A MULTI-RESOLUTION DISCONTINUOUS GALERKIN METHOD FOR RAPID SIMULATION OF THERMAL SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Daniel Gempesaw

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
December 2011

A MULTI-RESOLUTION DISCONTINUOUS GALERKIN METHOD FOR RAPID SIMULATION OF THERMAL SYSTEMS

Approved by:

Professor Yogendra Joshi, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Professor Satish Kumar
School of Mechanical Engineering
Georgia Institute of Technology

Professor Hao-Min Zhou
School of Mathematics
Georgia Institute of Technology

Date Approved: 26 August 2011

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor, Dr. Yogendra Joshi, for his tireless assistance during the long journey that has been my Master's degree. I would also like to thank my parents for their encouragement and support, as well as my girlfriend Karen Shih. Lastly, the various members of the CEETHERM/METTL group at the Georgia Institute of Technology were always very helpful and considerate and I'd like to thank them for their kindness.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . **iii**

LIST OF TABLES . . . . **vi**

LIST OF FIGURES . . . . **vii**

LIST OF SYMBOLS OR ABBREVIATIONS . . . . **viii**

SUMMARY . . . . **ix**

I INTRODUCTION **1**

1.1 Motivation: Thermal Management of Data Centers 1

1.1.1 Motivation 2

1.2 Objective 4

1.3 Reproducibility of Results 5

II PREVIOUS WORK **8**

2.1 Discontinuous Galerkin Methods 8

2.1.1 Hybrid and Easy Discontinuous Galerkin Environment 9

2.2 Wavelets and Multiwavelets 9

2.2.1 Introduction To Wavelets 10

2.2.2 Wavelets and CFD 12

2.2.3 Wavelet Turbulence Modeling 14

2.2.4 Wavelet-Based Numerical Methods 16

III MULTI-RESOLUTION DISCONTINUOUS GALERKIN METHOD **19**

3.1 Multi Resolution Approximation 19

3.1.1 Describing the Approximation Space 19

3.1.2 Series Approximation 21

3.1.3 Wavelet Thresholding 22

3.2 The Discontinuous Galerkin Method 24

3.3 Time Stepping Method 26

3.4 Choice of Flux 27

3.5 Incompressible Navier-Stokes Equations with Heat Transfer 28

IV SIMULATION RESULTS **31**
4.1 Mesh Refinement Comparison 31
4.1.1 Runtime and Accuracy Comparisons 33
4.2 Simple Steady State Flow 35
4.3 Steady State Channel Flow 35
4.3.1 Cross Section Comparisons 39
4.4 Transient Channel Flow 43
V CONCLUSION **45**
5.1 Future Work 45
5.1.1 Extension to 3-D 46
5.1.2 Method Applicability 46
REFERENCES **48**

LIST OF TABLES

1 General Hardware and Software Versions 7
2 Software Versions for Hedge and Direct Dependencies 7

LIST OF FIGURES

1.1 Hot Aisle Cold Aisle 2

1.2 Defense Thermal Management Scales 3

1.3 Domain Decomposition and Discontinuities 5

1.4 Single Scale vs Multi Scale 6

2.1 Multi Resolution Approximation 11

2.2 Comparison of Classic Methods and their Wavelet Analogues 13

3.1 Midpoint subdivisions 20

3.2 Thresholding a Noisy Signal 23

3.3 DG Subtask Decomposition 26

3.4 Model Problem 28

4.1 A comparison between the basic mesh used in the simulations and the refined mesh used in this section for verification. 31

4.2 Comparison of velocity vectors between basic mesh and a refined mesh for a transient simulation. 32

4.3 Run Time Comparison 33

4.4 MRDG Laminar Flow 36

4.5 Fluent Laminar Flow 36

4.6 Steady state channel flow with bluffs after 50s of flow development; comparison between DG and MRDG shows an advantage for the MRDG method. The results generated using ANSYS ® FLUENT are shown in the bottom row. 38

4.7 Comparisons of the velocity profiles and temperature contours taken at different cross-sections throughout the channel. The simulations shown are produced by the MRDG simulation as well as FLUENT for simulation verification. 40

4.8 Comparisons of the temperature contours taken at different cross-sections throughout the channel. 41

4.9 Transient channel flow with a different starting condition. 43

LIST OF SYMBOLS OR ABBREVIATIONS

CUDA	Compute Unified Device Architecture, nVidia's GPU-enhanced parallel computing efforts.
DG	Discontinuous Galerkin.
FD	Finite Difference.
FEM	Finite Element Method.
FV	Finite Volume.
GPU	Graphics processing unit.
MPI	Message Passing Interface.
MRDG	Multi Resolution Discontinuous Galerkin.
N	Polynomial degree.
PDE	Partial Differential Equation.

SUMMARY

Efficient, accurate numerical simulation of coupled heat transfer and fluid dynamics systems continues to be a challenge. Direct numerical simulation (DNS) packages like FLUENT exist and are sufficient for design and predicting flow in a static system, but in larger systems where input parameters can change rapidly, the cost of DNS increases prohibitively. Major obstacles include handling the scales of the system accurately - some applications span multiple orders of magnitude in both the spatial and temporal dimensions, making an accurate simulation very costly. There is a need for a simulation method that returns accurate results of multi-scale systems in real time. To address these challenges, the Multi-Resolution Discontinuous Galerkin (MRDG) method has been shown to have advantages over other reduced order methods. Using multi-wavelets as the local approximation space provides an inherently efficient method of data compression, while the unique features of the Discontinuous Galerkin method make it well suited to composition with wavelet theory. This research further exhibits the viability of the MRDG as a new approach to efficient, accurate thermal system simulations. The development and execution of the algorithm will be detailed, and several examples of the utility of the MRDG will be included. Comparison between the MRDG and the "vanilla" DG method will also be featured as justification of the advantages of the MRDG method.

