also follow closely with how people tend to think about data in the first place. A standard interval, $x \to \xi \in (-1, 1)$, is used to develop a mapping function which can be extrapolated to any standard cartesian cell.

$$x \in D^k : \quad x = \mathcal{M}^k(\xi) \tag{2.12}$$

$$\mathcal{M}^k(\xi) = x_a^k + \frac{1+\xi}{2}\left(x_b^k - x_a^k\right) \tag{2.13}$$

2.1.2 Basis Functions

The basis functions were correlated with the state such that

$$u_h^k(\xi) = \sum_{m=0}^{N} \tilde{u}_m^k \pi_m(\xi) = \sum_{n=0}^{N} \hat{u}_n^k \ell_n(\xi) \tag{2.14}$$

Where $\tilde{u}_j^k$ corresponds to the modal state and $\hat{u}_j^k$ the nodal state. This is particularly useful because both nodal and modal analysis have different strengths for analysis. This ability to move between the two gives this method another advantage over traditional FV methods.

Modal Analysis

A traditional thought in the creation of a modal analysis is the development of a monomial based on the given polynomial order to set the basis function. Then using an L^2-projection to recover the state $\tilde{u}_m^k$ leads to

$$M\tilde{u} = u \tag{2.15}$$

The problem is that this system fails due to the matrix M being very poorly conditioned and thus the modal state cannot be recovered from an inverse of M. This led to the idea of using a L^2 based Gram-Schmidt orthogonalization in order to develop a basis that was orthonormal.

$$\begin{gathered} \sqrt{\beta_{m+1}}\pi_{m+1}(\xi) = (\xi - \alpha_m)\pi_m(\xi) - \sqrt{\beta_{m-1}}\pi_{m-1}(\xi) \\ \pi_{-1} \doteq 0, \quad \pi_0 \doteq 1/\sqrt{\beta_0} \end{gathered} \tag{2.16}$$

This resulted in the choice of the Legendre polynomials which is found by selecting certain values for α_m and β_m. The Legendre coefficients are listed as

$$\alpha_m = 0, \;\; \beta_m = \begin{cases} 2 & \text{if } m = 0 \\ 1/(4 - m^{-}2) & \text{if } m \geq 1 \end{cases} \tag{2.17}$$

An important property of these legendre polynomials is that

$$\int_{-1}^{1} \pi_i(\xi)\pi_j(\xi)\mathrm{d}\xi = \delta_{ij} \doteq \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.18}$$

A modal basis is not as useful though in a direct analysis of the data due to the extra calculations that will need to be undertaken to translate the boundary conditions and initial conditions to a modal system. It is though a great system that works well for the development of the sensor which will be detailed later. A transformation matrix is used to move between nodal and modal values for the state.

Nodal Analysis

An ideal nodal basis function is the Lagrange equation due to it guaranteeing nodal accuracy for any quadrature which could be used. The Lagrange polynomial is created through

$$\ell_h(\xi) = \prod_{i=0, i\neq n}^{N} \frac{\xi - \xi_i}{\xi_n - \xi_i} \tag{2.19}$$

and contains the following powerful property

$$\ell_j(\xi_i) = \delta_{ij} \doteq \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.20}$$

With a powerful nodal representation the quadrature needs to be established based around the Legendre polynomials needs. This leads to the Legendre-Gauss-Lobatto quadrature detailed below.

Numerical Quadrature

Quadrature is an important decision in the development of any numerical technique. This especially holds true if one wants to perform numerical integration which is based on

the distribution of the quadrature and is performed as follows.

$$\int_{-1}^{1} f(\xi)d\xi \simeq \sum_{j=1}^{N_p} \omega_j f(\xi_j) \tag{2.21}$$

Here ω_j represents the weights associated with a particular quadrature Since a nodal basis has been already chosen for the development of this method Gaussian quadrature seems the most logical due it being an exact integration technique for polynomials.

In the development of this method the Gauss-Lobatto rule was applied since the Gauss rule does not include the boundary points thus a Jacobi-Lobato matrix was created.

$$J_{p+2}^{L} \doteq \begin{bmatrix} J_{p+1} & \sqrt{\beta_{p+1}^{*}}\vec{e}_{p+1} \\ \sqrt{\beta_{p+1}^{*}}\vec{e}_{p+1}^{\mathrm{T}} & \alpha_{p+1}^{*} \end{bmatrix} \tag{2.22}$$

where J_p represents the Jacobi matrix

$$J_p \doteq \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & \dots & \dots & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & \dots & 0 \\ \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \sqrt{\beta_{p-1}} & \alpha_{p-1} \end{bmatrix} \tag{2.23}$$

The orthonormal property of the modal analysis needs to be kept in place so the application of the legendre values for α and β were used. This leads to the weights and node locations being found from the eigenvalues and eigenvectors for the Jacobi-Lobatto matrix where

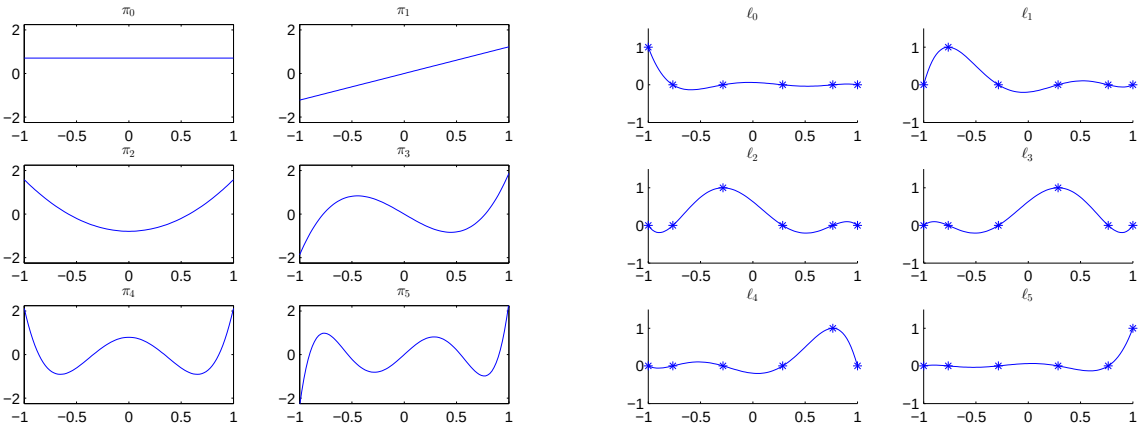
$$\xi_i = \lambda_i^L, \quad \omega_i = \beta_0 (v_{(i)}^L)^2, \quad i = 0, ..., p+1 \tag{2.24}$$

here λ represents the eigenvalues and v represent the eigenvectors.

The application of the Lagrange basis to these points that satisfy the Legendre polynomial is not a problem and thus the two methods now have a standard quadrature.

Combining the Two Analysis

The transformation matrix between the modal and nodal states was something needed and was developed through the application of the Lagrange polynomials to the same points that the Legnedre polynomials were built around. An example of how these two polynomials look up to a 5^{th} order can be seen in Figure 2.3.



(a) Modal basis with $\pi_m(\xi) \in \mathbb{P}_N$ (b) Nodal basis with $\ell_n(\xi) \in \mathbb{P}_N$

Figure 2.3: Basis Function of $\mathbb{P}_N$

By applying the legendre polynomial as the basis function for $\tilde{u}$ it is found that

$$u(\xi_i) = \sum_{m=1}^{N_p} \tilde{u}_m P_{m-1}(\xi_i) \tag{2.25}$$

Where P_{m-1} represents the legendre polynomials which leads to

$$V\tilde{u} = u \tag{2.26}$$

where V represents a generalized Vandermonde matrix. Now both the Vandermonde matrix and its derivatives can be related to the legendre polynomials and their derivatives such that

$$V_{ij} \doteq \pi_j(\xi_i), \quad (V_\xi)_{ij} \doteq \pi_j'(\xi_i) \tag{2.27}$$

This helps to create a bridge between Lagrange and Legendre polynomials so that the state can be altered between a nodal or modal value. This allows the DG method to take advantage of the unique properties provided by modal and nodal analysis in the orthonormality of the modal function with the simplicity of the nodal system.

$$\pi_j(\xi) = \sum_{i=0}^{N} V_{ij}\ell_i(\xi) = \sum_{i=0}^{N}(V^T)_{ji}\ell_i(\xi) \iff \ell_i(\xi) = \sum_{j=0}^{N}(V^{-T})_{ij}\pi_j(\xi) \tag{2.28}$$

$$u_h(\xi) = \sum_{m=0}^{N} \tilde{u}_m\pi_m(\xi) = \sum_{n=0}^{N} \hat{u}_n\ell_n(\xi) \iff \sum_{m=0}^{N} V_{im}\tilde{u}_m = \sum_{n=0}^{N} \delta_{in}\hat{u}_n \tag{2.29}$$

The Legendre-Gauss-Lobatto quadrature leads to a well-conditioned matrix in the Vandermonde matrix. With the basis established the next step entails the manipulation of Equation 2.11 to be of an operator form that can be solved numerically.

2.2 Operator Form

Returning now to the initial one dimensional problem Equation 2.1 and its discretized form

$$\int_{D^k} b_i^k \frac{\partial u_h^k}{\partial t}\mathrm{d}x - \int_{D^k} \frac{\partial b_i^k}{\partial x} f_h^k \mathrm{d}x + \oint_{\partial D^k} b_i^k f^* \left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right) \mathrm{d}s = 0 \tag{2.30}$$

In order to solve the problem computationally the discretized form of the equations needs to be broken down into a matrix form based on the application of the basis function to the state. The flux is then broken up the same way the state is based on the basis function

$$f_h^k(x) = \sum_{j=0}^{N} \tilde{f}_j^k(t) b_j^k(x) \tag{2.31}$$

The time derivative term is broken down through a coordinate transformation between x and ξ. This transformation is based on

$$x \in D^k : \ \mathrm{d}x = \mathcal{J}^k \mathrm{d}\xi, \ \ \mathcal{J}^k = \frac{x_b^k - x_a^k}{2} \tag{2.32}$$

and is then inserted into the time derivative term such that

$$\begin{aligned}\int_{D^k} b_i^k \frac{\partial u_h^k}{\partial t} \mathrm{d}x &= \int_{D^k} b_i^k(x) \frac{\partial}{\partial t} \left(\sum_{j=0}^{N} \tilde{u}_j^k(t) b_j^k(x) \right) \mathrm{d}x \\ &= \sum_{j=0}^{N} \left(\int_{D^k} b_i^k(x) b_j^k(x) \mathrm{d}x \right) \frac{\mathrm{d}\tilde{u}_j^k}{\mathrm{d}t} \\ &= \sum_{j=0}^{N} \left(\int_{-1}^{1} b_i(\xi) b_j(\xi) \mathcal{J}^k \mathrm{d}\xi \right) \frac{\mathrm{d}\tilde{u}_j^k}{\mathrm{d}t} \\ &= \mathcal{J}^k \sum_{j=0}^{N} M_{ij} \frac{\mathrm{d}\tilde{u}_j^k}{\mathrm{d}t}\end{aligned} \tag{2.33}$$

The same method is then applied to the spatial derivative term in order to break it down into a derivative matrix which can be applied ot the flux term

$$\begin{aligned}\int_{D^k} \frac{\mathrm{d}b_i^k}{\mathrm{d}x} f_h^k \mathrm{d}x &= \int_{D^k} \frac{\mathrm{d}b_i^k(x)}{\mathrm{d}x} \left(\sum_{j=0}^{N} \tilde{f}_j^k b_j^k(x) \right) \mathrm{d}x \\ &= \sum_{j=0}^{N} \left(\int_{D^k} \frac{\mathrm{d}b_i^k(x)}{\mathrm{d}x} b_j^k(x) \mathrm{d}x \right) \tilde{f}_j^k \\ &= \sum_{j=0}^{N} \left(\int_{-1}^{1} \frac{\mathrm{d}b_i(\xi)}{\mathrm{d}\xi} b_j(\xi) \mathrm{d}\xi \right) \tilde{f}_j^k \\ &= \sum_{j=0}^{N} \left(S^T \right)_{ij} \tilde{f}_j^k\end{aligned} \tag{2.34}$$

The boundary term is trickier in that specific nodes based on faces and edges have to be used in the application of the flux. This does not encompass the global flux running through

the system.

$$
\begin{aligned}
&\oint_{\partial D^k} b_i^k f^* \left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right) \mathrm{d}s \\
&\qquad = \left[b_i^k f^* \left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right)\right]_{x_a^k} + \left[b_i^k f^* \left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right)\right]_{x_b^k} \\
&\qquad = b_i^k(x_a^k) \sum_{m=0}^{N} \widetilde{(f_a^*)}_m^k b_m^k(x_a^k) + b_i^k(x_b^k) \sum_{n=0}^{N} \widetilde{(f_b^*)}_n^k b_n^k(x_b^k) \\
&\qquad = \sum_{m=0}^{N} \left(b_i^k\left(x_a^k\right) b_m^k\left(x_a^k\right)\right) \widetilde{(f_a^*)}_m^k + \sum_{n=0}^{N} \left(b_i^k\left(x_b^k\right) b_n^k\left(x_b^k\right)\right) \widetilde{(f_b^*)}_n^k \\
&\qquad = \sum_{m=0}^{N} \left(b_i(-1) b_m(-1)\right) \widetilde{(f_a^*)}_m^k + \sum_{n=0}^{N} \left(b_i(1) b_n(1)\right) \widetilde{(f_b^*)}_n^k \\
&\qquad = \sum_{m=0}^{N} \left(L_a\right)_{im} \widetilde{(f_a^*)}_m^k + \sum_{n=0}^{N} \left(L_b\right)_{in} \widetilde{(f_b^*)}_n^k
\end{aligned}
\tag{2.35}
$$

This leads to the original discretized form of the equation going from

$$
\int_D^k b_i^k \frac{\partial u_h^k}{\partial t} \mathrm{d}x - \int_{D^k} \frac{\partial b_i^k}{\partial x} f_h^k \mathrm{d}x + \oint_{\partial D^k} b_i^k f^* \left(u_h^{k,-}, u_h^{k,+}; n_x^{k,-}\right) \mathrm{d}s = 0 \tag{2.36}
$$

to

$$
\mathcal{J}^k \sum_{j=0}^{N} M_{ij} \frac{\mathrm{d}\tilde{u}_j^k}{\mathrm{d}t} - \sum_{m=0}^{N} \left(S^T\right)_{ip} \tilde{f}_p^k + \sum_{m=0}^{N} \left(L_a\right)_{im} \widetilde{(f_a^*)}_m^k + \sum_{n=0}^{N} \left(L_b\right)_{in} \widetilde{(f_b^*)}_n^k \tag{2.37}
$$

Where there are now mass, stiffness, and lifting matrices for the development of the method.

This can further be reduced into a matrix vector form.

$$
\mathcal{J}^k M \frac{\mathrm{d}\tilde{u}^k}{\mathrm{d}t} - \mathcal{S}^T \tilde{f}^k + L_a \widetilde{(f_a^*)}^k + L_a \widetilde{(f_a^*)}^k \tag{2.38}
$$

2.3 Elementwise Operator

As previously mentioned the nodal method has been chosen in this research due it having an advantage in dealing with boundary data as well as nonlinear flux functions. But,

there is no reason not to exploit some of the greater properties of the modal transformation in its orthonormal basis and its hierarchal construction.

The ability of the Vandermonde matrix to manipulate the state between the two basis is a huge advantage in the development of the mass, stiffness, and lifting matrices. This can be seen in the development of the mass matrix

$$\begin{aligned} M_{ij} &\doteq \int_{-1}^{1} \ell_i(\xi)\ell_j(\xi)\mathrm{d}\xi \\ &= \int_{-1}^{1} \sum_{m=0}^{N} \left(V^{-T}\right)_{im} \pi_m(\xi) \sum_{n=0}^{N} \left(V^{-T}\right)_{jn} \pi_n(\xi)\mathrm{d}\xi \\ &= \sum_{m=0}^{N}\sum_{n=0}^{N} \left(V^{-T}\right)_{im} \left(\int_{-1}^{1} \pi_m(\xi)\pi_n(\xi)\mathrm{d}\xi\right) \left(V^{-1}\right)_{nj} \\ &= \sum_{m=0}^{N}\sum_{n=0}^{N} \left(V^{-T}\right)_{im} \delta_{mn} \left(V^{-1}\right)_{nj} \\ &= \sum_{m=0}^{N}\sum_{n=0}^{N} \left(V^{-T}\right)_{im} \left(V^{-1}\right)_{mj} \quad \Longleftrightarrow \quad M = \left(VV^T\right)^{-1} \end{aligned} \tag{2.39}$$

Then with the fore knowledge that there are recursion-based formulas for Legendre derivatives, the creation of a derivative matrix can easily be accomplished from the derivative of the Jacobi matrices.

$$\begin{aligned} D_{ij} &\doteq \ell_i'\left(\xi_j\right) \\ &= \sum_{m=0}^{N} \left(V^{-T}\right)_{jm} \pi_m'\left(\xi_i\right) \\ &= \sum_{m=0}^{N} \left(V^{-T}\right)_{jm} \left(V_\xi\right)_{im} \\ &= \sum_{m=0}^{N} \left(V_\xi\right)_{im} \left(V^{-1}\right)_{mj} \quad \Longleftrightarrow \quad D = V_\xi V^{-1} \end{aligned} \tag{2.40}$$

The stiffness matrix for a nodal basis is built by

$$S_{ij} \doteq \int_{-1}^{1} \ell_i(\xi)\ell_j'(\xi)\mathrm{d}\xi \quad \Longleftrightarrow \quad \left(S^T\right)_{ij} \doteq \int_{-1}^{1} \ell_j'(\xi)\ell_i(\xi)\mathrm{d}\xi \tag{2.41}$$

which when applied to the mass matrix and derivative matrix gives

$$\begin{aligned}
\sum_{n=0}^{N} M_{in} D_{nj} &= \sum_{n=0}^{N} \left(\int_{-1}^{1} \ell_i(\xi) \ell_n(\xi) \mathrm{d}\xi \right) \ell_j' \left(\xi_n \right) \\
&= \int_{-1}^{1} \ell_i(\xi) \left(\sum_{n=0}^{N} \ell_j' \left(\xi_n \right) \ell_n(\xi) \right) \mathrm{d}\xi \\
&= \int_{-1}^{1} \ell_i(\xi) \left(\sum_{n=0}^{N} \widehat{\left(\ell_j' \right)} \ell_n(\xi) \right) \mathrm{d}\xi \\
&= \int_{-1}^{1} \ell_i(\xi) \ell_j'(\xi) \mathrm{d}\xi \\
&= S_{ij} \quad \Longleftrightarrow \quad S = MD
\end{aligned} \tag{2.42}$$

There is a powerful property here due to the need of the transpose of the stiffness matrix. By applying the transpose and through manipulation using the Vandermonde matrix the development of the stiffness matrix can be found through just the Vandermonde matrix and tis derivatives in the necessary directions.

$$\begin{aligned}
S = MD \quad \Longleftrightarrow \quad S^T &= D^T M^T \\
&= \left(V_\xi V^{-1} \right)^T \left(V^{-T} V^{-1} \right)^T \\
&= \left(V_\xi V^{-1} \right)^T \left(V^{-T} V^{-1} \right) \\
&= \left(V_\xi V^{-1} \right)^T \left(V V^T \right)^{-1}
\end{aligned} \tag{2.43}$$

The development of the lifting Matrices are difficult though because they have to extrapolate the flux from the edge of a cell to the rest of a cell. The suffix a and b will be applied ot the Lifting matrices to denote which the cell walls in any direction.

$$\begin{aligned}
\left(L_a \right)_{ij} &\doteq \ell_i(-1) \ell_j(1) & \left(L_b \right) &\doteq \ell_i(1) \ell_j(1) \\
&= \ell_i \left(\xi_0 \right) \ell_j \left(\xi_0 \right) & &= \ell_i \left(\xi_N \right) \ell_j \left(\xi_N \right) \\
&= \delta_{0i} \delta_{0j} & &= \delta_{Ni} \delta_{Nj}
\end{aligned} \tag{2.44}$$

The numerical flux is then defined by the same procedure to each cell wall

$$\begin{aligned}
(f_a^*)_{ij}^k &= \sum_{j=0}^{N} \widehat{(f_a^*)}_j^k \ell_j(-1) & (f_b^*)_{ij}^k &= \sum_{j=0}^{N} \widehat{(f_b^*)}_j^k \ell_j(1) \\
&= \sum_{j=0}^{N} \widehat{(f_a^*)}_j^k \ell_j(\xi_0) & &= \sum_{j=0}^{N} \widehat{(f_b^*)}_j^k \ell_j(\xi_N) \\
&= \sum_{j=0}^{N} \widehat{(f_a^*)}_j^k \delta_{0j} & &= \sum_{j=0}^{N} \widehat{(f_b^*)}_j^k \delta_{Nj} \\
&= \widehat{(f_a^*)}_0^k & &= \widehat{(f_b^*)}_N^k
\end{aligned} \tag{2.45}$$

This all leads to the combination of the lifting matrices and cell fluxes such that.

$$\sum_{i=0}^{N} (L_a)_{ij} \widehat{(f_a^*)}_j^k = \sum_{i=0}^{N} \delta_{0i}\delta_{0j}\delta_{0j} \widehat{(f_a^*)}_j^k = \sum_{i=0}^{N} \delta_{i0} \widehat{(f_a^*)}_0^k \tag{2.46}$$

$$L_a \widehat{(f_a^*)}^k = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & 0 \end{bmatrix} \begin{Bmatrix} \widehat{(f_a^*)}_0^k \\ 0 \\ \vdots \\ 0 \end{Bmatrix} = \widehat{(f_a^*)}_0^k \begin{Bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} = \widehat{(f_a^*)}_0^k \vec{e}_0 \tag{2.47}$$

$$\sum_{i=0}^{N} (L_b)_{ij} \widehat{(f_b^*)}_j^k = \sum_{i=0}^{N} \delta_{Ni}\delta_{Nj}\delta_{Nj} \widehat{(f_b^*)}_j^k = \sum_{i=0}^{N} \delta_{iN} \widehat{(f_b^*)}_N^k \tag{2.48}$$

$$L_b \widehat{(f_b^*)}^k = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & 1 \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ \widehat{(f_b^*)}_N^k \end{Bmatrix} = \widehat{(f_b^*)}_N^k \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{Bmatrix} = \widehat{(f_b^*)}_N^k \vec{e}_N \tag{2.49}$$

These new matrices are then plugged into the local matrix form in Equation 2.38 giving a complete system to create the spatial derivatives for the problem.