CHAPTER I

INTRODUCTION

1.1 Motivation: Thermal Management of Data Centers

A data center is a facility that houses computers and their necessary support infrastructure. This includes the storage systems for the computers as well as the environmental control systems that maintain the temperature, humidity, and air quality in the data center. Until a few years ago, energy usage in data centers was not a primary concern for many businesses, and data centers were expanded as needed. However, the newer servers generate more heat than their older counterparts and have been decreasing in size, meaning more servers with higher heat generation rates can fit in the same data center[33]. These trends have led to data centers consuming more power and becoming an area where increasing efficiency would save a significant amount of money. In today's data centers, approximately half of the power goes towards powering the actual IT equipment; the rest of the power is used for cooling, backup power, and lost in power conversions [30]. In fact, the server hardware itself is no longer the most expensive part of a data center: the price of a new server is less than the cost of power and cooling for the lifetime of that server [8]. These trends have motivated a focus on data center energy efficiency, and in particular methods for advanced cooling strategies.

Currently, the industry standard for server organization the data center has been servers stacked vertically to form racks, which are then placed side by side to form a row. A row faces another row and cold air is fed to the front of these racks, passed through the servers, and expelled on the other side of the row. This configuration is called Hot Aisle Cold Aisle (HACA) because of the alternating nature of the aisles created as seen in Figure 1.1. Computer room air conditioning units (CRAC) supply the cold air for the servers and recycle the warm air after it passes through the server. A common method for providing the cold air to the servers is by pushing through an under-floor plenum to the cold aisle

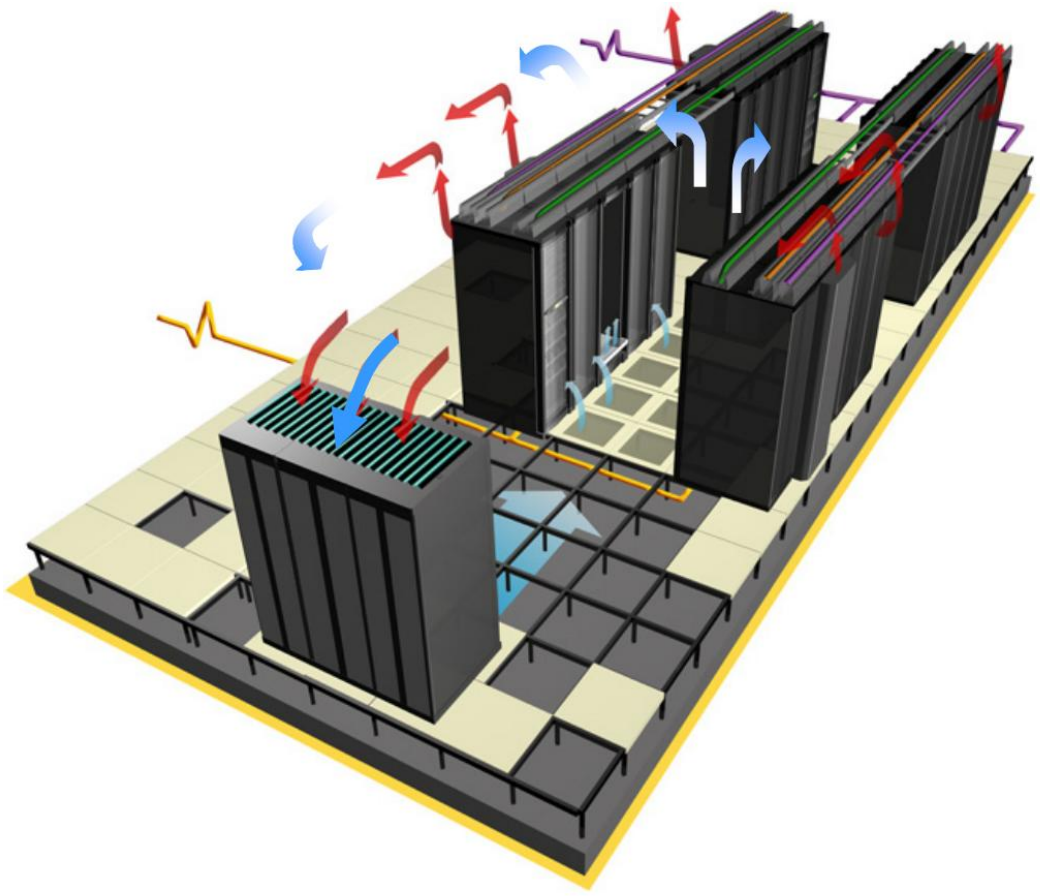


Figure 1.1: A image depicting the Hot Aisle Cold Aisle Layout [54]

where it rises through perforated tiles to reach the racks. After passing through the racks, the warm air rises and is returned to the CRAC units.

1.1.1 Motivation

Various avenues for improving cooling efficiency have been considered. Variations on the HACA them described above include containment for one or both of the hot and cold aisle to prevent recirculation of warm air to the servers [54]. Deploying baffles throughout the room and in the under-floor plenum to obstruct and direct airflow is another strategy that is often practiced as a makeshift solution. Other options include abandoning the HACA organization completely and using a pod based system for orienting the racks [64]. However, regardless of the physical layout of the data center, it is crucial to understand

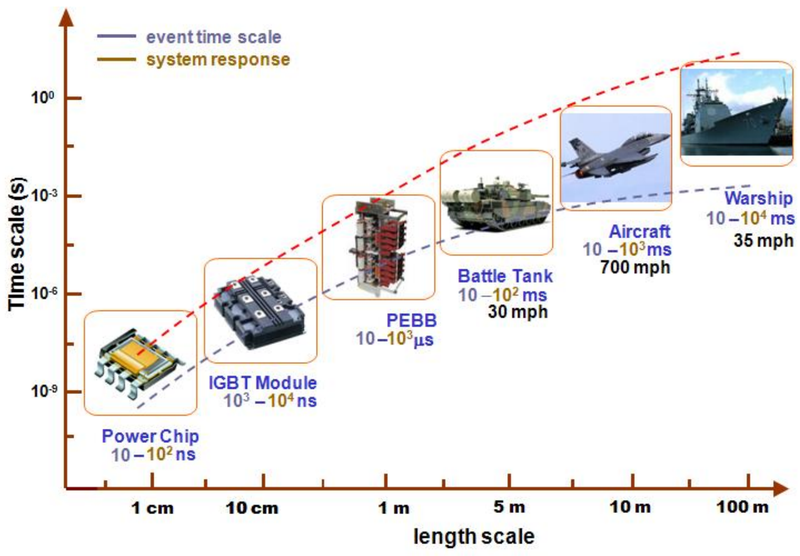


Figure 1.2: The scales in defense thermal management range from nanometers to hundreds of meters. Simulating a system incorporating a wide range of scales can be costly and time consuming.

the airflow patterns and temperature distribution in the data center. Temperature and humidity sensors can be deployed throughout a data center to provide information on the temperatures at a finite number of points, but the empirical nature of this data hinders its predictive capabilities. Using computer simulations, one can predict the temperature and velocity fields everywhere within the data center and anticipate problems before they occur.

For example, defense thermal management systems involve multiple length scales over ten or more decades, making the task of simulating these systems intrinsically difficult. The usual simulation efforts partition a domain into smaller elements and solve a discretized version of the governing equations on each of these smaller elements. If the solution changes rapidly in known regions of the domain, the partition is refined there in order to achieve additional solution accuracy. In defense systems, the multiple length and time scales seen in Figure 1.2 become a primary obstacle to rapid thermal simulations. Computing the solution on an element takes a finite amount of time, and for a large domain with many small elements, classic simulation packages are unable to quickly produce an accurate solution. In simulating these complicated domains, there is a tradeoff between speed and accuracy.

Simulating convection-dominated air flow is intrinsically a difficult problem for a number of reasons. Air flow is often turbulent, and the multi-scale nature of the problem adds another level of complexity. The major heat generating components in an electronic system are chips or packages, often less than 40mm^2 in footprint area. Meanwhile, the other end of the lengths scale spectrum includes aircraft and warships with sizes reaching the kilometers. While there are commercial software packages that are able to model parts of these complicated systems, they are time consuming and expensive from a computing load standpoint. There is a need for fast simulation methods that are able to accurately resolve flow and thermal fields in multi-scale systems on a real time basis.

1.2 Objective

We present this study of the Multi Resolution Discontinuous Galerkin (MRDG) method for the simulation of incompressible flows and with heat transfer. The MRDG method is a combination of two distinct theories: the Discontinuous Galerkin (DG) method for partial differential equations PDE has been well documented since its inception in the 1960s. Various manifestations of the DG method have been successfully applied to fluid flow and heat transfer [46]. The multi-resolution analysis afforded by multiwavelet theory provides an efficient function representation and the associated operators for application as the approximation space. A model problem presenting the advantages of the MRDG method is presented as validation of the theory.

The MRDG method discussed here uses the DG formulation for the spatial discretization. Similar to the continuous Galerkin method, the domain is partitioned into elements and on each element, we solve the weak form of the coupled fluid flow and heat transfer equations. In the continuous Galerkin framework, each element communicates with its neighbor elements by enforcing strict continuity across inter-element boundaries, but in the DG method, continuity across elements is not required as seen Figure 1.3. Instead, solutions are coupled by enforcing appropriate fluxes across elements for the inviscid and viscous flux terms.

In the majority of prior work, the DG method uses Legendre polynomials or Jacobi polynomials as the basis for the solution space on each element [71, 14]. The MRDG

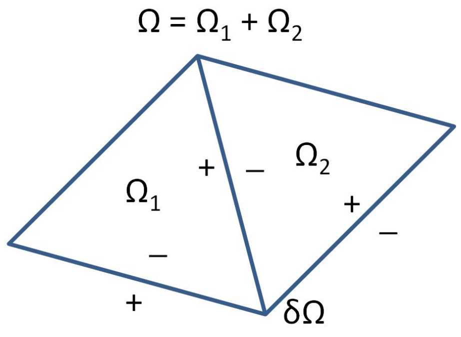


Figure 1.3: The DG method has discontinuities between elements in the domain. This is shown in the example two dimensional domain above.[17]

method uses multiwavelets as the approximation space instead of polynomials, and by doing so introduces a number of key advantages to the method. In general, *a priori* knowledge of the flow field is unavailable, making it difficult to select an optimal partitioning of the domain that would have more elements in areas where the flow field changes rapidly. Using multiwavelets as the approximation space in the MRDG provides a method that tracks the solution and can refine the mesh during run time, providing a significant savings in simulation length and cost while still maintaining the desired accuracy. The advantage of multi-scale resolution over single scale is depicted in Figure 1.4 [63]. In addition to refining the mesh in a solution-driven fashion, the multiwavelets also afford the ability to save computations by coarsening the mesh in areas of the solution where further detail is unnecessary, providing another venue to reduce simulation time.

1.3 Reproducibility of Results

In other branches of academia, experimental results published in journals have the potential to be verified by other practicians. When researchers obtain empirical results with physical experimentation, the details and construction of the experiment are outlined in the paper to provide the reader with a path to follow to reproduce the results. However, in papers about computational methods, the authors often only offer the outline and general spirit of the method presented in the paper in the interest of brevity; the papers themselves end up being mere advertisments for the knowledge and education represented by the software programs,

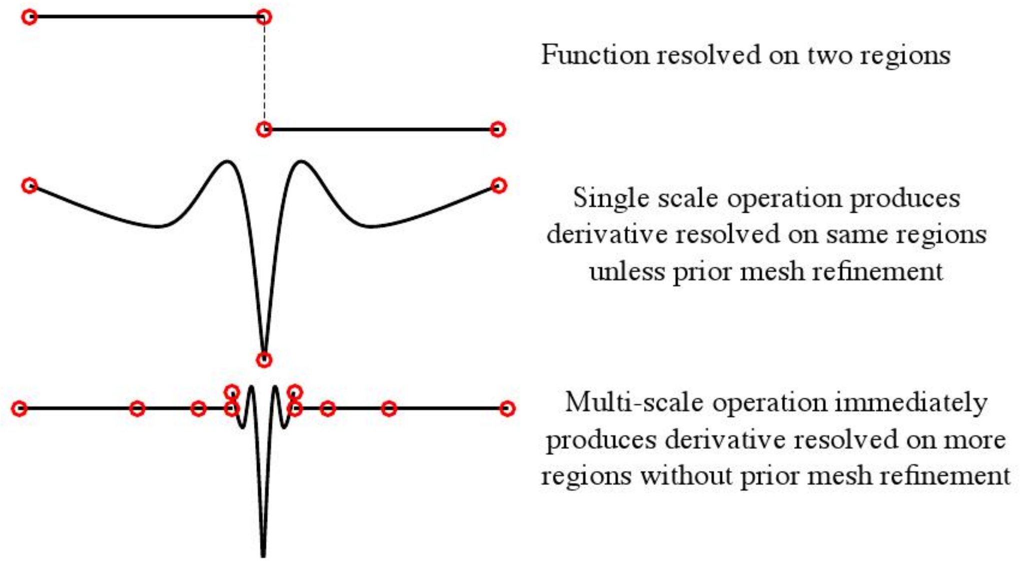

Function resolved on two regions
Single scale operation produces derivative resolved on same regions unless prior mesh refinement
Multi-scale operation immediately produces derivative resolved on more regions without prior mesh refinement