Chapter 3

Implementation of Euler Equations

For this project the Euler equations were chosen due to the large scale eddies being the important factors. Therefore the small scale turbulent structures and the turbulence of the problem is not as important especially since any structure that needed to be developed can be tripped in the initial condition with an input to the velocity of some kind instead of by turblence. The Kelvin-Helmholtz problem is a prime example of this where a sinusoidal velocity in the y direction is used to cause the instability to emerge in the jet. The Euler equations are

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0 \tag{3.1}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2 + p}{\partial x} + \frac{\partial \rho uv}{\partial y} + \frac{\partial \rho uw}{\partial z} = 0 \tag{3.2}$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial \rho uv}{\partial x} + \frac{\partial \rho v^2 + p}{\partial y} + \frac{\partial \rho wv}{\partial z} = 0 \tag{3.3}$$

$$\frac{\partial \rho w}{\partial t} + \frac{\partial \rho uw}{\partial x} + \frac{\partial \rho vw}{\partial y} + \frac{\partial \rho w^2 + p}{\partial z} = 0 \tag{3.4}$$

$$\frac{\partial E}{\partial t} + \frac{\partial u(E+p)}{\partial x} + \frac{\partial v(E+p)}{\partial y} + \frac{\partial w(E+p)}{\partial z} = 0 \tag{3.5}$$

An assumption is made that the air can be represented as an ideal gas such that the specific heat leads to $\gamma = C_p/C_v$. This assumption also leads to a constant ideal gas constant R which will be important for normalization. A constant γ was chosen of 1.4. This also causes a simple relationship between the energy of the problem and the pressure, density and velocities.

$$E = \frac{p}{\gamma - 1} + \frac{\rho}{2}\left(u^2 + v^2 + w^2\right) \tag{3.6}$$

For simpler analysis the state and fluxes can be stored into vectors so that the Euler equation breaks down into one vector equation

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} = 0, \tag{3.7}$$

Where each vector represents

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{pmatrix}, \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{pmatrix}, \mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \end{pmatrix}, \mathbf{H} = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E+p) \end{pmatrix} \tag{3.8}$$

A common choice for the flux is the Lax-Friedrichs Flux function but this is not the best choice especially when working in CFD. The AUSM$^+$-up for all speeds flux function developed by Meng-Sing Lou[14][15] was chosen due to its capabilities for both high and low speed flows and its dependence on the physics of the problem.

3.1 Normalization

The variables for this research were all normalized by standard values. The assumption of the air being an ideal gas as previously stated allows for the constant specific heat and gas constants. A reference length was chosen which depends on the problem. For the shock tube and Kelvin-Helmholtz instability the length factor can be chosen indiscriminately due to the nature of the problem, but for the vortex problem this reference length plays great weight in the development of the vortex bursts initial condition. Thus the radius core length r_c was chosen. Finally a reference speed was chosen as the speed of sound of stagnant air

which is a constant. This led to the major variables being nondimensionalized as

$$\begin{aligned} x^* &= \frac{x}{r_c}, & y^* &= \frac{y}{r_c}, & z^* &= \frac{z}{r_c}, \\ u^* &= \frac{u}{a_\infty}, & v^* &= \frac{v}{a_\infty}, & w^* &= \frac{w}{a_\infty}, \\ \rho^* &= \frac{\rho}{\rho_\infty}, & p^* &= \frac{p}{p_\infty}, & T^* &= \frac{T}{T_\infty}, \\ c_p^* &= \frac{\gamma}{\gamma - 1}, & e^* &= \frac{e}{a_\infty^2} & \Gamma^* &= \frac{\Gamma}{\Gamma_0} \end{aligned} \tag{3.9}$$

After this point the superscript * will be dropped and all variables unless otherwise specified will be nondimensional.

3.2 AUSM$^+$-up for all speeds

The AUSM flux function family was developed with the goal of building a better flux function that captures the accuracy of flux differencing methods with the efficiency of flux vector splitting methods. This method was advanced substantially in [14][15] becoming first the AUSM$^+$ method and then the AUSM$^+$-up method. The first improvement gave this flux function the ability to exactly capture a shock or discontinuity as well as an improvement in accuracy with an easier adaptation to other conservation laws. The AUSM$^+$-up for all speeds method was then created 10 years later with a goal that as $M \to 0$ the method provides accurate solutions therefore creating a function that works for all speeds improving the AUSM$^+$ even further.

For the development of this method the flux is broken up into a convective and pressure fluxes such that

$$\mathbf{F} = \dot{m}\vec{\psi} + \mathbf{P} \tag{3.10}$$

where

$$\psi = (1, u, H)^T \tag{3.11}$$

Here the left state which corresponds to the state from the left cell will be upheld by a subscript L and the right state will have a subscript R. The mach number M is found

$$\mathrm{M}_{L/R} = \frac{u_{L/R}}{a_{1/2}} \tag{3.12}$$

where $a_{1/2}$ represents a common speed of sound found by

$$a_{1/2} = \frac{a_L + a_R}{2} \tag{3.13}$$

A common speed of sound is very useful because through the use of a common speed of sound the exact capturing of both shocks and discontinuities becomes possible.

The Mach number at the interface is represented by a function $\mathscr{M}_{(m)}$ where m represents the order of the function

$$\mathscr{M}^{\pm}_{(1)}(\mathrm{M}) = \frac{1}{2}(\mathrm{M} \pm |\mathrm{M}|) \tag{3.14}$$

$$\mathscr{M}^{\pm}_{(2)}(\mathrm{M}) = \frac{1}{4}(\mathrm{M} \pm 1)^2 \tag{3.15}$$

$$\mathscr{M}^{\pm}_{(4)}(\mathrm{M}) = \begin{cases} \mathscr{M}^{\pm}_{(1)}(\mathrm{M}), & \text{if } |\mathrm{M}| \geq 1, \\ \mathscr{M}^{\pm}_{(2)}(\mathrm{M})(1 \mp 16\beta \mathscr{M}^{\mp}_{(2)}(\mathrm{M})), & \text{otherwise}, \end{cases} \tag{3.16}$$

here β represents a constant value. With the Mach functions set the Mach number at the interface is then found by applying

$$\mathrm{M}_{1/2} = \mathscr{M}^{+}_{(4)}(\mathrm{M}_L) - \mathscr{M}^{-}_{(4)}(\mathrm{M}_R) + \mathrm{M}_p \tag{3.17}$$

where M_p represents the pressure diffusive terms effects on speed. The pressure diffusive term is represented by

$$\mathrm{M}_p = \frac{K_p}{f_a}\max(1 - \sigma\bar{\mathrm{M}}^2, 0)\frac{p_R - p_L}{\rho_{1/2}a^2_{1/2}}, \quad \rho_{1/2} = \frac{\rho_L + \rho_R}{2} \tag{3.18}$$

where f_a is a scaling function and K_p and σ are constants. The mean local mach number ($\bar{\mathrm{M}}$) is found through

$$\bar{\mathrm{M}}^2 = \frac{(u_L^2 + u_R^2)}{2a_{1/2}^2} \tag{3.19}$$

Which is then used to find a reference mach number

$$\mathrm{M}_0^2 = \min\left(1, \max\left(\bar{\mathrm{M}}^2, \mathrm{M}_\infty^2\right)\right) \quad \in [0,1] \tag{3.20}$$

which gives the scaling function used above

$$f_a(\mathrm{M}_0) = \mathrm{M}_0(2 - \mathrm{M}_0) \quad \in [0,1] \tag{3.21}$$

Using the new Mach number at the interface and the common speed of sound the mass flow rate is found

$$\dot{m}_{1/2} = a_{1/2}\mathrm{M}_{1/2} \begin{cases} \rho_L & \text{if } \mathrm{M}_{1/2} > 0, \\ \rho_R & \text{otherwise} \end{cases} \tag{3.22}$$

The pressure term at the interface is then found by use of a $\mathscr{P}_{(n)}$ function which is of varying degree n like the Mach function previously.

$$\mathscr{P}_{(5)}^{\pm} = \begin{cases} \frac{1}{M}\mathscr{M}_{(1)}^{\pm} & \text{if } |\mathrm{M}| \geq 1, \\ \mathscr{M}_{(2)}^{\pm}[(\pm 2 - M) \mp 16\alpha M \mathscr{M}_2^{\mp}] & \text{otherwise} \end{cases} \tag{3.23}$$

where α is a function based on the scaling function f_a. This function is then used to develop the new pressure term for the function where

$$p_{1/2} = \mathscr{P}_{(5)}^{+}(\mathrm{M}_L)p_L + \mathscr{P}_{(5)}^{+}(\mathrm{M}_R)p_R - p_u \tag{3.24}$$

p_u represents the velocity diffusion term effecting low velocity flow and is

$$p_u = K_u \mathscr{P}^+_{(5)} \mathscr{P}^-_{(5)} (\rho_L + \rho_R)(f_a a_{1/2})(u_R - u_L) \tag{3.25}$$

here K_u is a user defined constant.

The previous variables α and β are then found through

$$\begin{aligned} \alpha &= \frac{3}{16}(-4 + 5f_a^2) \in [-\frac{3}{4}, \frac{3}{16}], \\ \beta &= \frac{1}{8} \end{aligned} \tag{3.26}$$

where β is left as a constant and now α has become a function which differs from the AUSM$^+$ method. Finally the flux is returned as

$$\mathbf{f}_{1/2} = \dot{m}_{1/2} \begin{cases} \vec{\psi}_L, & \text{if } \dot{m}_{1/2} > 0 \\ \vec{\psi}_R & \text{otherwise} \end{cases} + \mathbf{p}_{1/2} \tag{3.27}$$

For supersonic flows the pressure term in the Mach number calculation and the velocity term in the pressure calculation fall away revealing the method to be the same as the AUSM$^+$. Therefore the method is only being adjusted for low speed flows.

3.3 Integration through time

The DG method allows for a superb method for finding the derivatives in the spatial direction but the problem remains with integrating these with respect to time. Before a scheme can be built for this a time step needs to be established. A reasonable time step is the basis for any numerical calculation. Here the time step can be found by

$$\Delta t \sim C \left(\lambda_{max} \frac{N^2}{h} + \|\nu\|_{L^\infty} \frac{N^4}{h^2} \right) \tag{3.28}$$

where λ_{max} represents the maximum characteristic velocity $\lambda_{max} = \sqrt{\gamma p/\rho} + \sqrt{u^2 + v^2 + w^2}$, h is the minimum length, and ν is the maximum applicable viscosity. The maximum viscosity coefficient is be found by using a simple relationship where

$$\nu = \lambda_{max} \frac{h}{N^2} \tag{3.29}$$

With the time step established a system for integrating through time needs to be found. The most common and useful method for doing this is through a Runge-Kutta integration. A total variation diminshing (TVD) Runge-Kutta was used in order to remove any superfluous oscillations caused by the integration through time. This is a common method of applying the Runge-Kutta for a more accurate solution.

For this simulation a 3$^{\text{rd}}$ order TVD Runge-Kutta integration was used such that

$$\begin{aligned} u^{(1)} &= u^n + \Delta t \mathcal{R}\left(u^n\right) \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t \mathcal{R}\left(u^{(1)}\right) \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t \mathcal{R}\left(u^{(2)}\right) \end{aligned} \tag{3.30}$$

where $\mathcal{R}$ represents $\mathrm{d}u/\mathrm{d}t$.

3.4 Artificial Viscosity Sensor

The DG method has a large advantage over traditional FV codes due to it being able to achieve a high order of accuracy. There is a problem though. Inherent in any high order scheme are oscillations created by the Gibb's phenomena. This phenomena can lead to instabilities. Therefore a need is found for a scheme to develop artificial viscosity to be used for discontinuities. Traditional finite volume schemes use two different methods for this implementation. The first is to allow the inherent numerical dissipation of the problem to remove these problems. This is only applicable to linear first order systems and are not useful for solving problems. The second is applied to second order and higher methods which

is accomplished by making the scheme nonlinear. This is done by either applying a limiter to reduce the order of the simulations so that the inherent numerical dissipation to affect the solution or by creating an explicit viscosity term based on the state and applying it. The concept of limiters were dropped in order to allow for a representation of the physics to be the base upon which the viscosity is applied. This is artificial damping.

A simpler way would be use a global viscosity on all cells much like the inherent dissipation present in lower order methods but this can cause the erasing of important small scale physics that should not have been affected if this were a real world experiment. If global viscosity is not useful, other traditional codes apply a type of limiting [17] where the DG method is scaled back in certain areas to lower the order of accuracy and insert some of that numerical dissipation. This is also not a useful method because it lowers the overall accuracy of the solution for those cells even if the cells are small enough to capture the "sharp" features. Therefore a sensor approach is much more useful in that it senses which cells need to have viscosity applied for them to function without any reduction in accuracy for those cells. The current sensor being used was developed by Andreas Klockner [12] as an extension to DG codes and can be used with the Euler equations. Some improvements have been made to this code and will be noted below.

The governing equations are listed below where the right hand side contains an artificial viscosity with ν representing the viscosity coefficient of unknown value.

$$\partial_t \rho + \nabla_x \cdot (\rho \mathbf{u}) = \nabla_x \cdot (\nu \nabla_x \rho), \tag{3.31}$$

$$\partial_t (\rho \mathbf{u}) + \nabla_x \cdot (\mathbf{u} \otimes (\rho \mathbf{u})) + \nabla_x p = \nabla_x \cdot (\nu \nabla_x (\rho \mathbf{u})), \tag{3.32}$$

$$\partial_t E + \nabla_x \cdot (\mathbf{u} (E + p)) = \nabla_x \cdot (\nu \nabla_x E) \tag{3.33}$$

In the development of this sensor density was used as the sensing variable upon but it is a choice of the programmer in selecting which variable to sense upon. Here q will represent the chosen state.

Once the choice is made the Vandermonde matrix is applied to transform from a nodal basis into a modal basis. Here $\hat{q}$ represents the given state after having been transformed into a modal form. It is assumed that the decay can be approximated as

$$|\hat{q}_n| \sim cn^{-s} \tag{3.34}$$

which by taking the logarithm

$$\log|\hat{q}_n| \sim \log(c) - s\log(n) \tag{3.35}$$

gives an algebraic system of equations to solve. The coefficients are found by a least squares fit through utilization of the normal equations where

$$A^T Ax = A^T b \tag{3.36}$$

$$A = \begin{bmatrix} 1 & -\log(1) \\ 1 & -\log(2) \\ \vdots & \vdots \\ 1 & -\log(n) \end{bmatrix}, \quad b = \begin{Bmatrix} \log|\hat{q}_1| \\ \log|\hat{q}_2| \\ \vdots \\ \log|\hat{q}_n| \end{Bmatrix}, \quad x = \begin{Bmatrix} \log(c) \\ s \end{Bmatrix} \tag{3.37}$$

s represents the decay coefficient and is key in determining how much viscosity should be applied to a cell. The problem is this can be misled very easily by white noise in the data or for a "kink" where certain modes fall to zero leading to the data being misinterpreted and giving a very high decay coefficient value.

To help fix this two different fixes were applied to the modes before the normal equations were solved. The first fix was by applying skyline pessimization which changes the modes so that they follow a monotone mode profile by eliminating spurious nodes. Skyline

pessimization is implemented such that

$$\overline{q_n} := \max_{i\in\{\min(n,N_p-2),\ldots,N_p-1\}} |\hat{q}_i| \quad \text{for } n \in \{1,2,\ldots,N_p-1\} \tag{3.38}$$

Thus producing a new set of modal coefficients.

The small white noise perturbations are removed by adding a representation of scale to the model in the form of baseline decay

$$|\hat{b}_n| \sim \frac{1}{\sqrt{\sum_{i=1}^{N_p-1} \frac{1}{n^{2N}}}} \frac{1}{n^N} \tag{3.39}$$

$$\sum_{n=1}^{N_p-1} |\hat{b}_n|^2 = 1 \tag{3.40}$$

Which is then added to the original modal array by

$$|\tilde{q_n}|^2 := |\hat{q}_n|^2 + \|q_N\|^2_{L^2(\mathrm{D}_\kappa)} |\hat{b}_n|^2 \quad for \in \{1,\ldots,N_p-1\} \tag{3.41}$$

This addition drowns out the floating point white noise in the system allowing for only the data to be examined.

With the value of s determined it is noted that an analogy can be shown between Fourier decay and this modal decay leading to the knowledge that $s \approx 1$ for a discontinuous solution, $s = 2$ for a C^0 solution, $s = 3$ for a C^1 solution, etc. This leads to the artificial viscosity being determined by

$$\nu(s) = \nu_0 \begin{cases} 1 & s \in (-\infty, 1)\,, \\ \frac{1}{2}\left(1 + \sin\left(-\left(s-2\right)\pi/2\right)\right) & s \in [1,3]\,, \\ 0 & s \in (3,\infty). \end{cases} \tag{3.42}$$

where ν_0 is the maximum viscosity coefficient discussed previously.

Their method only included details up to the second dimension so in order to use Klöckner's sensor in three dimensional space some new implementations had to be added. These changes caused both an increase in the speed of the sensor by augmenting the process where the inverse Vandermonde matrix was applied. A change was implemented that moved the two dimensional application from using the Vandermonde based off of a two dimensional system to that based off of a one dimensional system. The state vector q was split from a vector of N_p points to a N_{fp}xN_{fp} matrix allowing for application of the Vandermonde matrix based off of a one dimensional system. This was done through a manipulation of pointers in fortran 90. The norm was also no longer used because as the number of dimensions increased the norm increased and with this increase the norm could buffer out some of the important data by making the baseline decay normalization too big. Another problem exists because for some data like, axial velocity in the burst problem, the norm would go to zero removing a key piece of the sensor's development. A "fix" was made to allow for sensing on this data and is implemented in Chapter 4.