Figure 1.4: The top image is a standard step function. The middle image is an approximation to the weak derivative that is distorted due to the lack of sufficient mesh points. The bottom image displays the advantage of multi-scale operation. From [63]

which is counter-intuitive. As noted by Leveque in a recent paper about this issue [45], experimental results are expected to be reproducible, and yet scientific computing seems to exempt itself from this necessity.

In the spirit of encouraging a similar culture for computational experiments, this section outlines the software that I used to generate my experimental results. First, Table 1 describes the general software programs that I used that were not directly related to the MRDG method. Additionally, Table 2 outlines the particular versions of hedge [40] that I used in my experimental results. As of May 2011, the most recent version of hedge and associated software is acccesible at the Andreas Klockner's personal website: http://mathema.tician.de/software/hedge. The wavelet package was incorporated through the use of PyWavelet, available at http://pypi.python.org/pypi/PyWavelets/.

Table 1: Versions of generic hardware and software used in producing my results

Item	Version	Notes
Processor	Intel(R) Core $^{\mathrm{TM}}$2 Duo CPU E8200	Speed: 2.66 GHz, 32-bit
Operating System	Ubuntu 10.04 “Lucid Lynx”	Linux version 2.6.32-30-generic
GNU C Compiler	4.4.3	Ubuntu 4.4.3-4ubuntu5
Boost C++ library	1.43	
Python	2.6.5	
numpy	1.3.0	
PyTest	2.0.1	
Virtualenv	1.49	

Table 2: Versions of hedge’s direct software dependencies used to generate my results. All projects were on the “master” branch of the tree, except for hedge itself, wherein I was using the staging branch. Versions are provided as SHA1 version hash IDs which are used by the “git” version control system. Each hash uniquely specifies a version for each project.

Name	git Version ID	Notes
aksetup	3f214fa85d08a1783b2e6b92b415d1a1e437ccab	
jhbuild	fd51787375a2206bad7adfe4df29911e51c6bb9b	
pytools	9950c1266d9316e300e949e602b1f9d6463c387e	
pymbolic	0eff7c0726ec1307445ab9b80172c0a2d7ab07a9	
pyublas	46d422f55d4a38282d8eecfb3c59e6ac13712f6e	
meshpy	7ee3b8ce9acfe7380245af9f096d098f94205ae5	
datapyle	cbcba42216b1e53860fb9a8f245e52bf6c5cdda9	
pyvisfile	f5effabd532523e4b9a3f8b0e4fe58ccf7900aed	
codepy	18402a351ad475e52cfe916fbad603863e602c67	
pymetis	0d9c6467ff03e2c580a3be127129082ad4bbe6f2	
pyublasext	07115d184f6b0e127c0b6b0f1498c2a82a8d8ded	
quiet-beam-sampling	89207d2acb7aea4fe26f48ad6c4d729ab2baec03	
hedge	4bc593f9fd6ebceb3818ec0c3adc269748a4585a	Branch: Staging
pywavelet	354bb40e5ce60865315bd284a591da0317431416	

CHAPTER II

PREVIOUS WORK

2.1 Discontinuous Galerkin Methods

The DG method can be considered a natural extension of the finite volume (FV) and finite element methods (FEM) to higher order. However, there are certain distinctions between the DG and other methods: notably, the finite element method generates a solution as a composition of continuous piecewise polynomials. On the other hand, the solution obtained via the DG method is composed of discontinuous piecewise polynomials, but due to this, the DG method is able to more easily handle local mesh refinement [59]. Both methods use the variational formulation of the partial differential equation in question, but the DG method has a number of advantages over both finite difference (FD) and FV methods [63].

Complex geometries are not an obstacle when working with DG methods because the DG method is able to handle unstructured grids. Also, the method is suited to handle non-conforming elements that have multiple hanging nodes, whereas the FE method can only handle one hanging node per element and requires special functions to do so. Also, because each element only requires input from its adjacent elements, the method is easily parallelizable, also simplifying the coding efforts required [50]. Adjusting the polynomial degree for DG methods is also significantly easier when compared to FEM: modifications are only necessary in the routine that calculates the basis functions, whereas the FEM requires different codes for polynomial degrees [59]. In fact, due to the discontinuous nature of the relationship between elements, the degree of the solution and even the equations that govern the system can change from element to element without disrupting the overall method [65]. Moreover, the function space basis used to approximate the solution admits any linear space and does not necessarily have to be the same on all elements or for all time. [71]

2.1.1 Hybrid and Easy Discontinuous Galerkin Environment

Hedge is an open source Discontinuous Galerkin code written primarily by Andreas Klockner [41]. The user-facing interface is written in Python, a high-level scripting-type language, for ease of use, while a smaller core is written in C++ for speed optimization [67]. Hedge comes from an interest in running simulations on a Graphics Processing Unit (GPU) instead of the default and usual way of running fluid simulations on the CPU itself. Due to the intended workload difference between GPUs and CPUs, it becomes obvious that the GPU is well fitted to performing the types of calculations that are crucial in flow simulations. CPUs are tasked with mashalling workloads such as web browsers, word processors, and a wide array of miscellaneous desktop programs - the workload consists of high complexity and a lesser emphasis on parallelization. On the other hand, GPUs have been designed for the application of uniform, reasonably complicated floating point operations to a large amount of data and are often constructed with parallel schema in mind. By using Andreas Klockner's PyCUDA interface, a useful bridge is created bewteen the Python scripting language and the generation of optimized code that runs on Nvidia's Compute Unified Device Architecture (CUDA) [55, 42]. Moreover, Hedge was written with an optional component that allows it to run on parallel infrastructure, which fits in very well with the DG method as well as the wavelet representation. As mentioned previously, the DG method parallelizes easily since interactions between elements is minimal [41].

2.2 Wavelets and Multiwavelets

Wavelets first became a topic of interest in the mathematical and engineering communities in the 1980s, with applications in signal processing, image compression, statistical analysis, and other fields. There have been a number of studies on the use of wavelets for solving partial differential equations [12, 25, 58]; the wavelet-collocation method arose for solutions to the Navier Stokes equations for turbulent flow [10, 48], among others [24]. The Coherent Vortex Simulation (CVS) method [26, 27] and the Stochastic Coherent Adaptive Large Eddy Simulation (SCALES) method [29] are more recent, engaging examples of the usefulness of wavelets in turbulent flow analysis. The basis of these methods is the principle that applying wavelet decomposition to a turbulent flow field provides the capability to identify and resolve only the more energetic eddies, avoiding the cost of resolving the entire flow [6].

The motivation behind incorporating multiwavelet theory into this DG method is slightly different: choosing multiwavelets as the approximation basis allows for adaptive compression of the local solution estimate. Traditionally, DG methods use polynomial bases - Legendre polynomials and Jacobi polynomials are popular choices in literature [49, 36]. However, Yuan et al. explored the use of trigonometric and exponential approximation spaces for DG methods. For certain PDE systems, the alternate basis was more efficient than the "stock" classic orthogonal polynomials [71]. It is in this spirit that we consider using an alternate wavelet basis for the DG method in order to utilize the advantages of wavelets.

2.2.1 Introduction To Wavelets

2.2.1.1 Classic Wavelets

In the classic setting, the scaling function ϕ is defined as

$$\phi(x) = \sqrt{2}\sum_k h_k \phi(2x-k)$$

The h_k are recursion coefficients and define the relationship between scales. Refinement of this function and incorporating wavelets leads to a multi-resolution approximation (MRA). In an MRA, a function is decomposed as the sum of a smooth background and fluctuations, somewhat similar to the idea behind Reynolds-averaging: the mean flow is the smooth background and the turbulent component are the fluctuations. By recursively applying this concept, we construct a hierarchy of scales and are able to decompose the function into the sum of the smooth background and fluctuations; Figure 2.1 displays this concept for a 1D function. At coarse scales, the fluctuations serve as comparatively smooth background, while for smaller scales, the fluctuations are the fine details [51]. The scaling function at each scale represents the smooth background while the wavelets at each scale contribute the fluctuations. Inherent to an MRA are decomposition and reconstruction algorithms that transfer data between scales; generalizing the wavelet theory leads to such functions as ridgelets, curvelets, wavelet packets, frames, second generation wavelets, and others.

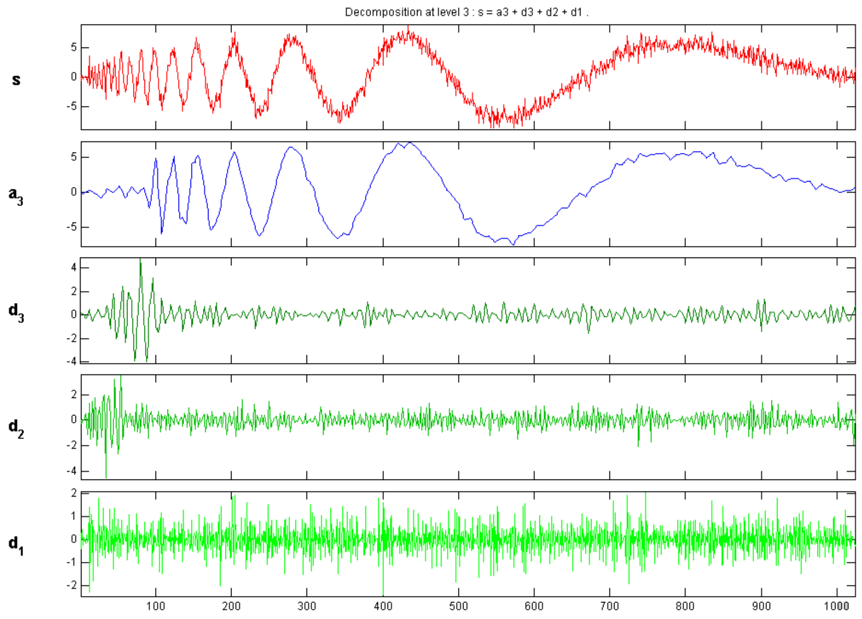


Figure 2.1: The original signal is shown at the top and labeled 's'. The smooth background is shown as 'a_3' and then three detail functions are shown of descending scale. Note how the background function captures the overall motion of the original function, and the detail functions add increasingly more accuracy to the approximation.

2.2.1.2 Multiwavelets

Multiwavelets are one of the generalizations of the classic wavelet theory; they were introduced in the 1990s. Whereas before the scaling function was a scalar, now we consider a vector of scaling functions called a multiscaling function.

$$\Phi(x) = \begin{pmatrix} \phi_1(x) \\ \vdots \\ \phi_r(x) \end{pmatrix}$$

Refinement is accomplished as

$$\Phi(x) = \sqrt{m} \sum_k H_k \Phi(mx - k)$$

where H_k now represents a matrix of size $r \times r$. Multiwavelets introduce some advantages compared to classic wavelets: their support is more flexible than wavelets while affording high smoothness, and it is possible for multiwavelets to be symmetric and orthogonal. Multiwavelets also lend themselves to a similarly constructed multi-resolution approximation and analogous decomposition and reconstruction algorithms like scalar wavelets.

An important characteristic of multiwavelets in this implementation is handling inter-scale effects. Traditionally, adaptive mesh methods would only take into consideration the interactions between the boundary provided and then adjust the mesh as a result of these considerations. However, the multiwavelet construction admits information exchange between elements by moving to a higher or lower scale as necessitated by the problem.