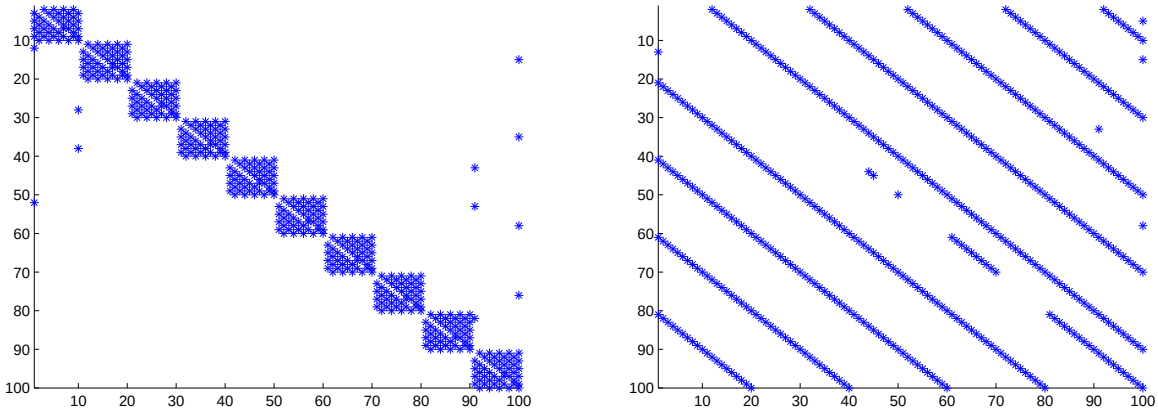
3.5 Improving the Efficiency of the Code to Increase its Speed

With all these other implementations two small but major changes were implemented in order to drastically increase the speed of the code as it was developed and moved from one dimension to a three dimensional problem. These two ideas were the creation of sparse matrices for stiffness and lifting matrices as well as the implementation of parallelization.

3.5.1 Sparse Matrix

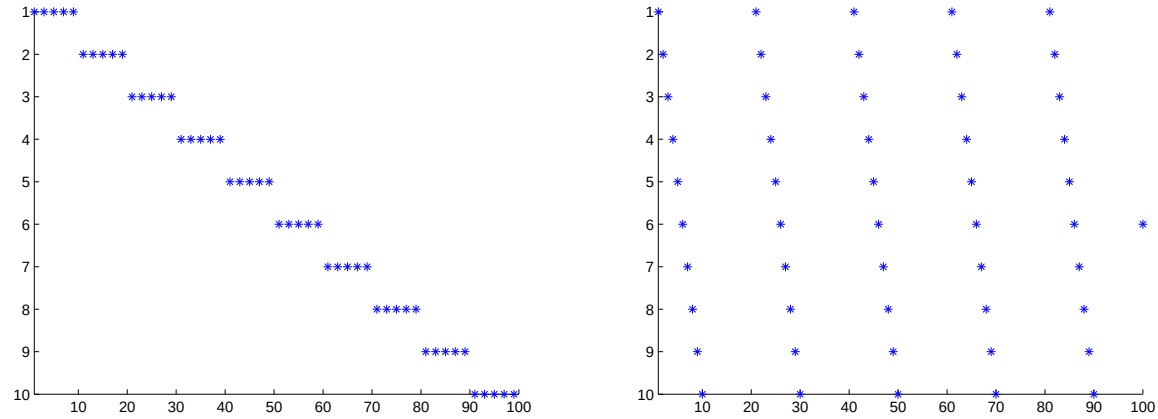
The point of sparse matrix multiplication is to remove any cells that are zero or roughly equate to zero in terms of double precision for computers. These matrices had a vast number of points where the stiffness matrix for two dimensional problems had dimension $[N_{fp}^2, N_{fp}^2]$ with only N_{fp}^3 points having values. Here N_{fp} represents the number of points to a edge or $N_{fp} = N + 1$. In terms of numbers this lays out to a 9^{th} order polynomial having a matrix

of 10,000 points but only 1000 points having needed data. This data emerges into patterns of numbers as seen in the stiffness and lifting matrices in the figures below.



(a) S_x (b) S_y

Figure 3.1: Location of all non zero values in stiffness matrices



(a) L_{x_a} (b) L_{y_a}

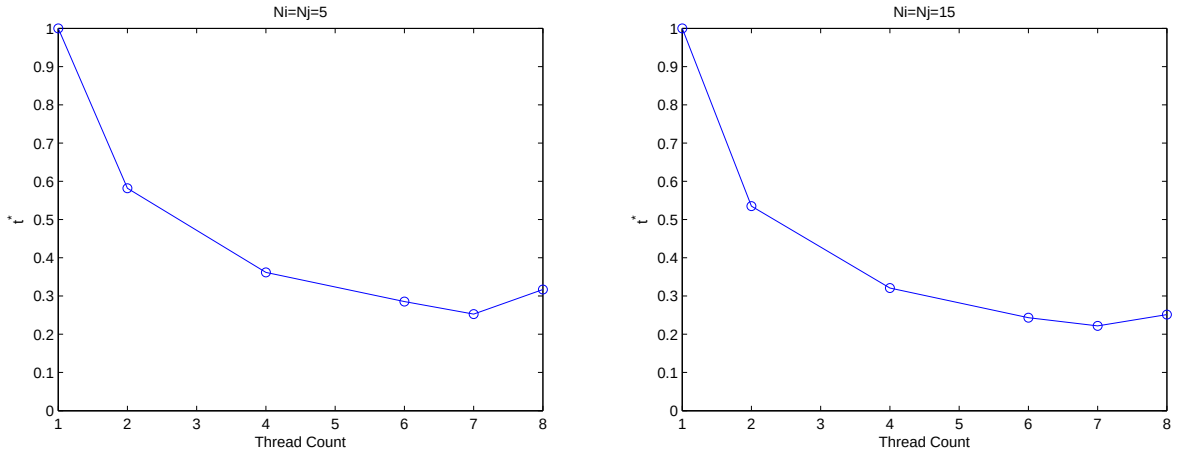
Figure 3.2: Location of all non zero values in lifting matrices

This is also only for two dimenisonal problems, the number of wasted data points, thus leading to pointless calculations, only grows larger when applied to three dimensional problems.

Therefore a method had to be and was developed in order to take advantage of this and speed up the code. A module was created in Fortran that searched the matrix for all pertinent data that resided above a tolerance and stored that into a new data array. This allowed for a speed up of the code due to a smaller flop count in the multiplication.

3.5.2 Parallel Processing

To obtain a faster code parallel processing was implemented such that the full power of the processor could be achieved. Parallel processing occurs when a series of loops is compressed and then broken up and sent into a user defined number of threads in a processor or graphics processing unit. For these simulations a built in method called openmp was utilized to process the parallized code. A simple experiment was done in order to determine the optimal number of threads that should be implemented for the xenon processor in the workstation used. The data can be seen in the figure below.



(a) $N_i = N_j = 5$ cells

(b) $N_i = N_j = 15$ cells

Figure 3.3: Wall time for utilizing multiple threads for the Isentropic Vortex Test Case.

This data shows that seven threads gives an optimal solution but this could also be weighed by the fact that during these simulation the computer had a user logged into it. Nothing was running but the eighth thread was occupied with the operating system and

running the standard desktop functions slowing down the system. Therefore if the user is logged in seven threads is optimal but for use on something like a cluster the maximum number of threads or 8 will be more optimal.

Chapter 4

Sensor Improvement

With the substantial cost in terms of time of the burst problem along with the different improvements made to the artificial viscosity sensor multiple test cases were run for validation to lead to the full scale problem. Previous research accomplished at Auburn University by Reitz [19] and Favors [7] provided insight into some 2D applications. For Reitz it was the implementation of the 9^{th} order polynomials as well as the importance of domain size being most favorable for speed and accuracy while Favor's thesis showed some of the benefits that comes with the Klöckner sensor used here versus an entropy physics based model.

4.1 Test Cases

For the development of the system two separate two dimensional test cases as well as a three dimensional test case were examined with the sensor. These cases are a shock tube adjusted to two dimensional space and the Kelvin-Helmholtz phenomena. Then before the implementation of the full three dimensional problem the shock tube was extrapolated to three dimensions to validate the changes that were made to the sensor as well as work with some new changes specifically in how the baseline decay is applied.

4.1.1 Kelvin-Helmholtz

The Kelvin Helmholtz phenomena was used as a simple 2D model in order to validate the changes as well as show an increase in the speed of the new implementation of the sensor. Another advantage to using this as a check, is that the Kelvin-Helmholtz phenomena is a naturally occurring phenomena that can occur in wakes giving one more example of the DG method modeling wake flow components. This test case also gives a special situation where

the sensor can be checked against numerical instabilities that are developed through time and are not abrupt like a shock is.

In creating the initial state $\mathbf{U}$ constant values are assumed for $\rho = 1$ and $p = 1/\gamma$. Constants $\epsilon = \pi/15$, $\delta = 0.1/4.9$, and $\mathrm{M} = 0.25$ are used in the development of the velocity profile for the system which is found through

$$s = \begin{cases} \tanh\left(\frac{3\pi/2 - y}{\epsilon}\right), & y > \pi \\ \tanh\left(\frac{y - \pi/2}{\epsilon}\right), & else \end{cases} \tag{4.1}$$

$$\begin{aligned} u &= 0.5\,(s+1)\,\mathrm{M} \\ v &= \mathrm{M}\delta\,(1.\sin(1.\,x/3.) - 1.\sin(2.\,x/3.) + 1.\sin(3.\,x/3.)) \end{aligned} \tag{4.2}$$

The vertical velocity here is developed with the purpose of perturbing the first three modes to allow for consistently reproducible simulations in order to compare results back and forth. A fifth state is added to the governing equation concerning the concentration factor s. This state is used both in the sensor and in the plotting of the initial conditions as seen in Figure 4.1. This allows for an easy comparison to other codes and experiments. The bounds for this problem are $x \in [0, 4\pi]$ and $y \in [-\pi, 3\pi]$.

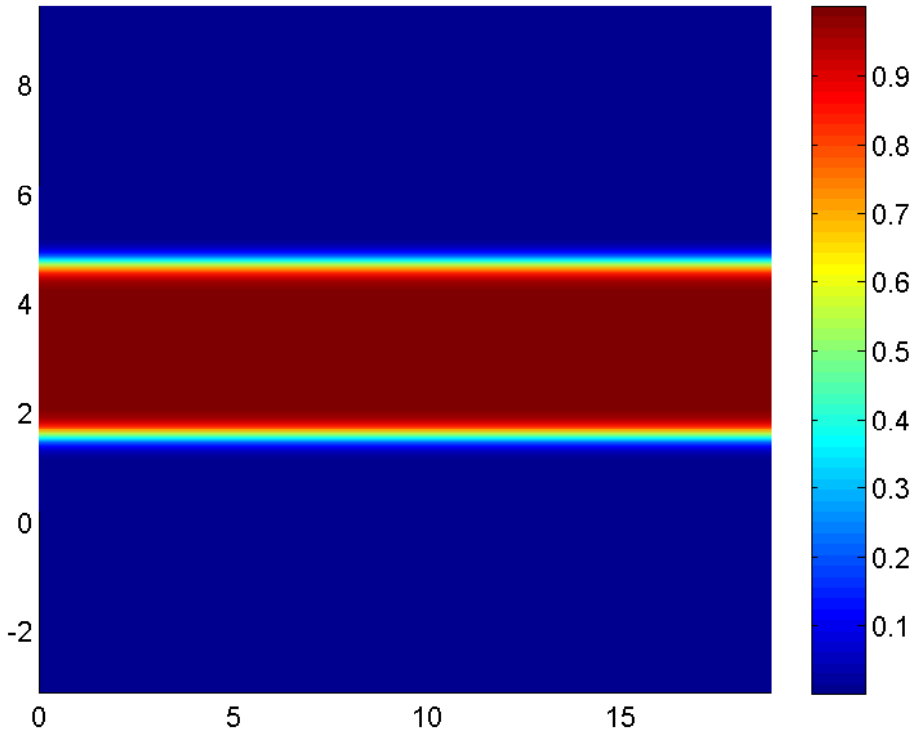


Figure 4.1: Initial condition showing the concentration s

4.1.2 Sod Shock Tube

The burst problem also deals with a direct shock like discontinuity when the vortices core radius changes due to the burst. Therefore an examination of a shock tube provides ample benefits in how the sensor handles such an abrupt change in the state. A shock tube is established in experiments by creating a cylinder with two different states for pressure and density on either side of a burst disk. At time t=0 the burst disk is then popped and the gases mix sending a shock wave and other cascading effects through the system. This problem from a CFD sense is a simple one dimensional problem that can easily be extrapolated to three dimensional space.

The problem here examines the common test case of the Sod shock tube and is set up for multiple dimensions by choosing a single direction and then declaring the location of the

burst disk between the two ends of the domain to exist in this direction. For this simulation a bounds of $[-1, 1]$ for both x and y. The computational burst disk was placed at 0 which is the center. The values on either side are

$$\rho = \begin{cases} 1. & x \leq 0. \\ 0.125 & x > 0. \end{cases} \tag{4.3}$$

$$p = \begin{cases} 1. & x \leq 0. \\ 0.1 & x > 0. \end{cases} \tag{4.4}$$

and an image of the initial condition can be found in Figure 4.2.

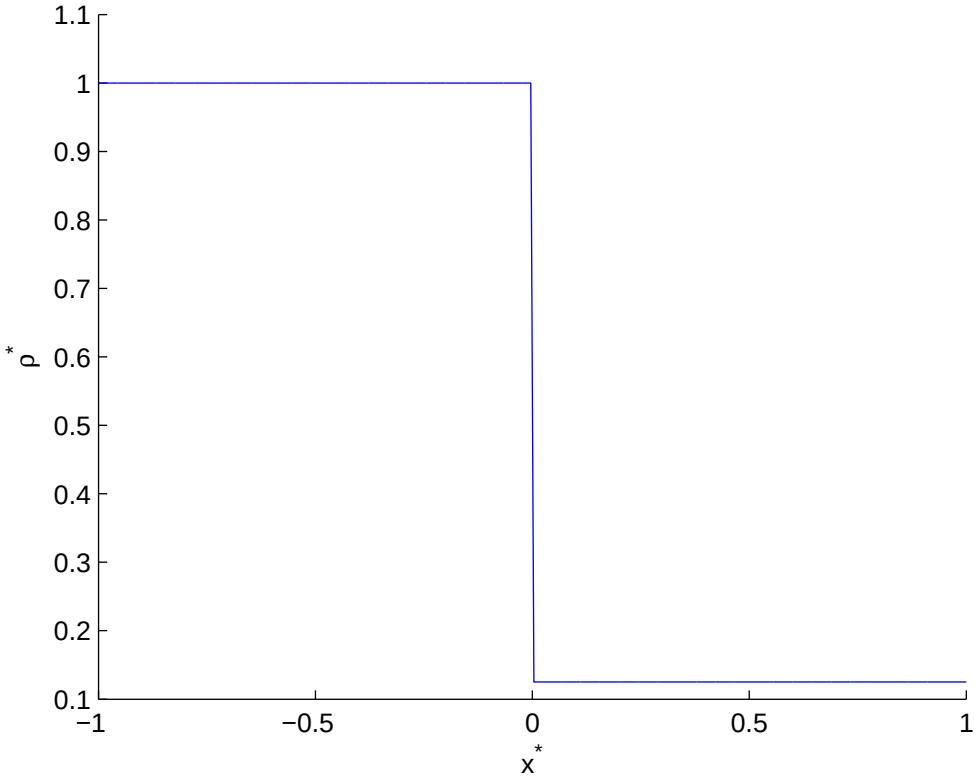


Figure 4.2: Initial Condition of density (ρ)

With the sensor fully validated the shock tube was then expanded into three dimensions in order to troubleshoot as well as develop the extrapolation of the sensor into three dimensional space. The same format for the development of the initial condition was followed for converting the two dimensional shock tube to a three dimensional shock tube.

4.2 Results for Sensor Validation

4.2.1 Sensor Validation in Two Dimensions

The modal sensor is a very useful tool, but for an optimal use in three dimensions it needs to be retooled to increase it's speed. This is an important concept when running three dimensional simulations which are already very costly concerning time. Before moving into the three dimensional problems, the sensor was first altered in the two dimensional space in order to troubleshoot as well as see if it altered the sensor's abilities. These changes can be found in algorithms 1 and 2. Here the original algorithm, algorithm 1, has a matrix multiplication that costs N_{fp}^4 flops. This is due to it involving a matrix multiplication of a N_{fp}^2xN_{fp}^2 matrix to a N_{fp}x1. The method is altered by using the inverse of the Vandermonde matrix based off of a one dimensional system instead of the current two dimension system. This Vandermonde matrix was then applied by matrix multiplication in the x or y direction while looping through all the nodes. This resulted in 2 N_{fp}xN_{fp} by N_{fp}x1 calculations inside of a loop over N_{fp} creating a flop count of $2 * N_{fp}^3$ for the two dimensional case. For the 3D case the savings are bolstered even higher. This case used to cost flops on the order of N_{fp}^6 but with the use of the smaller Vandermonde matrix it only costs $3 * N_{fp}^4$ flops. This translates to a savings in two dimensional of $N_{fp}/2$ and savings in three dimensional space of $N_{fp}^2/3$ flops which shows exponential savings between the two cases.

With proof that the sensor provides a substantial increase in the speed of the simulation the validity of the new application of the modal sensor was checked using the Kelvin-Helmholtz problem. This was done to prove the modes were still the same. The kinetic energy in the Kelvin-Helmholtz instability is a constant. Therefore the kinetic energy was

integrated using the gauss-lobatto quadrature and then was normalized by the initial kinetic energy. This was plotted in Figure 4.3. Global viscosity, or the application of viscosity to all locations, was also plotted to show how much more effective using a sensor is. The figure shows both the new approach and old approach giving an identical integrated kinetic energy over time while providing leaps and bounds better results than a global viscosity method proving the validity of the sensor while also showing how it obtains similar results.

The sensor was then compared as it activated in both x and y direction as seen in Figures 4.4 through 4.7. As can be seen in the figures the sensor turned on in the same spots with the exact same weight for both the new and old implementations which aligns with information found in Figure 4.3. This all logically follows since the modes are the same regardless of the method for finding them.