2.2.2 Wavelets and CFD

Wavelet methods are a relatively new research avenue and have been present in the literature only in the past decade. Simulation of turbulent flow and heat transfer is still a major challenge for the scientific community. The application of such simulations has impact in practically every field of science and engineering. Traditionally, direct numerical simulation (DNS) of the governing equations has been limited by the computing power available, which limits the application of such efforts to complicated problems of interest. While waiting for computing power to increase, the only other option is to model some of the turbulent phenomena in order to reduce the compute load. The Reynolds-averaged Navier-Stokes (RANS) approach was one of the early modeling approaches, but it was not able to accurately capture the spatial and temporal interactions at all the pertinent scales [22]. Modifications to the RANS approach include large-eddy simulations (LES) which employed a distinct scale separation by appyling a low-pass filter of sorts to the governing equations. However, neither of these methods really takes advantage of the fundamental characteristics of turbulence: it is multiscale, with coherent structures and intermittent spatial and temporal features.

Introducing wavelet based models comes from the realization that for a particular flow, the coherent structures and important “modes” of the flow are not spread evenly across time or space. A sparse representation of the flow characteristics would take advantage of the fact that fine details are only required intermittently throughout the simulation. Using wavelets to represent the flow can yield a compact description of turbulent flow and heat transfer by only preserving the particular wavelets with high dynamic action, as noted by the magnitude of the coefficients in the representation. Just as there are a wide range of approaches to traditional efforts to model and simulate turbulent heat transfer, there are a number of methods in which wavelets have been introduced to varying degrees of success. There are three main categories, which will be discussed in brief to give an idea of the position from which the MRDG method is presented. Figure 2.2 describes the relationship between some of the classic modeling efforts and the newer wavelet versions of these methods, including some wavelet methods that are not based on classic methodology.

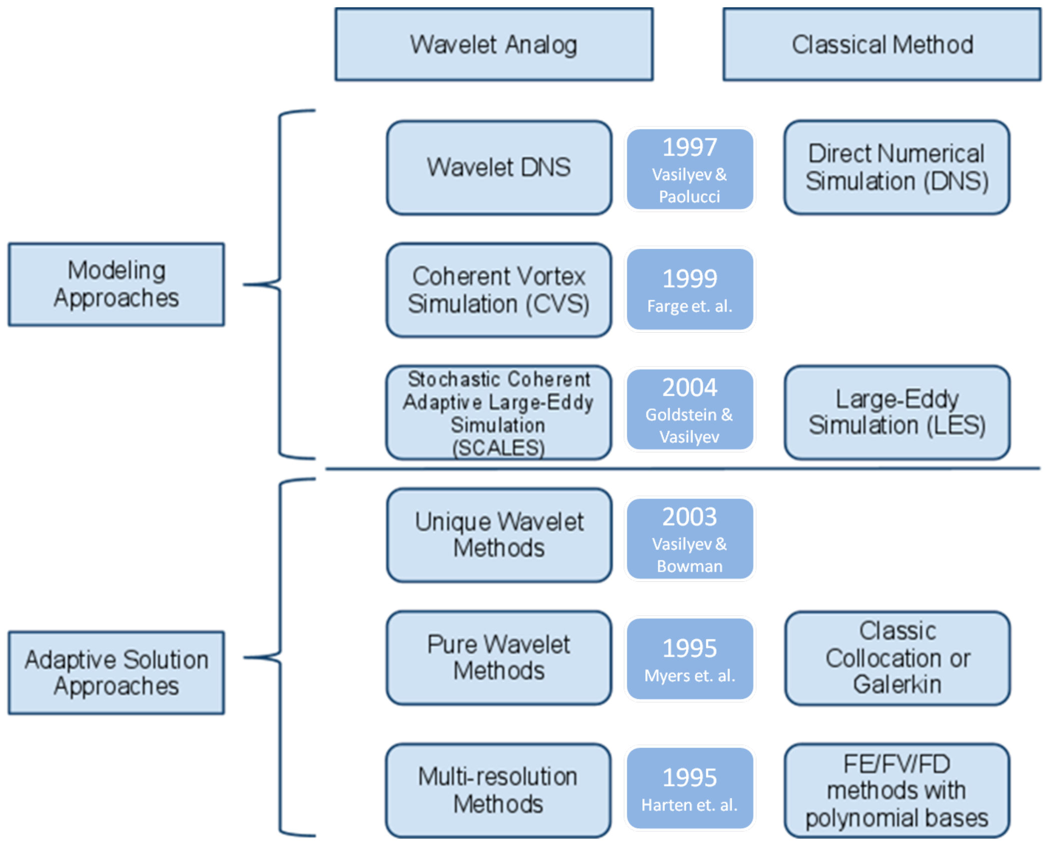

Wavelet Analog
Classical Method
Modeling Approaches
Wavelet DNS
1997 Vasilyev & Paolucci
Direct Numerical Simulation (DNS)
Coherent Vortex Simulation (CVS)
1999 Farge et. al.
Stochastic Coherent Adaptive Large-Eddy Simulation (SCALES)
2004 Goldstein & Vasilyev
Large-Eddy Simulation (LES)
Adaptive Solution Approaches
Unique Wavelet Methods
2003 Vasilyev & Bowman
Pure Wavelet Methods
1995 Myers et. al.
Classic Collocation or Galerkin
Multi-resolution Methods
1995 Harten et. al.
FE/FV/FD methods with polynomial bases

Figure 2.2: Some traditional methods have been modified and improved with the introduction of wavelet representation. Some of the wavelet methods do not have a close analogue to classic methods because they take advantage of particular wavelet characteristics. The middle column in darker blue containing years and names indicate the earliest publications of the respective wavelet method or a major publication that denoted a significant change in methodology.

As seen in Figure 2.2, DNS has a corresponding wavelet version appropriately titled Wavelet DNS (WDNS), which attempts to model all of the modes of the flow using a wavelet representation without using any modeling of small scale activity. Coherent Vortex Simulation (CVS) begins to incorporate modeling for the incoherent components of turbulence. Finally, the Stochastic Coherent Adaptive Large-Eddy Simulation (SCALES) method is an extension of the LES method that provides reduced computational complexity while still maintaining competitive accuracy. The bottom half of the image depicts mesh-adaptive models that include unique wavelet methods and so called "pure" wavelet methods that correspond to simple collocation or galerkin methods. Multi-resolution methods, including the MRDG, are included in this bracket and are a modification of FE/FV/FD methods that use polynomial bases.