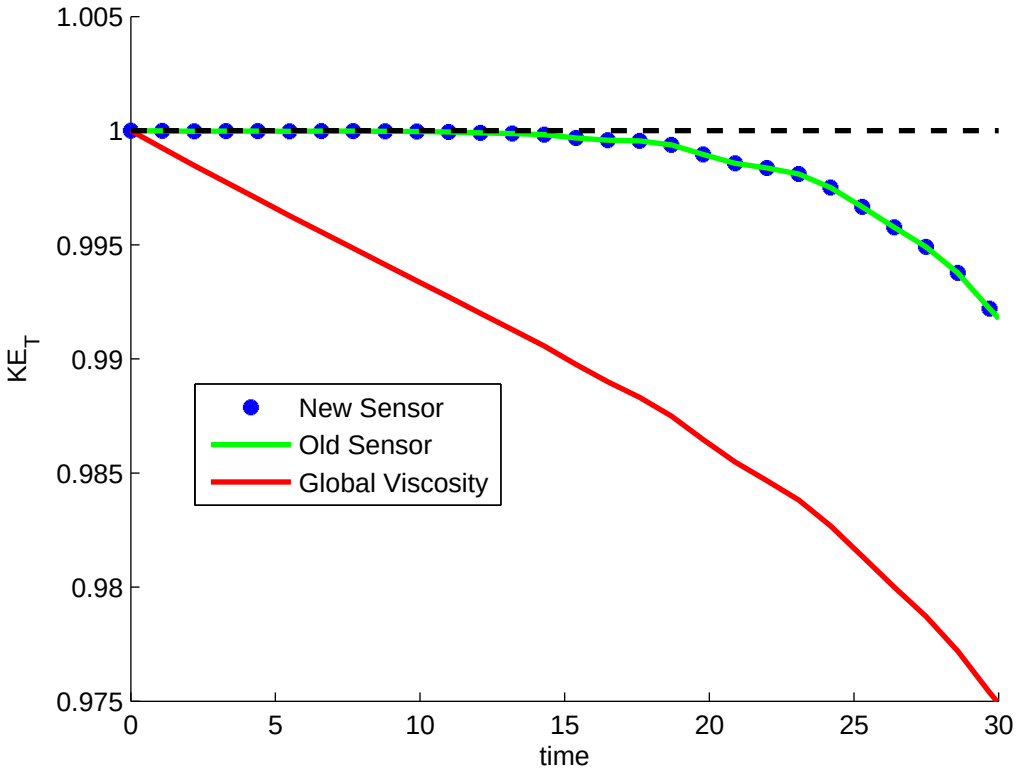
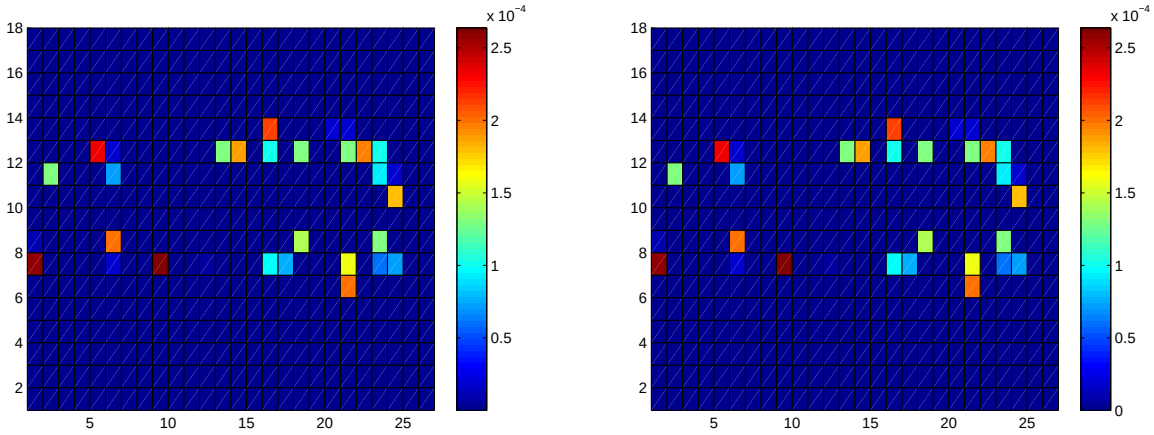
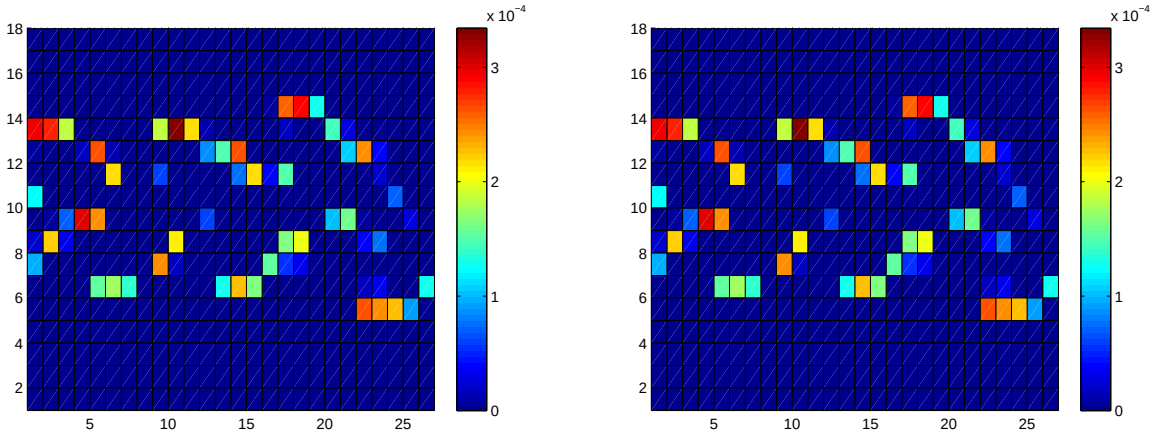


Figure 4.3: Total kinetic energy over time



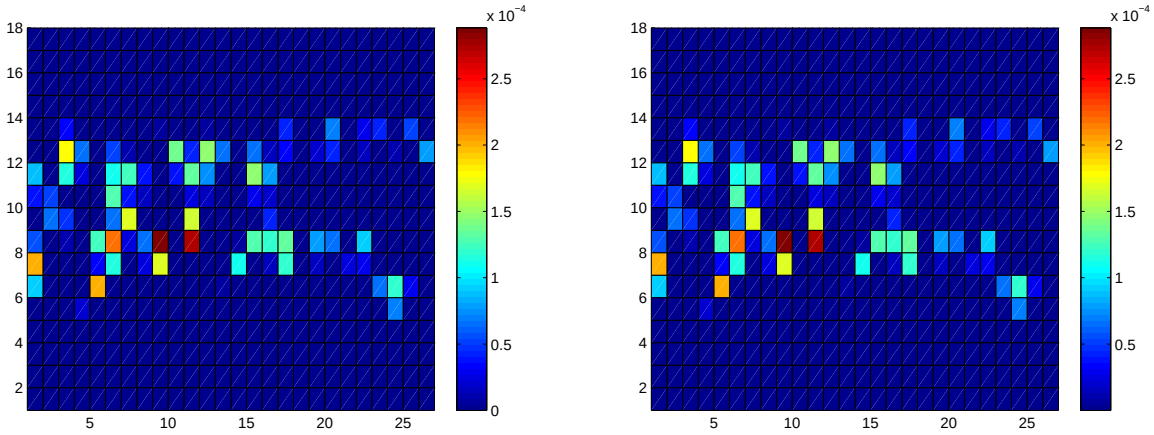
(a) e_x New (b) e_x Old

Figure 4.4: Sensor in X Direction at time 49.49 s



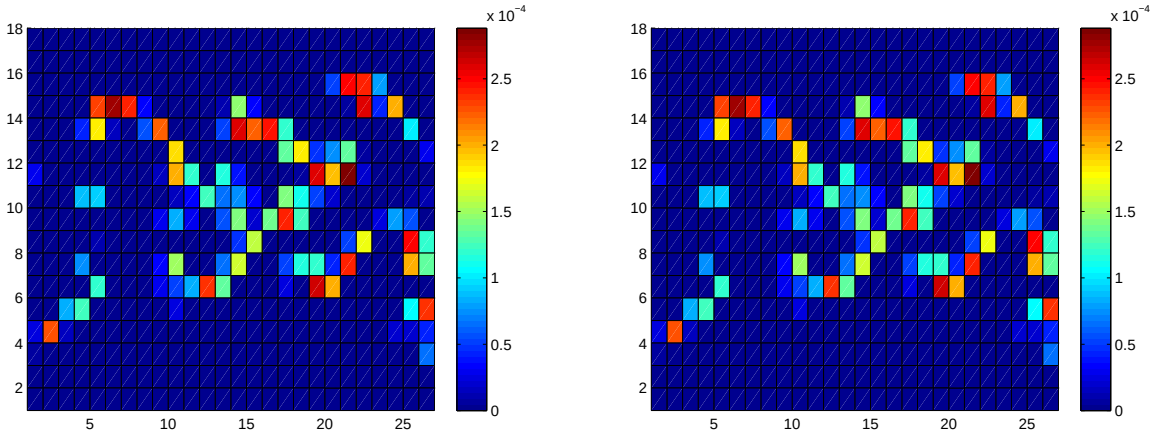
(a) e_y New (b) e_y Old

Figure 4.5: Sensor in Y Direction at time 49.49 s



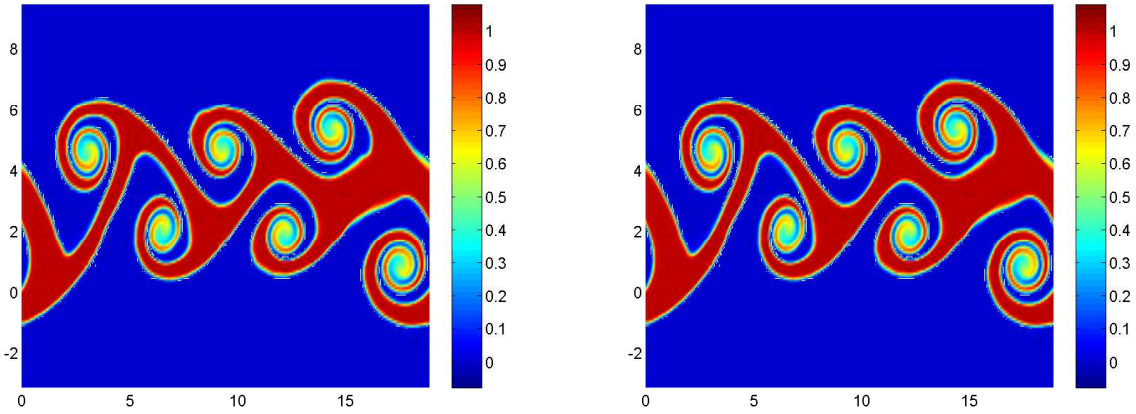
(a) e_x New (b) e_x Old

Figure 4.6: Sensor in X Direction at time 76.99 s



(a) e_y New (b) e_y Old

Figure 4.7: Sensor in Y Direction at time 76.99 s



(a) Kelvin-Helmholtz New

(b) Kelvin Helmholtz Old

Figure 4.8: Comparison of the two states at time 76.99 s

With validation that the new implementation returns the same amount of artificial viscosity for both implementations a cost comparison was done by utilizing both the Sod shock tube and the Kelvin-Helmholtz phenomena. Tables 4.1-4.3 show the new implementation of the sensor saves time for each case. In each of these tables the total time for running each simulation is what is recorded. Tables 4.2 and 4.3 also compare an increase in the order of the polynomial and how that affects the new method. For each fo the sod shock tube test cases a constant Δt was used in order to compare the results for the varying polynomial order. For the data stored in Table 4.2 the Δt was 0.00005 and for Table 4.3 it was 0.0002. For the Kelvin-Helmholtz there is a 4.0858% savings in time while for the Sod shock tube there is a 3.9611% savings in wall clock time for a ninth order polynomial. This shows a decrease in time but questions remain due to the small difference when comparing the sensor for the test cases of table 4.2. Based on polynomial order the decrease in time varies from 5.318 s. Therefore with the domain specifications of $N_i = 99\, N_j = 1$ the order of the polynomial was increased in order to judge whether or not the method did allow for more savings as polynomial order increased. For the tenth order polynomial the savings amounted to 7.278 s or 5.10 %, the eleventh order polynomial is 11.015 s or 6.24 %, the twelfth order polynomial

is 15.873 s or 7.83 %, and the thirteenth order polynomial gives 22.17 s or 9.06 % in savings. This proves that as the order of the polynomial is increased the savings also increase. Since the sensor is implemented for each cell and the new implementation saves on the order of $N_{fp}/2$ flops per iteration the number of cells was increased to $N_i = 50\, N_J = 50$ in order to see how substantial the savings can be for more realistic two dimensional meshes.

The results for these new runs can be seen in Table 4.3. Here there is a savings in time of 14.256 s or 2.20% for the ninth order polynomial, 30.507 s or 3.79 % for the tenth order polynomial, 41.649 s or 4.09 % for the eleventh order polynomial, 68.24 s or 5.58 %for the twelfth order polynomial, and 96.433 s or 6.31 % for the thirteenth order polynomial. This proves the trend found in the previous shock tube experiment where as the order of the polynomial is increased there are more savings from the augmentation to the sensor. The decrease in the percentage of time can be attributed to the increase in mesh size and the sensor not being the most costly part of the simulations.

Algorithm 1 Old Sensor Implementation

1: **for** $i = 1, N_i$ **do**
2: **for** $j = 1, N_j$ **do**
3: $Q = U(:, 5, i, j)/U(:, 1, i, j)$
4: $Q = \text{abs}(\text{matmul}(invV, Q(:, 1)) + 1.0e - 17)$ ▷ N_{fp}^4 flops
5: **for** $iii = 1, N_{fp}$ **do**
6: $qx(iii) = \text{sqrt}(\text{sum}(Q(K_{cell}(:, iii), 1) * *2.))$
7: $qy(iii) = \text{sqrt}(\text{sum}(Q(K_{cell}(iii, :), 1) * *2.))$
8: **end for**
9: ⋮
10: **end for**
11: **end for**

Algorithm 2 New Sensor Implementation

1: **for** $i = 1, N_i$ **do**
2: **for** $j = 1, N_j$ **do**
3: call wrapper$(din, U, uhat)$
4: $u_m =$ apply-along-all$(din, invV1D, uhat)$ ▷ $2 * N_{fp}^3$ Flops
5: $u_m = u_m * u_m$
6: $qx1 =$ sqrt(sum$(u_m, 2)$)
7: $qy1 =$ sqrt(sum$(u_m, 1)$)
8: $\vdots$
9: **end for**
10: **end for**

Sensor Implementation	Wall Clock Time
New Method	545.3109 s
Old Method	568.5409 s

Table 4.1: Kelvin Helmholtz, N=9, Ni=27, Nj=18

	Wall Clock Time				
Sensor	N=9	N=10	N=11	N=12	N=13
New Method	107.232 s	135.144 s	165.39 s	204.752 s	222.661 s
Old Method	112.55 s	142.422 s	176.405 s	220.393 s	243.831 s

Table 4.2: Sod Shock Tube, Ni=99, Nj=1

	Wall Clock Time				
Sensor	N=9	N=10	N=11	N=12	N=13
New Method	634.739 s	775.489 s	977.014 s	1154.97 s	1431.889 s
Old Method	648.995 s	805.996 s	1018.663 s	1223.21 s	1528.322 s

Table 4.3: Sod Shock Tube, Ni=50, Nj=50

4.2.2 Implementation into Three Dimensions

With both the validity of the new approach to the modal sensor proven as well as the savings shown to be substantial, the next step was to extrapolate the sensor into three dimensional space. When applying the sensor to the vortex bursting problem the sensor repeatedly failed due to it not activating fast enough on the data. Both density and energy were sensed to no avail. Therefore the sensor was further augmented to look for the highest value over all the states and apply that to the relative cell in all 3 directions. A problem emerges when applying this method though due to the norm for the state ρw will be zero, therefore the sensor may turn on due to the inherent floating point noise. This can " wash out" important aspects fo the physics of the problem. This meant that a new method for finding the stabilizing factor was needed for the application of the baseline decay to make the sensor more robust on all states even if it removed some of the simplicity of the sensor. To validate this new approach the Sod shock tube was returned to due to its simplicity as well as the speed with which the problem can be run.

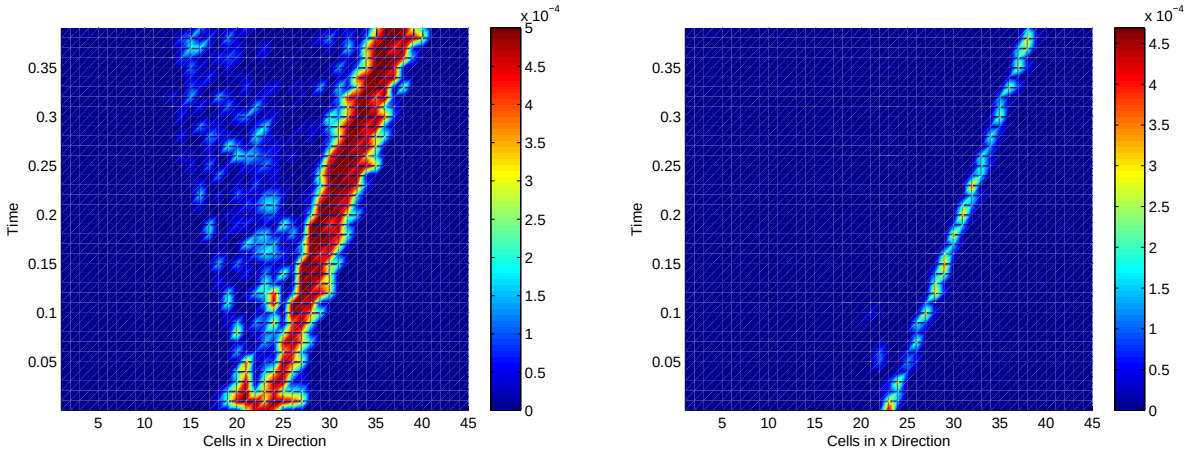
The exact, analytical answers for the Sod shock tube problem were found from a NASA website[20] and used to tabulate RMS errors for each simulation. The new scale for the norm was originally developed as something manually entered by the user for the simulations. For these simulations represented in Figures 4.9-4.11 the norm for each state can be found in the following vector $y_{norm} = [3.0, 0.005, 0.005, 0.005, 5.0]$. The values for states two through four

were chosen in order to see the level of impact the velocity would have. The values for the first and fifth state were chosen to be close to their norm values for a one dimensional case.

Returning to the problem the test cases were run with an activation just on density and then on all states in order to provide a good comparison of the data. Table 4.4 shows for both cases an RMS error of less than 0.02 was found. Figure 4.12 shows that by sensing on the whole state the results are increasingly smoothed over just sensing on one state due to the increase in the activation of the sensor. This can be further seen from Figures 4.9-4.11 which depicts a comparison of the sensor being turned on through time as the shock is propagated in any of the three directions. Image (b) in each of the Figures shows that the sensor actively tracks just the shock and keeps a good marker on it when just sensing on density while Image (a) in the Figures shows a more liberal application of the viscosity over a wider range of data.