2.2.3 Wavelet Turbulence Modeling

WDNS is the method that carries the highest degree of precision and a correspondingly high degree of computational complexity. By taking advantage of the compact nature of wavelet representation, the cost and memory requirements of the method are kept in check without performing modeling for the small scales. Starting in 2000, WDNS has been successfully employed for a wide array of flows for incompressible and compressible flow, including flow around cylinder(s) [61], 2D and 3D homogeneous turbulence [56], flow in a differentially heated cavity [70], and even combustion and thermoacoustic wave propagation [57].

2.2.3.1 Wavelet-Based Direct Numerical Simulation

An important feature of the WDNS method in particular takes advantage of the intermittent nature of the flow characteristics, and this is one area where the wavelet methods are able to reduce computational complexity. The previous computational estimates for the number of grid points required in a simulation of 2D and 3D decaying turbulence has been found to be $\mathcal{N} \propto Re^{\frac{3}{2}}$ and $\mathcal{N} \propto Re^3$, where $\mathcal{N}$ is the number of grid points. As a comparison for how the WDNS performs in 2D, the following relationships were found for the sample problem of for impusively started flow through a tightly packed cylinder array. The number of active grid points was found to satisfy $\mathcal{N}_{WDNS} \propto Re^{\frac{1}{2}}$, and computational complexity scaled similarly to Re; both of these relationships held over five orders of magnitude of $Re : 3 \times 10^1 \leq Re \leq 10^5$ [37]. Also, further studies in 2D turbulent decay confirmed the advantage of WDNS over previous methods, with the wavelet enhanced method boasting spatial modes scaling similar to $Re^{0.7}$ compared to the traditional estimate of $O(Re)$ [62].

2.2.3.2 Coherent Vortex Simulation

Coherent Vortex Simulation, CVS, was introduced in 1999 by Farge and colleagues [27], and was used to simulate turbulent flows. The driving idea behind this method was to extract coherent vortices in a mathematically objective fashion when looking at turbulent flows. Using a wavelet basis, the evolution of coherent vortices is calculated deterministically. The computation adapts to the spatial and temporal domains with strong gradients, while the incoherent components of the flow are modeled and discarded at each step to mimic the turbulent diffusion. CVS separates itself from LES by using wavelet threshold filters instead of the linear low-pass filters; the LES filters do not adapt to the flow as it changes, whereas the CVS filters depend on the current flow parameters. In a comparison between CVS and DNS for a time-dependent 3D turbulent mixing layer, the ratio of retained coefficients between DNS and CVS was found to range from $8\% to 15\%$ - that is, DNS retained over five times as many coefficients as the CVS model. Meanwhile, comparing the timewise evolution of the energy from each method showed that the CVS method maintained 99.6% of the energy described in the DNS simulation [60].

2.2.3.3 Stochastic Coherent Adaptive Large-Eddy Simulation

In the spirit of further reducing the computational complexity, the SCALES method was developed and introduced a higher degree of modeling compared to the CVS method [29]. The distinction between foreground and background flow characteristics was shifted towards the more energetic characteristics, meaning that the new "background" flow makes a relevant contribution to the flow and cannot be neglected. Introducing a model for the background flow makes the method similar to LES, but unlike LES, SCALES couples the grid and the SGS model, meaning that the local resolution of the mesh can be increased or decreased when necessary, making the method more responsive to changes in the flow realization. Due to the analogy between SCALES and LES, some of the same models used in LES methods have been used in the SCALES method, the most successful of which being a Lagrangian path-line/tube dynamic model [68].

In comparisons between DNS, WDNS, CVS, and SCALES, the advantages of the CVS and SCALES methods become obvious. Both approaches were found to match the DNS energy and enstrophy density spectra for the primary wave-numbers using much fewer degrees of freedom - the resolved kinetic energy for both CVS and SCALES was very close to that of the DNS prediction [62]. In a comparison of the percentage of compression, or discarded points, the SCALES method with two separate sub-models outperformed the compression of the standard LES method by four times. The SCALES method employed 99.6% compression compared to the DNS model, using only 0.4% of the points in comparison to the LES method which used 1.6%. [68, 66].

2.2.4 Wavelet-Based Numerical Methods

As seen in Figure 2.2, the family of wavelet methods can be grouped into three families: unique wavelet methods, pure wavelet methods, and multiresolution methods. The unique methods are new approaches to the numerical solution of PDEs and are dissimilar from previous methods. Pure wavelet schemes are called so because the wavelets are used for direct discretization nof the PDEs. Finally, multiresolution methods compose the family of methods that are based on finite-difference, finite-volume, or finite-element analysis but introduce wavelets as an improvement method. This last group is the family in which the MRDG method resides.

2.2.4.1 Unique Wavelet Methods

There are two distinct methods that are considered unique in their approaches. The first is the Lagrangian Wavelet method, which is an extension of adaptive wavelet methods and uses travelling wavelets [5]. This method is unique because both the position and scale of the wavelet bases change continuously in time; ideally this would provide the most compact representation of the flow because the basis readily adapts to any and all flow characteristics as the solution iterates through time. Although the method was found to work well for linear problems, its extension to nonlinear systems was difficult due to the problem of wavelet collision [9].

While the majority of methods, both wavelet and otherwise, depend on adapting to a problem in space, fewer methods are time-adaptive. Instead of using the timestep to control stability or limit the error, space-time wavelet methods introduce different time-adaptive stepping strategies. The first time adaptive wavelet methods came in 1992, when a scale-dependent method was presented for the Burgers' equation. [4]. More recently, Domingues et al. applied time-adaptive methods to the compressible Euler equations [20, 21]. Although the methods have some advantages, they still accrue error over time - Alam et al. presented a modified version of the approach that featured an extreme reduction in gridpoints while achieving similar global accuracy. However, the tradeoff came at the cost of a significantly larger memory requirement [1].