Values Sensed On	RMS Error
Sense on ρ	0.0078
Sense on all states	0.0142

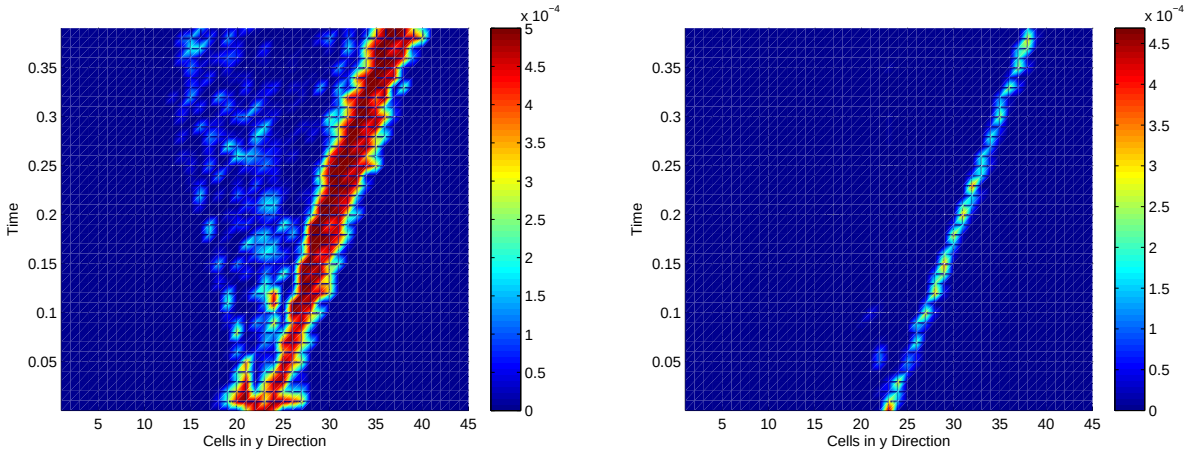
Table 4.4: RMS Error for Sod Shock Tube 3D



(a) X Direction All

(b) X Direction Rho

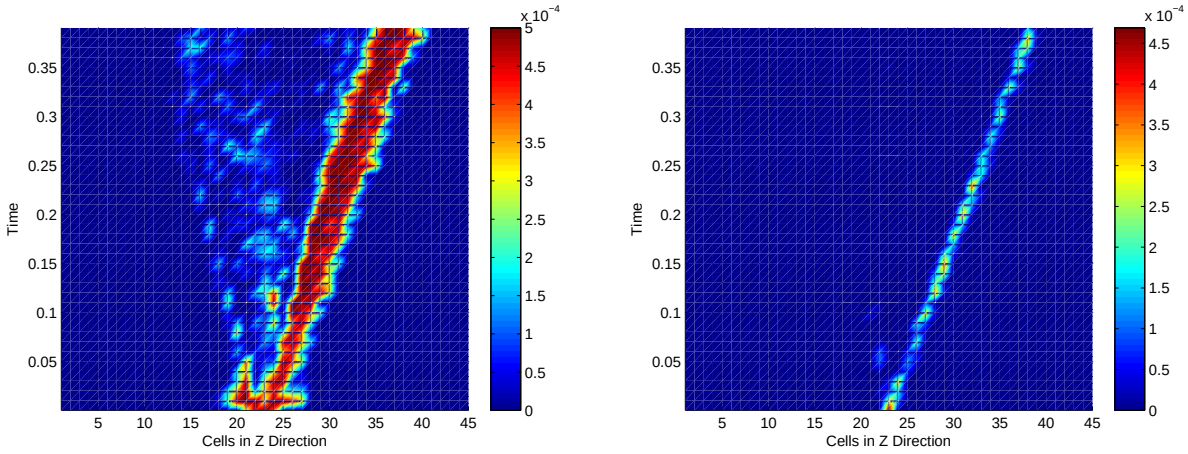
Figure 4.9: Sensor through time X



(a) Y Direction All

(b) Y Direction Rho

Figure 4.10: Sensor through time Y



(a) Z Direction All

(b) Z Direction Rho

Figure 4.11: Sensor through time Z

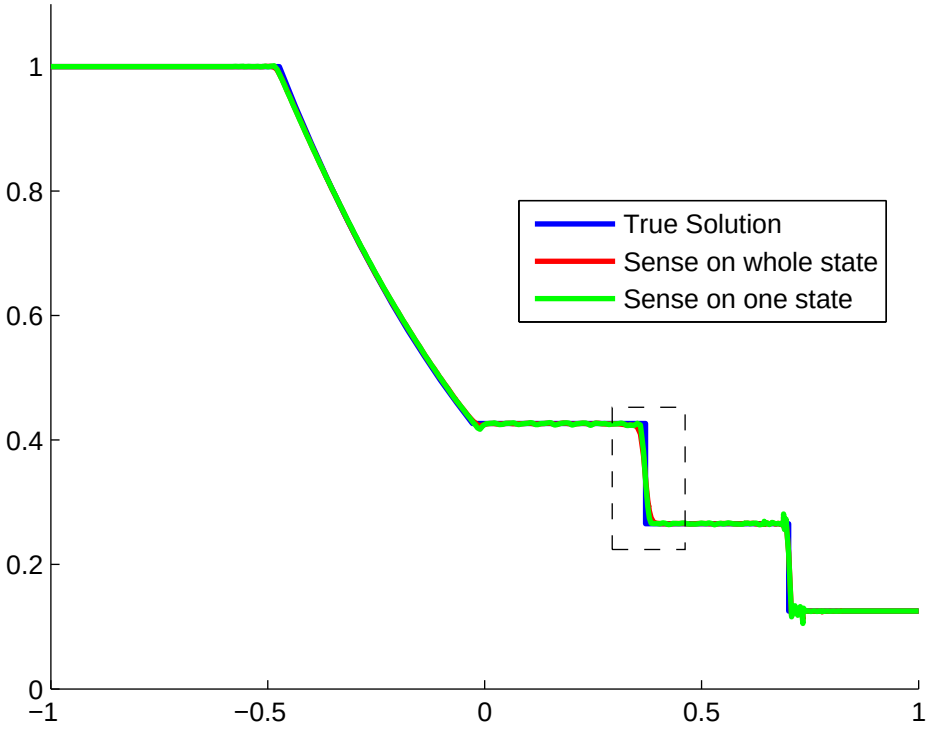


Figure 4.12: Comparison of Sensor using all states or one

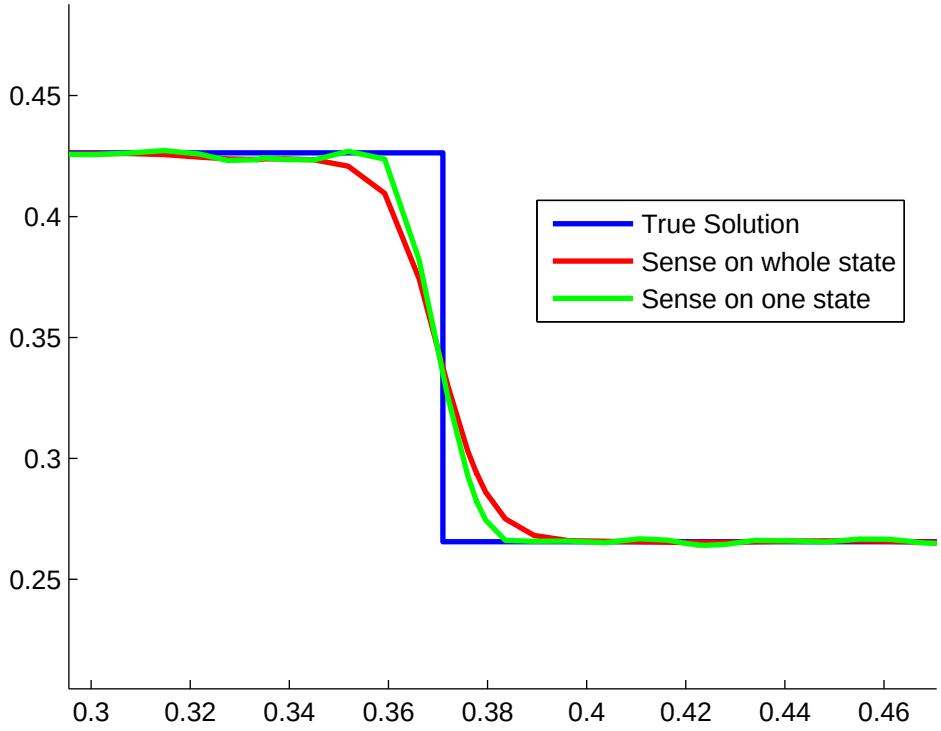
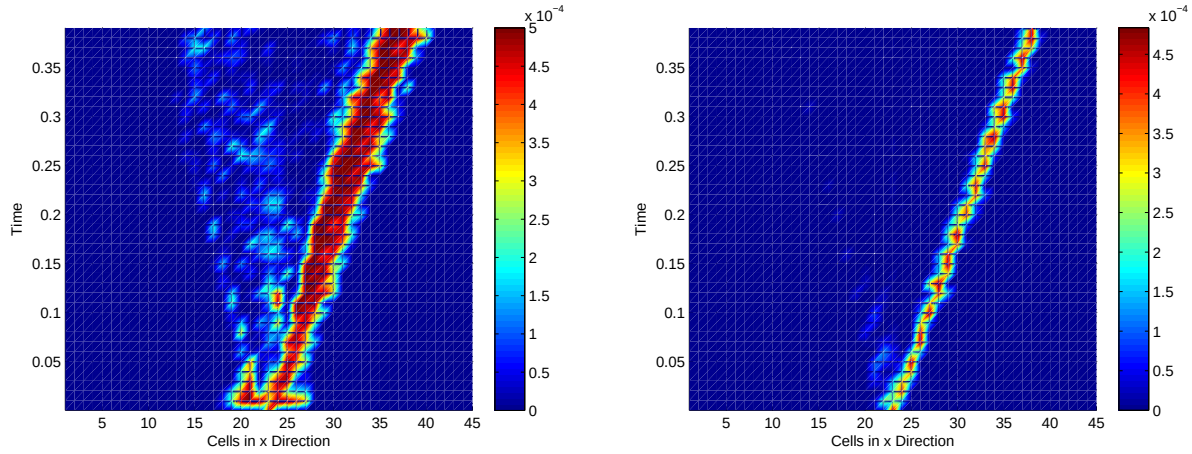


Figure 4.13: Zoomed in comparison of the baseline decay variation

As previously stated these simulations were all done with a predetermined scale of $y_{norm} = [3.0, 0.005, 0.005, 0.005, 5.0]$. This raised a question though of whether or not this was the best choice. The simulation was then rerun with a new scale of $y_{norm} = [3.0, 0.5, 0.5, 0.5, 5.0]$. A comparison of the shock tube sensing on all five state can be seen in Figure 4.14. These images show the sensor being activated much more sparsely due to the higher value on the scale for the velocity states. This follows the logic of the scale being used to balance the baseline decay to remove floating point noise. The lower the value for the norm the higher chance there is for noise to affect the sensor. Table 4.5 validates the previous reasoning by showing a much smaller RMS value for the new implementation of the scale versus the old one and being much more in line with RMS value for the sensor only utilizing density.



(a) Norm value 0.005

(b) Norm value 0.5

Figure 4.14: Comparison of different preselected weighting vectors

Values Sensed On	RMS Error
Velocity norm 0.005	0.0142
Velocity norm 0.5	0.0088

Table 4.5: RMS Error for Sod Shock Tube 3D

All this bears the question of how to derive a system to accurately predetermine values for this new scale. As shown when discussing the sensor a value for s of 1 or lower leads to the maximum application of artificial viscosity. Whereas a value for s of 3 lead to the least amount of artificial viscosity applied. Any values of s over 3 causes no artificial viscosity to be applied. The question is then posed that for basic phenomena such as floating point noise or a step function what is the maximum value that can be used to trip the sensor for either of these situations.

The previous method for deriving the scale was also found to cause different values for a step function based at 0 and a step function based at any other height. In order to incorporate this the step function to be used will be based at one. In order to determine this the first thing that was done was to choose a set of s values to examine. For this research

the values of $s = [1, 2, 3]$ were chosen. Then three separate polynomial orders were chosen to be used as a framework for the solution. The polynomial orders chosen were $x = [5, 9, 15]$. Initial guesses were found for these polynomial orders in order to obtain the desired s value. These values can be seen in Equation 4.5 where the rows correspond to the distinct s values and the columns to the polynomial orders.

$$y = \begin{bmatrix} 0.0034 & 0.012 & 0.066 \\ 0.019 & 0.22 & 6.6 \\ 0.071 & 1.72 & 180 \end{bmatrix} \tag{4.5}$$

With the initial conditions built a MATLAB script was run that created an initial guess by utilizing this data and an intrinsic interpolation function in MATLAB. Then a minimizing function was used called FZERO. The function being fed to FZERO would take the polynomial order and determine an initial condition based off of the desired test case and then run the sensor on it. With the value of s obtained the function would then determine the cost function with $s_{desired} - s_{sensor}$. The system would continually run until it achieved an s value within a tolerance to the desired value. These values for both the step function as well as noise can be seen below in Figures 4.15 and 4.16. In these figure's the average value needed to activate the sensor was divided by the amplitude to allow for an easy application to different problems.

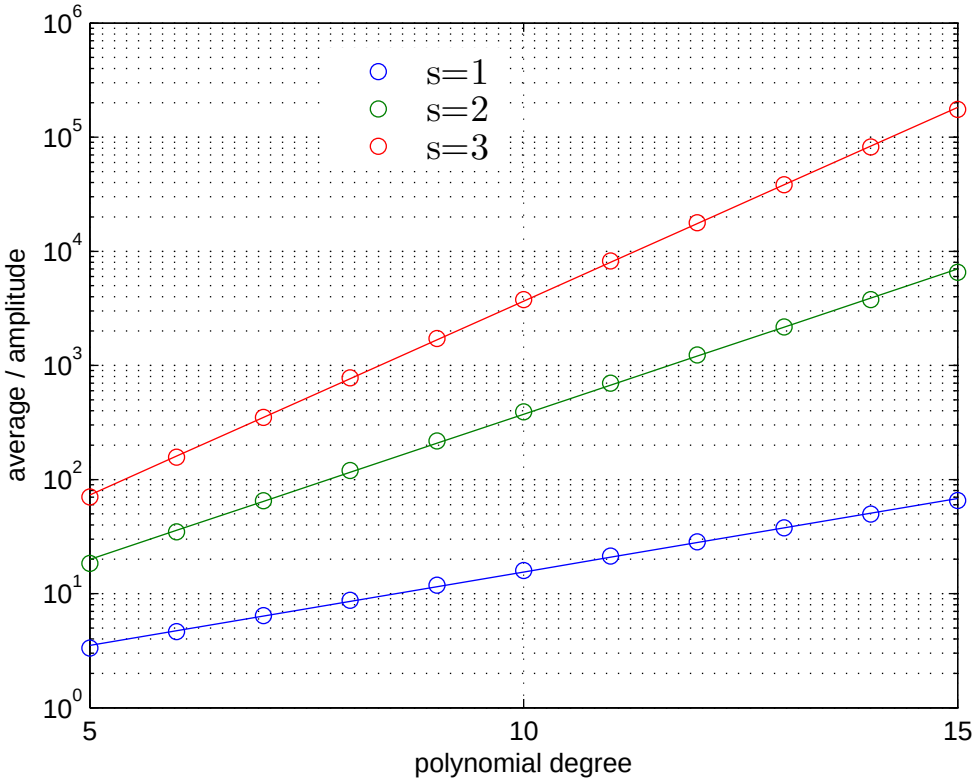


Figure 4.15: The minimum necessary normalized average to trip a specific s value for noise

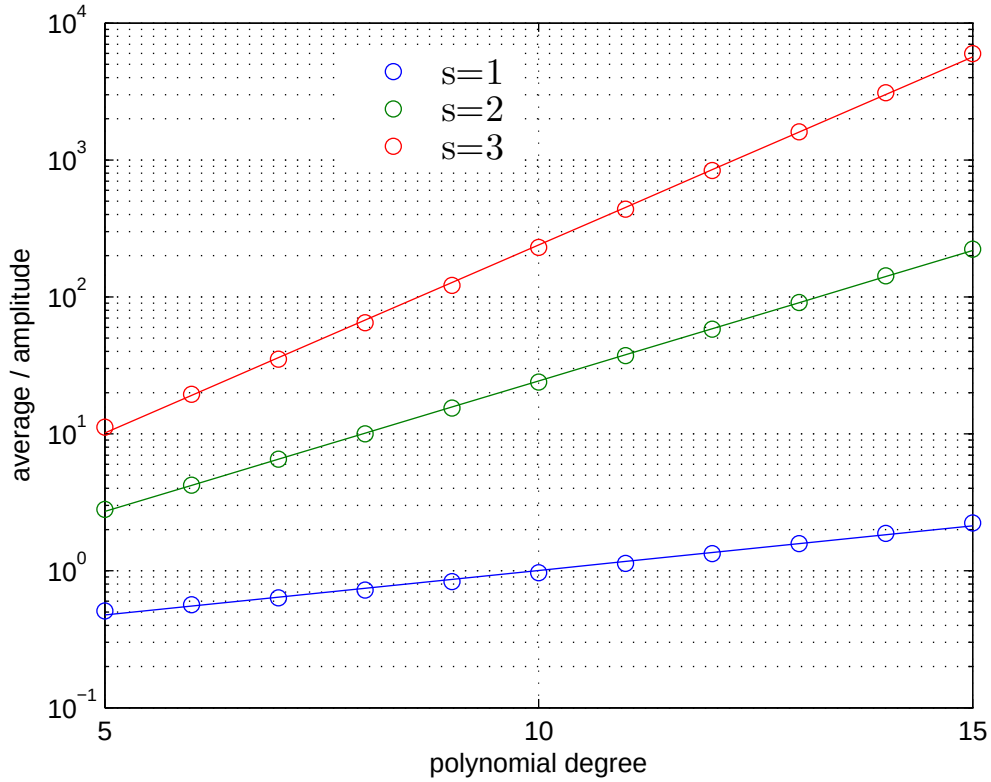


Figure 4.16: The minimum necessary normalized average to trip a specific s value for a step function

As previously stated a standard polynomial order of 9 was used. The values for a 9^{th} order polynomial fall in Tables 4.6. In order to prevent the sensor from turning on too much an average of the values for $s = 2$ for Noise and $s = 3$ for the step were used. This average could then by multiplied by the amplitude of the desired state in order .

s	Noise	Step
1	11.844	0.83421
2	217.75	15.425
3	1717.9	121.47

Table 4.6: s values for 9^{th} order polynomials

The shock tube was then rerun with these new scales to determine their accuracy. Figure 4.17 and Table 4.7 show that the new RMS error as well as a plot of the new density are nearly identical to the values previously decided upon. These show that the new scale function will provide a good result while removing any guess work previously being utilized. This is a huge improvement in that the sensor is more applicable to any type of state at a relatively small cost of being now based on the order of the polynomial while still retaining it's independence of the problem which many other sensors struggle with.

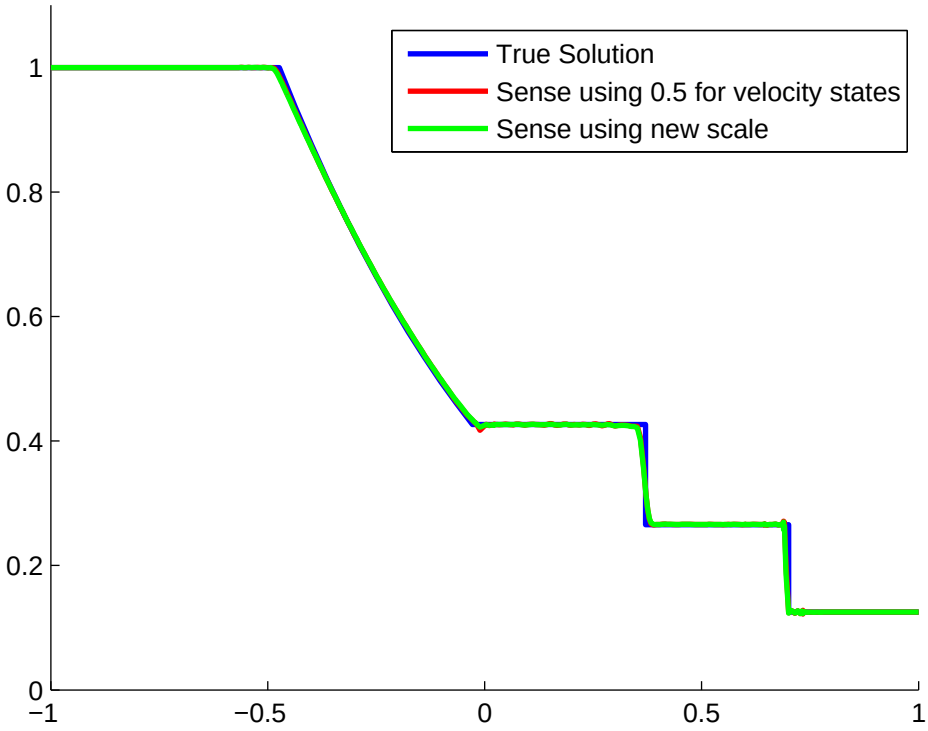


Figure 4.17: Using the new scale derivation to replace y_{norm}

Values Sensed On	RMS Error
Utilizing built in scale	0.0093
Velocity norm 0.5	0.0088

Table 4.7: RMS Error for Sod Shock Tube 3D

Chapter 5

Three Dimensional Vortex Problems

5.1 Generation of the Initial Condition

The burst vortex is a complicated problem when generating the initial condition. It requires pseudo steady state simulations as well as extrusions for a two dimensional vortex into a three dimensional field.

5.1.1 Velocity Profile

In order to establish the real world application of the bursting an accurate representation of the velocity profile needs to be made. For this problem a Gaussian vortex velocity profile was chosen that is developed as

$$v_\theta = \frac{\Gamma_0}{2\pi r}\left(1 - e^{-\beta(r/r_c)^2}\right) \tag{5.1}$$

where Figure 5.1 shows the non-dimensional velocity profiles for the two vortices with cores sizes of $r_{c_1} = 2.6$ and $r_{c_2} = 5.2$.

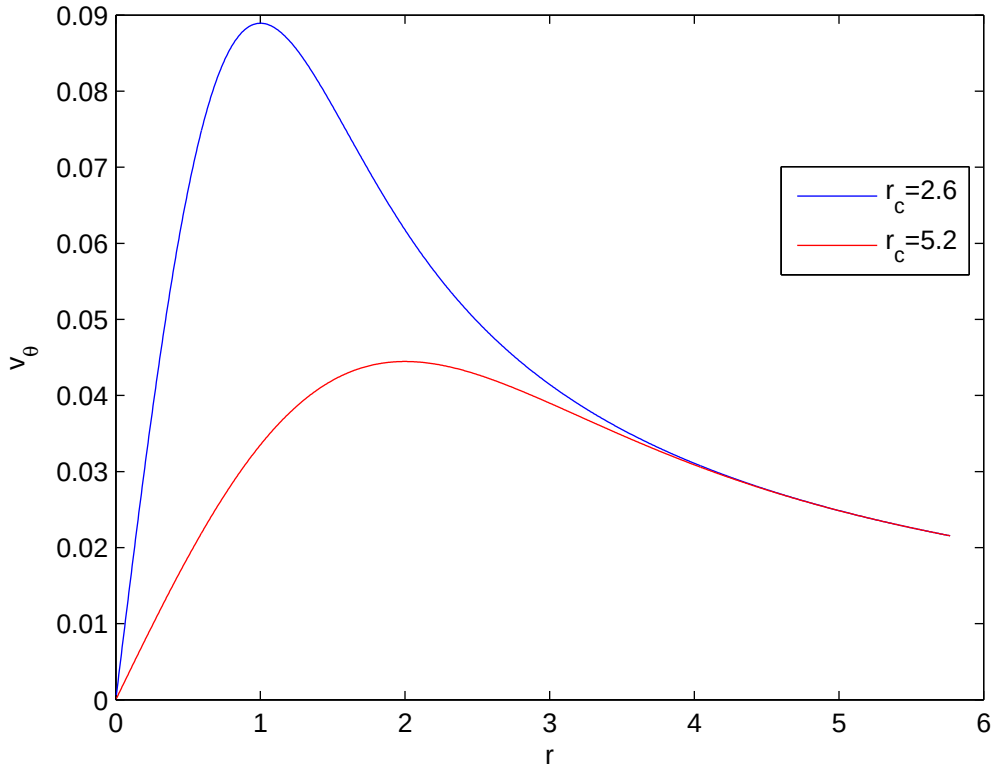


Figure 5.1: v_θ for the vortices of two different core sizes

With the velocity profile built in radial coordinates it then had to be shifted to Cartesian coordinates. This was done by first transforming the velocity equation into a velocity potential equation by

$$f = \frac{v_\theta}{r} \tag{5.2}$$

There is a problem with the profile though. At the center $r = 0$ the current velocity profile would reach a singularity, therefore a high order algebraic model was used only at $r = 0$. The velocity potential for all radii can then be found from

$$c = \frac{\Gamma_0}{2\pi a} \tag{5.3}$$

$$f = \begin{cases} \frac{c}{r_{c_1}} \left(\frac{(r/r_c)^2 + 2\gamma}{\left((r/r_c)^2 + \gamma\right)^2} \right) \\ \frac{c}{r_{c_1}\left(r/r_{c_1}\right)^2} \left(1.0 - e^{-\beta(r/r_c)^2}\right) \end{cases} \tag{5.4}$$

where Γ_0 represents the circulation.

5.1.2 Boundary Condition

With a velocity profile the boundary conditions then had to be developed. This way the pseduo-steady state simulations could be run for the two dimensional slices. This was done with symmetric boundary conditions in the transverse directions[22]. These are applied by imagining a ghost set of domains surrounding the vortex and each "ghost" domain bears a reflection of the vortex. This reflection is captured in rings around the current domain. Each ring contains more reflections of the domain due to a larger radius away from the original domain but, the further out the ring is the less the velocity of this "ghost" domain influences the development of the velocity profile. The symmetric boundary condition allows no velocity to cross its boundaries therefore the domain of the problem is of the utmost importance.

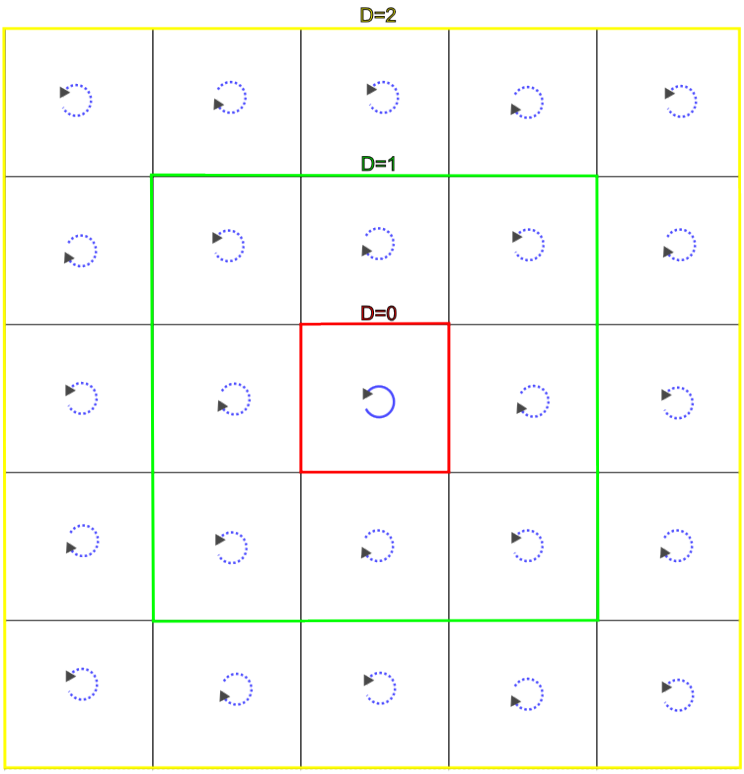


Figure 5.2: Example of Symmetric Boundary Condition on a Vortex

For this problem 25 rings were generated and then a Van Wijngaarden transformation was used in order to accelerate the series and find a more exact solution for the velocity than simply doing a summation. This is performed by first performing an Euler's transform which is accomplished by doing a summation of the array in question

$$s_{0,k} = \sum_{n=0}^{k} (-1)^n a_n \tag{5.5}$$

here s is a matrix of data for computing the Euler's transformation. The velocity summations are then stored in the top row of data of s at which point the Euler's transformation is performed where

$$s_{j+1,k} = \frac{s_{j,k} + s_{j,k+1}}{2} \tag{5.6}$$

Van Wijngaarden's contribution lay in noticing that it was pointless to takes this all the way to the end of the data. Rather by stopping two-thirds of the way through the transformation a more accurate value for the true summation was found. This contribution allowed for less rings to be used while obtaining much more accurate results for the velocity.

5.1.3 Two Dimensional Initial Condition

While the velocity field is well represented and the boundary conditions have been derived, there is no simple way to derive the pressure and density from the velocity. These values are intrinsically linked in the Euler equations. This led to a need to develop a pseudo-steady two dimensional initialization algorithm. This is accomplished by creating a standard flow field with uniform density and pressure with no velocity and then slowly introducing the true velocity field over a period of time and then running the simulation till the residual reaches three orders of magnitude below it's peak to show a steady state solution.

The introduction of the velocity is done through a factor based on a cosine function going from zero to one. After a certain set time that factor stays at one so that the initial condition can come to a constant value. This introduction equation is

$$s = \cos\left(\frac{t}{100} * \frac{\pi}{2} - \frac{\pi}{2}\right) \tag{5.7}$$

so at a time of one hundred seconds the value for s remains at one.

To achieve the introduction of the velocity slowly into the right hand side of the solution is as follows.

$$rhs\left(:,2\right) = rhs\left(:,2\right) + \frac{s}{\tau} * \left(U\left(:,1,i,j\right) * \left(u_t\left(:,1,i,j\right) - \frac{U\left(:,2,i,j\right)}{U\left(:,1,i,j\right)}\right)\right) \tag{5.8}$$

$$rhs\left(:,3\right) = rhs\left(:,3\right) + \frac{s}{\tau} * \left(U\left(:,1,i,j\right) * \left(v_t\left(:,1,i,j\right) - \frac{U\left(:,3,i,j\right)}{U\left(:,1,i,j\right)}\right)\right) \tag{5.9}$$

$$rhs\left(:,4\right)=rhs\left(:,4\right)+\frac{s}{\tau}*\left(U\left(:,2,i,j\right)*\left(u_t\left(:,1,i,j\right)-\frac{U\left(:,2,i,j\right)}{U\left(:,1,i,j\right)}\right)\right)$$
$$+\frac{s}{\tau}*\left(U\left(:,3,i,j\right)*\left(v_t\left(:,1,i,j\right)-\frac{U\left(:,3,i,j\right)}{U\left(:,1,i,j\right)}\right)\right) \tag{5.10}$$

Here i and j corresponds to the cells in the x and y direction, τ is a constant held to one half here so that there is more weight on the velocity inclusion, and u_t and v_t represent the true velocities in the x and y direction respectively This

5.1.4 Three Dimensional Extrusion

The two two dimensional vortices were then generated with the differing core radii previously mentioned. The ratio of the two core radii was altered in the exponential term of Equation 5.4. This way the strength of the vortex was unaffected by the changing core radius allowing for the circulation to be accurately maintained and the vortices to be merged. With the two dimensional initial conditions made all that was left was the generation of the three dimensional model.

The two dimensional slices were merged by using a sinusoid function going from $[1,-1]$ for $5.<z.<10.$ and $[-1,1]$ for $45.<z<50.$. This is represented by the function theta and can be seen below.

$$U=\begin{cases} U_{r_{c_2}} & z<5. \\ \theta\,U_{r_{c_2}}+(1-\theta)\,U_{r_{c_1}} & 5.<z<10. \\ U_{r_{c_1}} & 10.<z<45. \\ \theta\,U_{r_{c_2}}+(1-\theta)\,U_{r_{c_1}} & 45.<z<50. \\ U_{r_{c_2}} & z>50. \end{cases} \tag{5.11}$$

$$\theta=\begin{cases} \frac{1}{2}+\frac{1}{2}\sin\left(-\pi/5\,z+3\pi/2\right) & 5.<z<10. \\ \frac{1}{2}+\frac{1}{2}\sin\left(\pi/5\,z-19\pi/2\right) & 45.<z<50. \end{cases} \tag{5.12}$$

This generates the initial condition. The velocity in the transverse directions was then perturbed using a random perturbation applied as $u=u_{\text{base}}\left(1+\epsilon\tilde{u}\right)$ where $\epsilon=1x10^{-2}$ and $||\tilde{u}||\leq 1$. This trips the helical instability in the problem. The boundary condition for

the axial direction of the vortex bursting problem was also a symmetric boundary condition which due to the velocity being independent of z is more like a reflecting wall boundary condition.

5.2 Results

The CFL number plays a huge role in the development of the time step for all CFD applications. For the two dimensional cases this CFL number can approach one for most problems but this is not the case for three dimensional problems. In order to determine the appropriate CFL number, the number was varied and the three separate test cases were run. These test cases had varying CFL numbers of CFL = $[0.25, 0.3, 0.35]$ with a grid size of 20x20x20. The simulation corresponding to the CFL number of 0.35 failed, therefore to keep the code running as fast as possible the CFL number was established at 0.3.

The minimum pressure is plotted for varying grid sizes and polynomial order in Figures 5.3-5.8. The plots show that the burst follows the same path as the vortex bursting pressure field found in Figure 5.9 [16]. These figures all show a constant velocity in the pressure wave collapsing into each other. In order to compare the time the normalization was shifted in post processing by a conversion factor of $v_{\theta_{max}}/(2a\pi)$. The burst is identified by the sharp increase in pressure as seen in each of the Figures and is found at a constant location. This lines up with the results in the Moet et. al. paper where they found that a constant propagation speed existed for all the test cases and the burst occurs at an exact time based off the ratio of the core radii. There is a difference in the amplitude of the pressure waves but this is understandable due to the difference in the normalization techniques. This project was normalized by utilizing the dynamic pressure as the normalization factor where the Moet paper uses.

$$p^* = \frac{p - p_{min}}{p_\infty - p_{min}} \tag{5.13}$$

The grid size and polynomial order where varied in order to find the most accurate solution and see how each change impacts the solution. The Figures show that the amount of cells in the axial direction do not have a large influence on the pressure wave. Instead the transverse cells hold the most influence. This is logical because since this plot is based on the minimum pressure and plotted along the axial direction. A very interesting note is that even though the 13^{th} order polynomial used less cells and degrees of freedom it obtained an equally accurate version of the solution when compared to the paper by Moet et. al. The downside to the this implementation was in the time needed to run the solution. Even though it had the least degree's of freedom the solution took the longest to run.

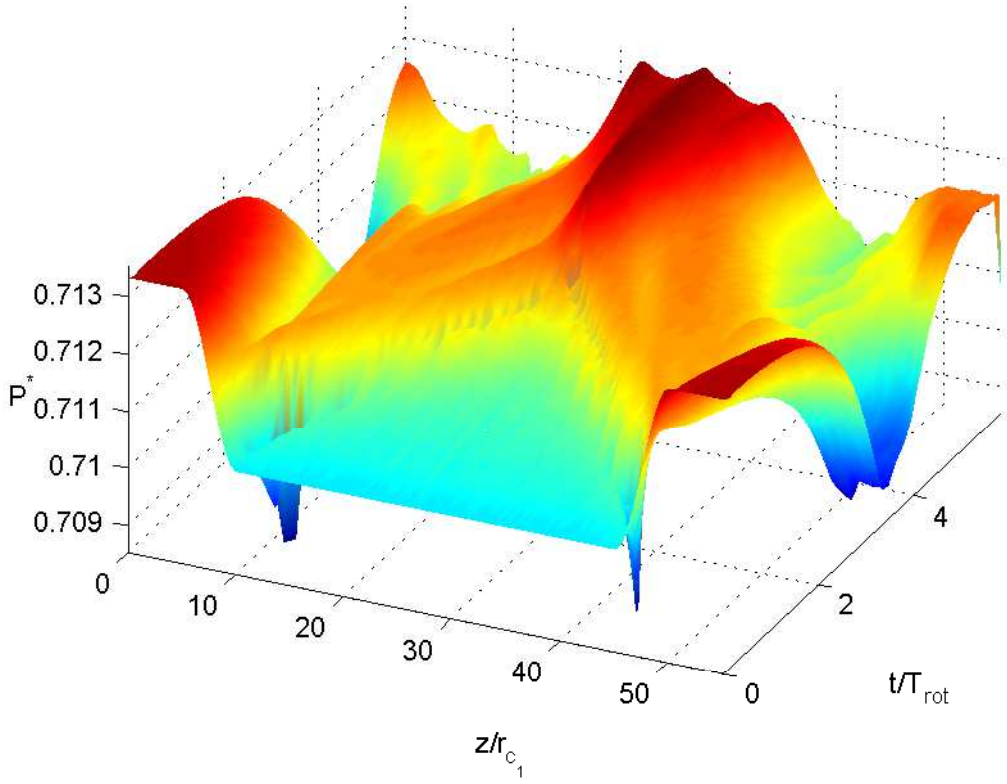


Figure 5.3: Plot of the minimum pressure evolving through time for a polynomial of the 9^{th} order with a grid N_i=N_j=20 and N_k=25

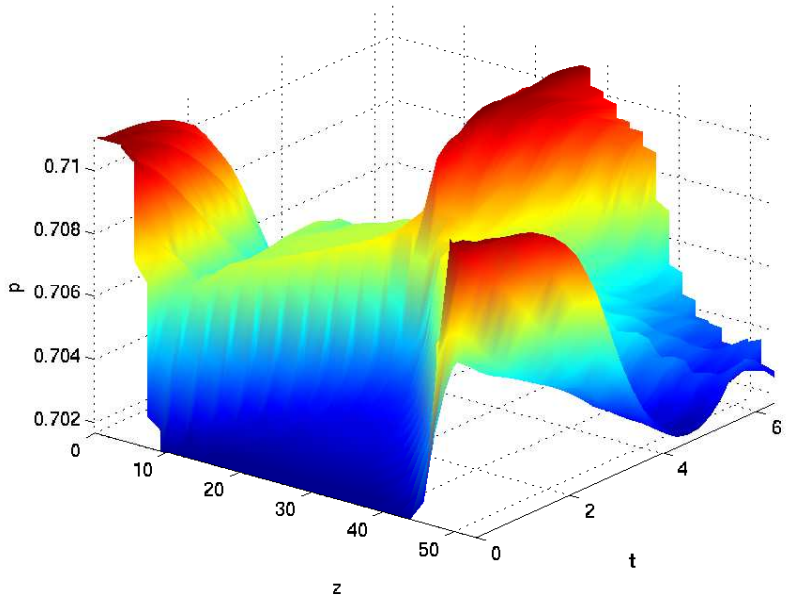


Figure 5.4: Plot of the minimum pressure evolving through time for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=30

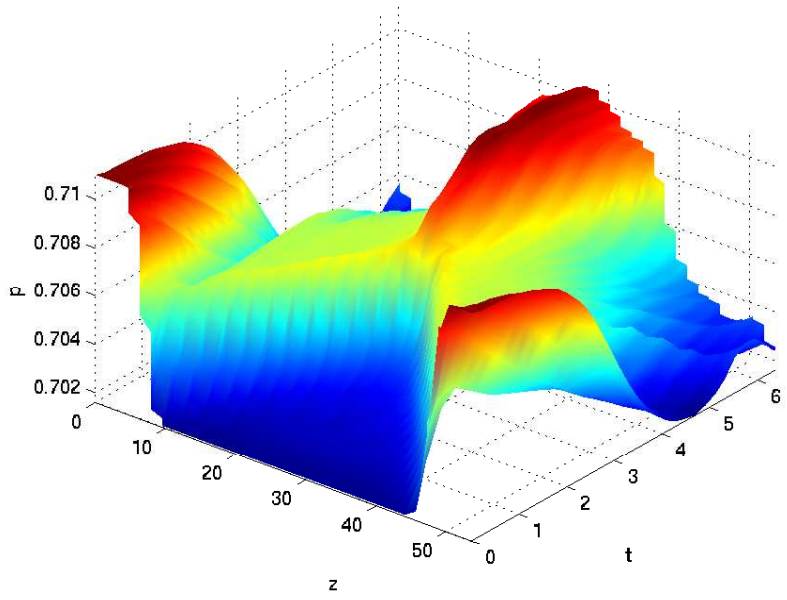


Figure 5.5: Plot of the minimum pressure evolving through time for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=35

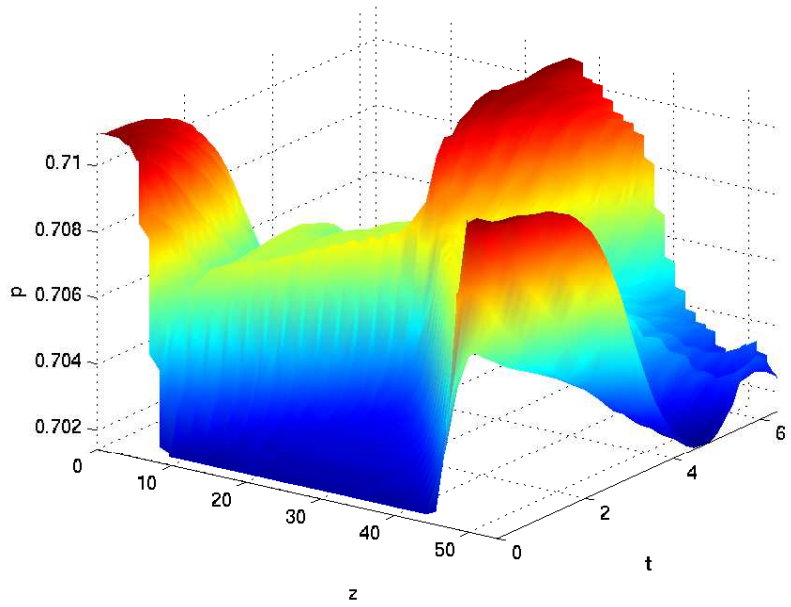


Figure 5.6: Plot of the minimum pressure evolving through time for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=40

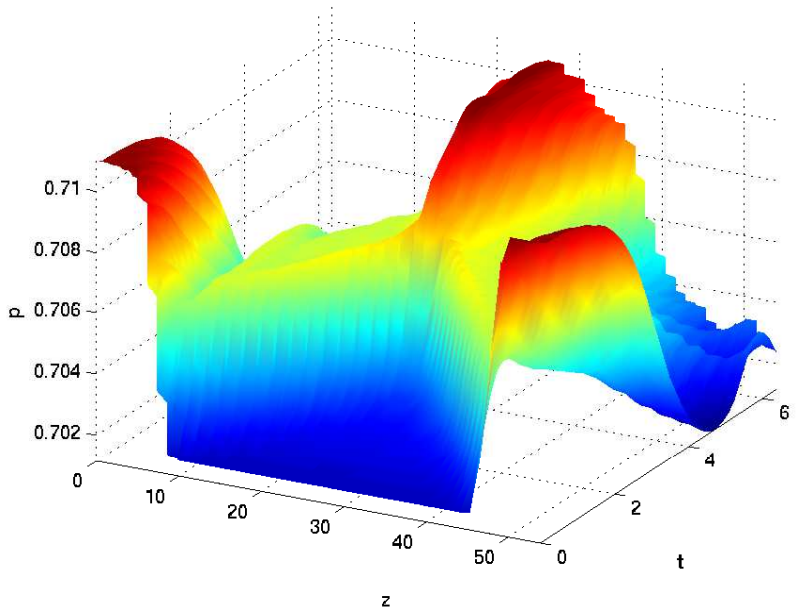


Figure 5.7: Plot of the minimum pressure evolving through time for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=45

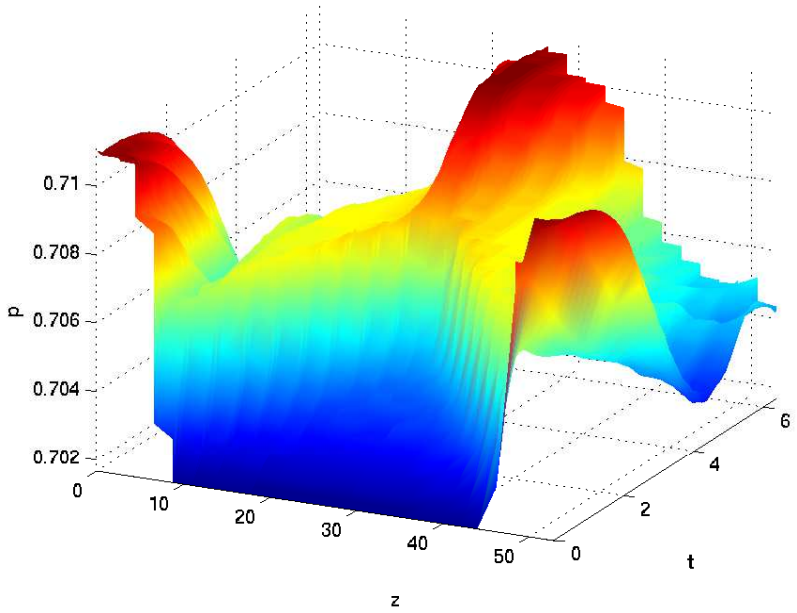


Figure 5.8: Plot of the minimum pressure evolving through time for a polynomial of the 13$^{\text{th}}$ order with a grid size of N_i=N_j=17 and N_k=25

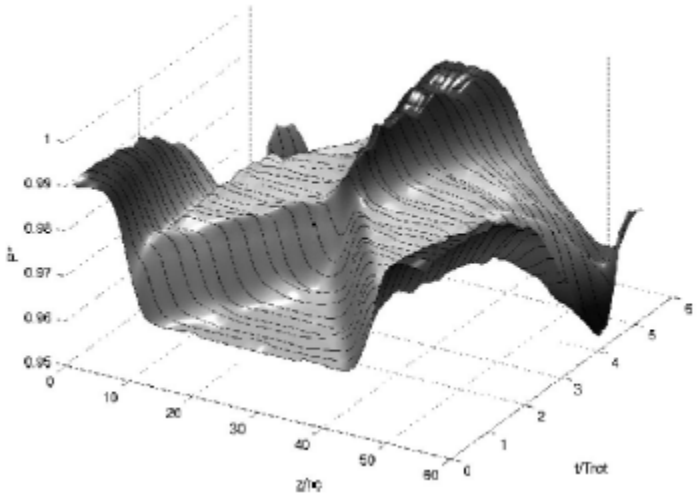


Figure 5.9: Temporal evolution of the profile of minimum pressure in the vortex core for simulation DNS$_2$[16]

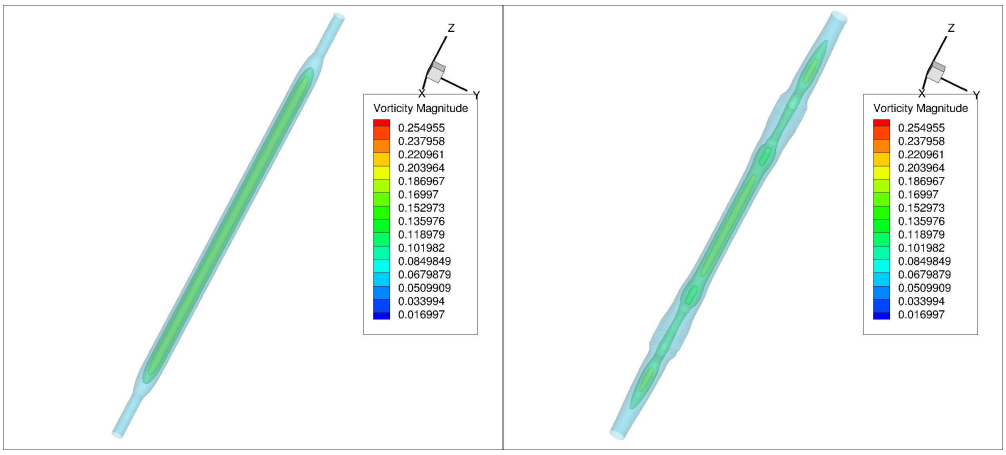
Since the pressure plot had poor quality when the transverse number of cells was below 25 in each direction the vorticity for those solutions was not plotted. The vorticity magnitude is then plotted for each of the remaining cases in Figures 5.10-5.14. They are listed with their

time being changed to the scale of the paper to allow for better comparisons. Due to the time step being derived by the simulation the times for each figure could not perfectly match but the physics still shows the same situations occurring in comparison to Figure 5.15. Image (c) for all figures shows the initial burst as the two pressure waves meet in the middle of the simulation. This show the core radius expanding at the point of impact. Image (d) for all the simulations then show the beginning of the helical instability. This shows that the sensor allowed for the natural creation of the instability without overpowering it. As the number of cells are increased in the z direction the instability becomes more clear as can be seen from Figures 5.12 and 5.13 when compared to Figures 5.10, 5.11, and 5.14. The increased polynomial order did not show the helical instability as quick as the other examples due to a lower resolution in the z direction. While image(e) in the figures all show the burst cloud becoming its own entity in the center of the simulations. The helical instability can easily be seen in the (e) image for all the figures.

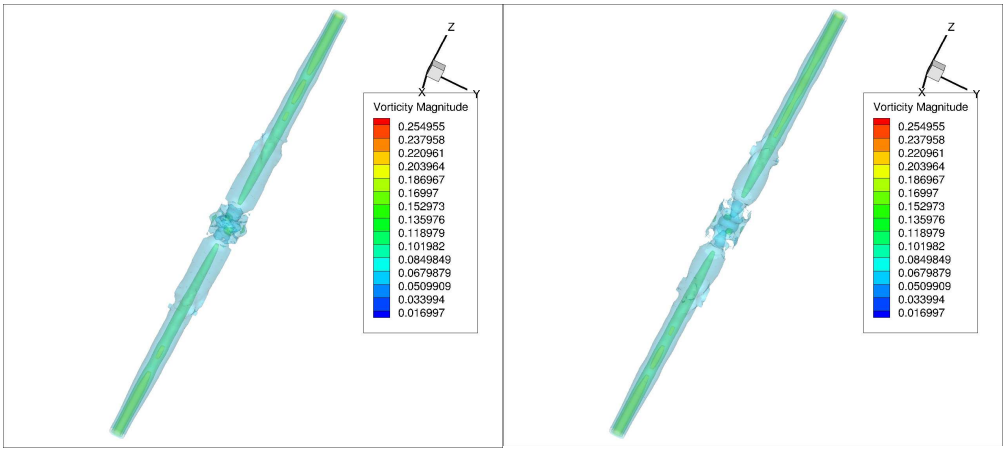
This has all been based on a visual inspection therefore a more quantifiable method was used to back up the analysis from the figures. Since this is a problem based on the Euler Equations and the boundary conditions have no flow passing through them the total kinetic energy should be a constant throughout time. Since artificial viscosity is being applied this will not hold true completely and therefore if too much viscosity is added the quality of the solution decreases and it can be tracked. Figures 5.16 and 5.17 show the total kinetic energy of the problem either normalized by the kinetic energy of the initial condition or the error when compared to the initial condition. These figures both show that there is only small incremental improvement with the increase of cells in the axial direction which with the increase in the time required for simulation makes a solution with around forty cells in the axial direction seeming to be most optimal. The best error was shown to occur on the increase in both the axial and radial cells which lines up with the other charts.

These vorticity magnitudes and kinetic energy plots show that this robust adaptation of the sensor was able to approximate the same viscous physics effects as both a sixth order

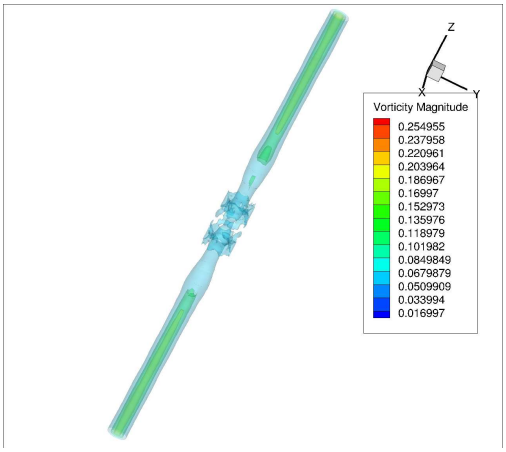
LES and DNS simulations without requiring the solving of the RANS equations and therefore without the need to utilize a turbulence model. These figures also show that increasing the axial resolution has only a slight effect in the decrease of the error. This shows the superior results to be for the 9^{th} order polynomial with a grid of $N_i = N_j = 28, N_k = 40$ because it is not only the most accurate.



(a) t=0. (b) t=1.9064

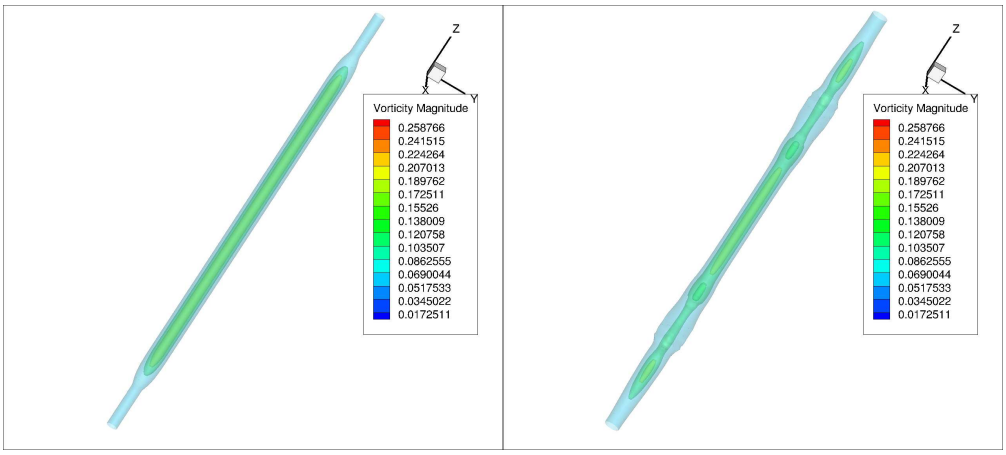


(c) t=4.0159 (d) t=4.5286



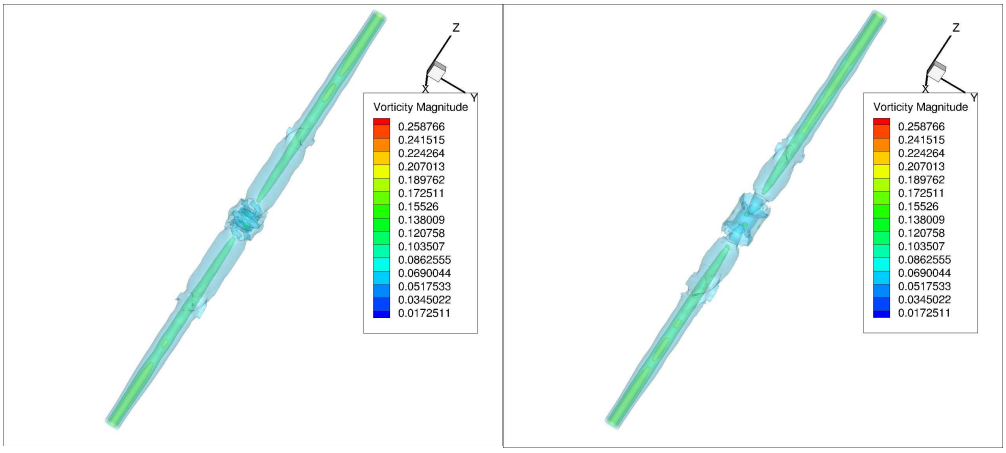
(e) t=6.3687

Figure 5.10: Isosurfaces of vorticity magnitude for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=30



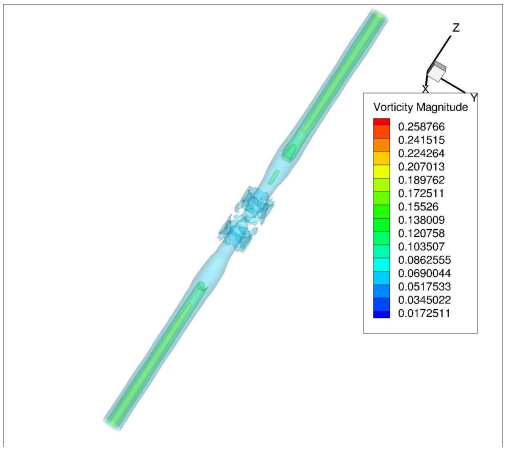
(a) t=0.

(b) t=1.8384



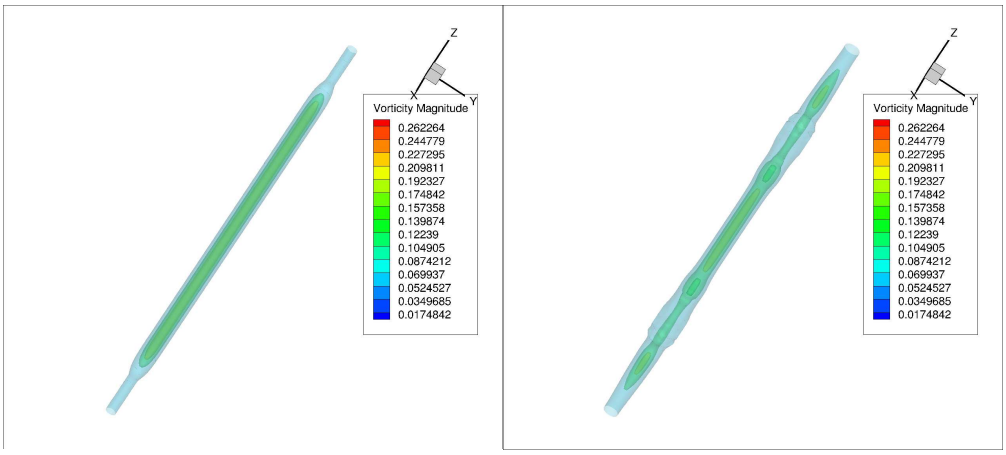
(c) t=3.8821

(d) t=4.4949

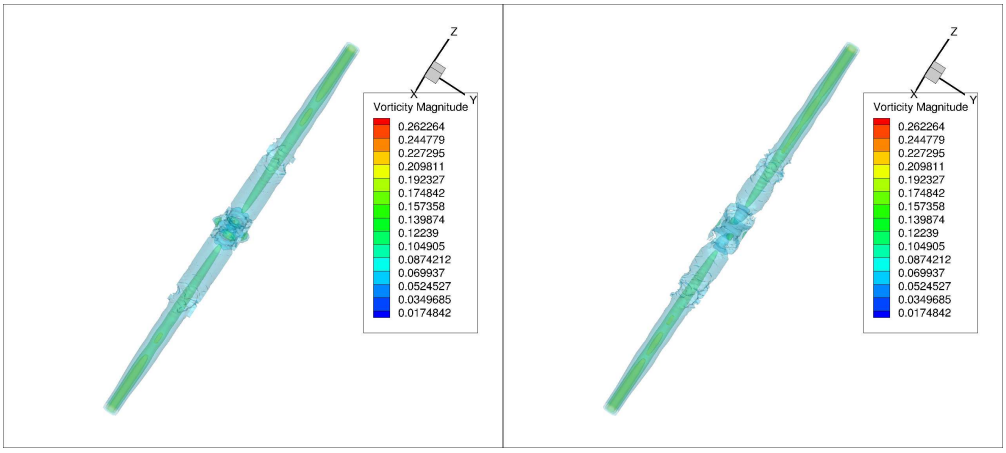


(e) t=6.3687

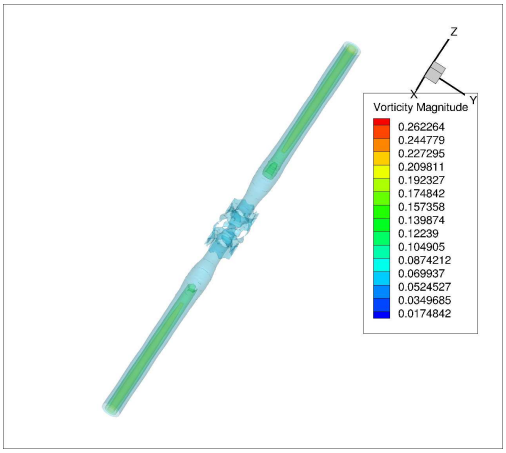
Figure 5.11: Isosurfaces of vorticity magnitude for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=40



(a) t=0. (b) t=1.9658

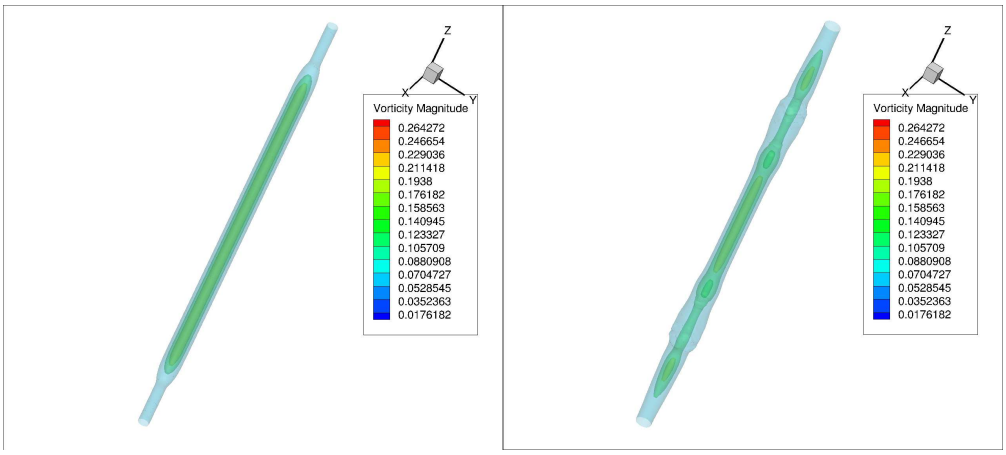


(c) t=3.933 (d) t=4.4694

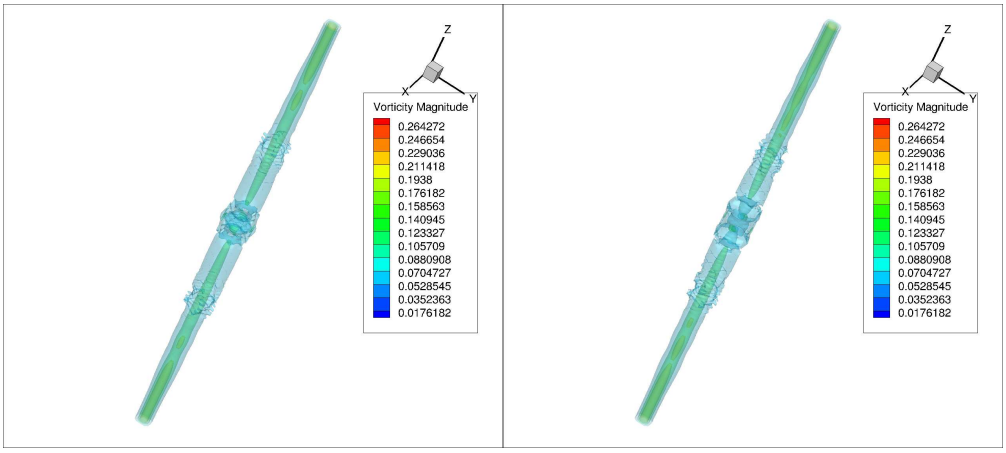


(e) t=6.3687

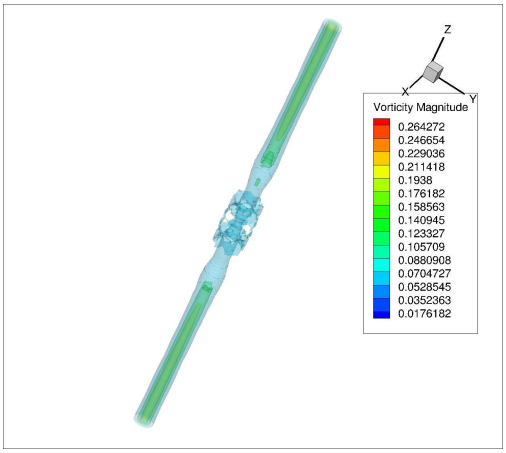
Figure 5.12: Isosurfaces of vorticity magnitude for a polynomial of the 9$^{\text{th}}$ order with a grid N_i=N_j=25 and N_k=40



(a) t=0. (b) t=1.9064

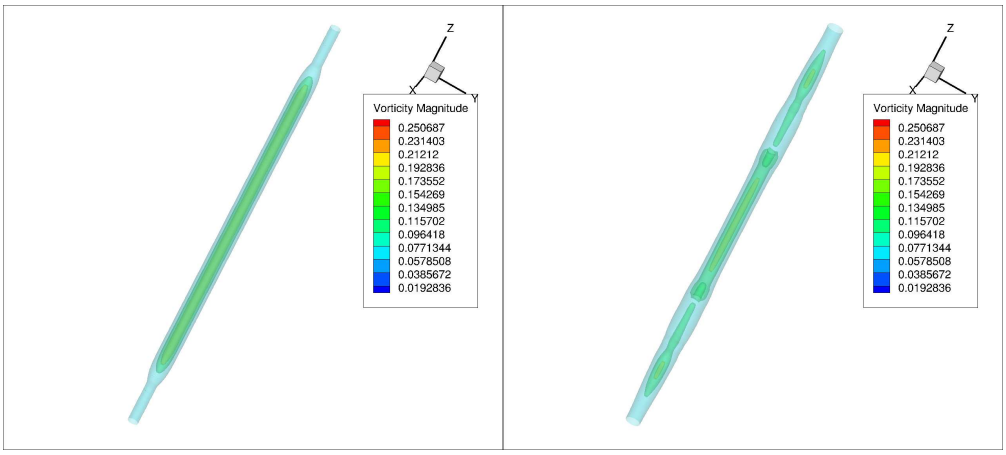


(c) t=3.9727 (d) t=4.6081

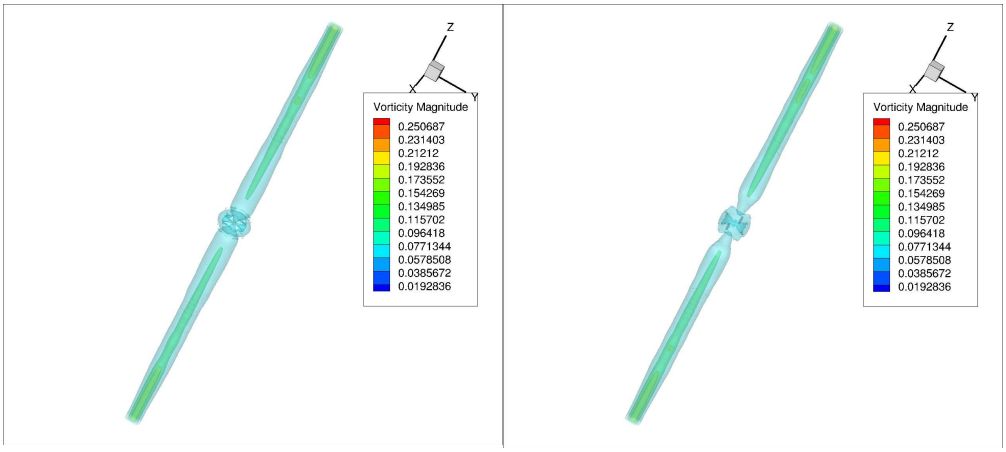


(e) t=6.3687

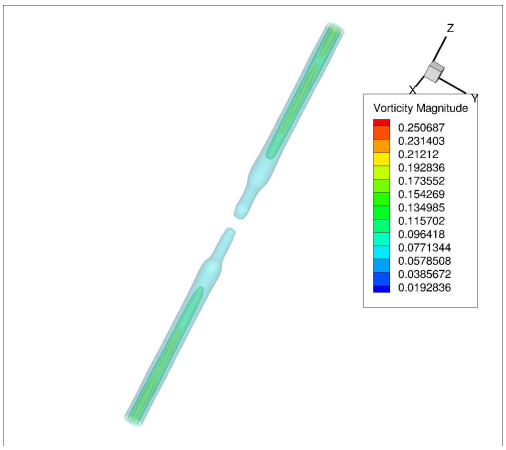
Figure 5.13: Isosurfaces of vorticity magnitude for a polynomial of the 9^{th} order with a grid N_i=N_j=25 and N_k=45



(a) t=0. (b) t=1.9205



(c) t=3.9798 (d) t=4.5289



(e) t=6.3687

Figure 5.14: Isosurfaces of vorticity magnitude for a polynomial of the 13^{th} order with a grid N_i=N_j=17 and N_k=25

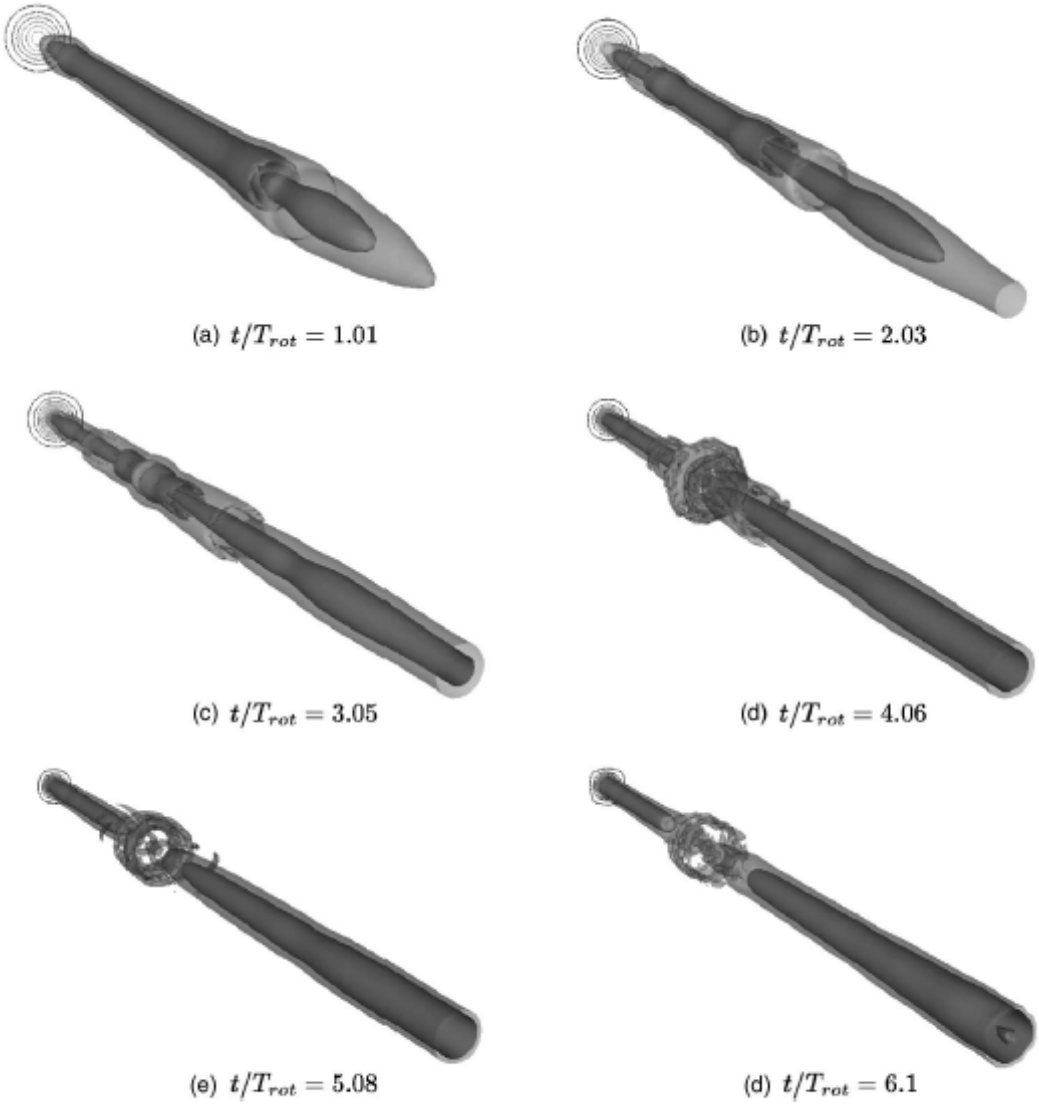


Figure 5.15: Isosurfaces of vorticity magnitude of run DNS_2 ($T_{rot} = 2\pi r_c/V_{\theta_{max}}$).[16]

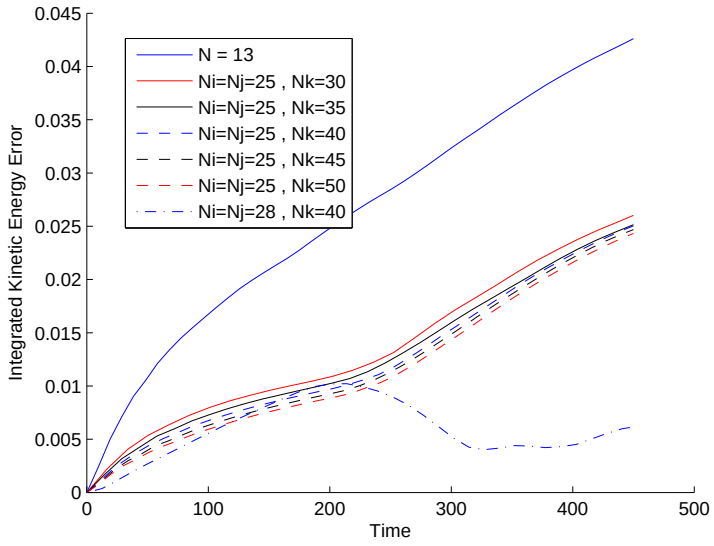


Figure 5.16: Plot of the integrated total kinetic energy error versus time for the cases

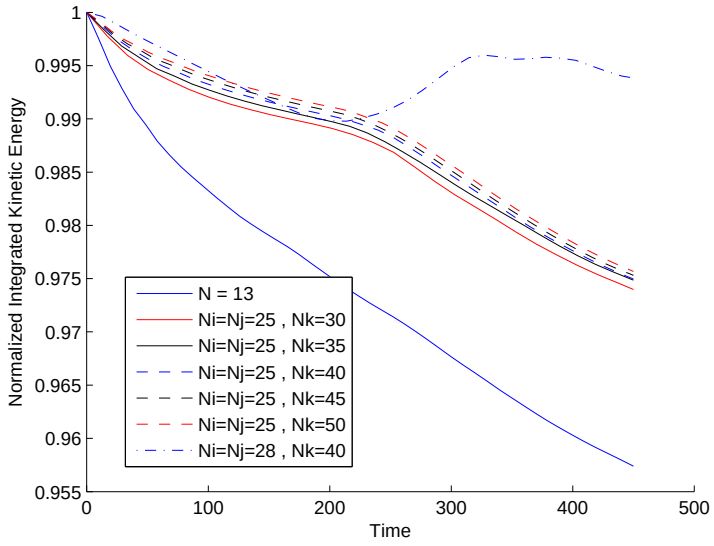


Figure 5.17: Plot of the integrated total kinetic energy normalized for each case versus time

Chapter 6

Conclusions

In conclusion these simulations show the high applicability of the DG method utilizing the Euler equations to a range of problems. The method was further improved by taking the Klöckner sensor and augmenting it to improve it's capabilities and make it more robust. In devising a new system with which to apply the baseline modal decay to the sensor allowing for the sensor to work on values with a norm value of zero as well as speed which was one of the main goals of this thesis and its most important findings. The savings in cost also reduce the flop count in a two dimensional simulation by $N_{fp}^3/2$ per cell per timestep and for three dimensional to reduce it by $N_{fp}^4/3$ flops.

This sensor was then taken and applied to the three dimensional vortex bursting problem and provided accurate results in showing the vortex burst occurring as well as the helical instability developing. This was done without the need to work with either a LES or DNS simulation saving immensely in the complexity of the code as well as the required number of arbitrary coefficients that must be utilized in a turbulence model. The number of cells in the axial direction was shown to be the most important factor in seeing the development of the helical instability while the resolution in the transverse directions played more of a role in the pressure distribution charts. The 9^{th} order polynomial was superior to the 13^{th} order due to the increase in speed it allowed as well as the better overall accuracy.

6.1 Future Work

While this work has shown the use of the DG method for capturing one piece of a wake future implementation would require some additional research and steps. One improvement would be to move this code from a Cartesian mesh to either a triangular based mesh or a

non-uniform Cartesian mesh. Pure Cartesian meshes hold a strong advantage when working with the far wake or other basic test cases but it has difficulty adapting around wings or other physical phenomena. They also can place an unnecessary amount of points in locations where the physics is not occurring. In this problem that would be in the corners. To make this shift it would require a change to the Jacobian calculation and in converting it from a scalar calculation to a matrix that needs to be calculated at each cell. Another possibility would be to change the grid to a nested Cartesian mesh instead of a non-uniform Cartesian mesh. In a nested mesh some cells share a wall with another cell. For these problems a number of points would be interpolated to fit the problem and then interpolated back out to extract the flux for all faces. This can damage the physics but since the point of interest is the center where the cells would be of uniform size it would not cause too many problems with the results.

Another improvement would be on implementing the Navier-Stokes Equations with a turbulence model so that the boundary layer can be modeled and thus the simulations an the model the formation of the wakes as well as wake phenomena which are basically inviscid . This change would also require another way to improve these simulations by adjusting the parallelization techniques to utilize either MPI or CUDA which could substantially increase the speed of the code. The use of openmp while allowing for some rudimentary parallelization it also hinders the code in restricting the number of threads available to two CPUs. This has been implemented by Klöckner actually in an open source system he developed called PyCUDA.

Finally this sensor is not based on any type of physics so it would be interesting to take this sensor and apply it to a problem previously solved only using global viscosity such as the solid rocket grain as seen in Albarado's thesis. This global viscosity also wasn't physical but was used so that the scheme was stable. This problem could then be translated to three dimensions and then apply the fully modified sensor to it as well.

Bibliography

[1] Albarado, K. "*Application of the Level Set Method to Solid Rocket Motor Simulation*, " Masters thesis, Auburn University, 2012.

[2] Bassi, F., and Rebay, S., "High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations," *Journal of Computational Physics*, Vol. 138, 1997, pp.251-285.

[3] Bassi, F., and Rebay, S., "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 131, 1998, pp.267-279.

[4] Cockburn, B., and Shu, C.-W., "The Local Discontinuous Galerkin Method for Time-Dependant Convection-Diffusion Systems," *Siam Journal of Numerical Analysis*, Vol. 35, No. 6, pp. 2440-2463.

[5] Cockburn, B., and Shu, C.-W., "Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems," *Journal of Scientific Computing*," Vol. 16, No. 3, 2001, pp.173-261.

[6] Crow, S. C., "Stability Theory for a Pair of Trailing Vortices," *AIAA Journal*, Vol. 8, No. 12, 1970, pp. 2172-2178.

[7] Favors, J. "*A Comparison of Artificial Viscosity Sensors for the Discontinuous Galerkin Method*, " Masters thesis, Auburn University, 2013.

[8] Gerz, T., Holzäpfel, F., and Darracq, D., "Commercial aircraft wake vortices," *Progress in Aerospace Sciences*, Vol. 38, 2002, pp. 181-208.

[9] Gottlieb, S., and Shu, C.-W., "Total Variation Diminishing Runge-Kutta Schemes," *Mathematics of Computation*, Vol. 67, No. 221, 1998, pp. 73-85.

[10] Hesthaven, J. S., and Warburton, T., "*Nodal Discontinuous Galerkin Methods - Algorithms, Analysis, and Applications*, " Springer, first ed., 2008.

[11] Holzäpfel, F., Hofbauer, T., Darracq, D., Moet, H. Garnier, F., and Gago, C. F., "Analysis of wake vortex decay mechanisms in the atmosphere," *Aerospace Science and Technology*, Vol. 7, 2003, pp. 263-275.

[12] Klöckner, A., Warburton, T., and Hesthaven, J. S., "Viscous Shock Capturing in a Time-Exlpicit Discontinuous Galerkin Method," *Mathematical Modelling of Natural Phenomena*, 2010, pp. 1-42.

[13] Lewellen, D. C., and Lewellen, W. S., "Large-Edyy Simulations of the Vortex-Pair Breakup in Aircraft Wakes," *AIAA Journal*, Vol. 34, No.11, 1996, pp. 2337-2345.

[14] Liou, M.-S., "A sequel to AUSM: AUSM$^+$," *Journal of Computational Physics*, Vol. 129, No. 2, 1996, pp.364-382.

[15] Liou, M.-S., "A sequel to AUSM, Part II: AUSM$^+$-up for all speeds," *Journal of Computational Physics*, Vol. 214, No. 1, 2006, pp.137-170.

[16] Moet, H., Laporte, F., Chevalier, G., and Poinsot, T., "Wave propagation in vortices and vortex bursting," *Physics of Fluids*, Vol. 17, 2005, pp. 054109(1-16).

[17] Persson, P.-O., and Peraire, J., "Sub-Cell Shock Capturing for Discontinuous Galerkin Methods," *In Proc. of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, Vol. 112,2006.

[18] Reed, W. H., and Hill, T. R., "Triangluar mesh methods for the neutron transport equation," *Los Alamos Scientific Labratory Report*, LA-UR-73-479, 1973

[19] Reitz, R. J. "*Resolution and Boundary Placement Effects on Discontinuous Galerkin Simulations of Counter-rotating Vortex Interaction in a Confined Domain*, " Masters thesis, Auburn University, 2013.

[20] "Shock Tube," `http://www.grc.nasa.gov/WWW/wind/valid/stube/stube.html` [retrieved 28 February 2012]

[21] Spalart, P. R., "Airplane Trailing Vortices", *Annual Review of Fluid Mechanics*, Vol. 30, 1998, pp. 107-138.

[22] Sreedhar, M., and Ragab, S., "Large eddy simulation of longitudinal stationary vortices," *Physicis of Fluids*, Vol. 6, 1994, pp. 2501-2514.

[23] Tannehill, J. C., Anderson, D. A., and Pletcher, R. H., "*Computational Fluid Mechanics and Heat Transfer*, " Taylor and Francis, second ed., 2004.

[24] Yan, J., and Osher, S., "A local discontinuous Galerkin method for directly solving Hamilton-Jacobi equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 232-244.