2.2.4.2 Pure Wavelet Methods

Pure wavelet methods consist of both Galerkin and collocation methods. Here, wavelets are used to directly discretize the operators in the PDE, or the properties of the wavelets are used to optimize the method, via grid adaption, pre-conditioning, or other schemes. The adaptive Galerkin methods attempts to find a solution that in the form of a wavelet basis, solving for the coefficients of the different wavelets and discarding the coefficients lesser than an absolute threshold. Wavelet Galerkin methods were primarily of focus in the 90s, with a number of studies describing different ways of implementing the method and in particular the evaluation of the nonlinear terms [53, 34, 62].

The wavelet collocation method avoids some of the issues encountered in the wavelet Galerkin methods, namely the difficulties that arise with the nonlinear terms and the issue of treating general boundary conditions. A distinction of the wavelet collocation method is its grid adaptation scheme - in a similar fashion to the previous method, the wavelets with trivial coefficients are discarded, but in the collocation method, the grid points that correspond to the discarded wavelets are also discarded. The collocation grid points are only preserved from time step to time step if the wavelet that corresponds to that grid point has a nontrivial coefficient [62].

2.2.4.3 Multiresolution Methods

Multiresolution methods were first developed in 1994 for hyperbolic conservation laws [31]. Recently, Harten's approach has been extended and developed in a number of studies with various applications [16, 52, 63]. As previously mentioned, multiresolution methods use a hierarchical representation of the data that is very well matched to the multiscale nature of turbulent flow. Using the wavelet basis to represent the flow, information about local smoothness of the flow can be deduced from the wavelet coefficients; in areas where the solution is smooth, additional coefficients can be discarded and the compression of the representation is improved. This gives way to an adaptive grid algorithm that uses coefficient thresholding to only retain the significant wavelets and their coefficients. Moreover, since the grid adaption error can be estimated as a function of the threshholding method, it is possible to control for the error, providing an extra advantage for multiresolution methods [62].

CHAPTER III

MULTI-RESOLUTION DISCONTINUOUS GALERKIN METHOD

In order to describe the MRDG method, we follow the previous works by Shelton [63], Li [46], Klockner [41], and Hesthaven [32]. We begin by outlining the MRA and the tools necessary for its application. Next, we describe the DG method and the pertinent items for the solution of the governing equations.

3.1 Multi Resolution Approximation

3.1.1 Describing the Approximation Space

Discretization of the problem domain Ω is the one of the first steps we will consider. We partition the aforementioned channel into a set of non-overlapping elements Ω_e of characteristic size h_e where the interface between any two given elements is either a point, an edge, or nothing:

$$\Omega = \bigcup_e \Omega_e, \qquad \Omega_e \cap \Omega_{e'} = \emptyset \quad \text{for} \quad e' \neq e \tag{3.1}$$

In two dimensions, we will consider a triangular canonical element, but quadrilaterals are also an option (tetrahedrals are the corresponding three dimensional analogy). Next, on each triangular element Ω_e we consider a further partition by n midpoint subdivisions of the element as depicted in Figure 3.1 as a starting point for the multi resolution approximation [63].

$$\Omega_e = \bigcup_{h=0}^{2^{2n}-1} \Omega_h, \qquad \Omega_h \cap \Omega_{h'} = \emptyset \quad \text{for} \quad h' \neq h \tag{3.2}$$

On each newly partitioned element, the continuous function space is approximated as a vector space $\mathcal{V}_p^n$. $f \in \mathcal{V}_p^n$ if f is polynomial of degree at most p and f is compactly supported. The spaces $\mathcal{V}_p^n$ form a nested sequence of closed subspaces:

$$\mathcal{V}_p^0 \subset \mathcal{V}_p^1 \subset \cdots \mathcal{V}_p^n \cdots \tag{3.3}$$

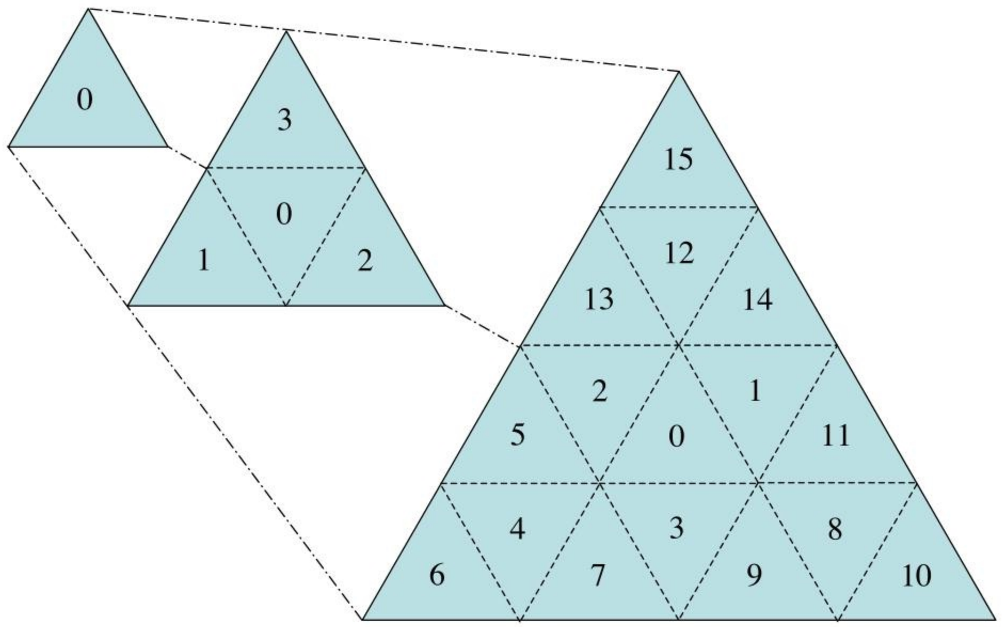


Figure 3.1: Partitioning an element via recursive mid-point division, and the numbering method for the elements within. From [63]

Next, we can define the space $\mathcal{W}_p^n$ as the orthogonal complement of $\mathcal{V}_p^n$ in $\mathcal{V}_p^{n+1}$:

$$\mathcal{W}_p^n = \mathcal{V}_p^{n+1} \ominus \mathcal{V}_p^n \tag{3.4}$$

This can be rewritten as $\mathcal{W}_p^n \oplus \mathcal{V}_p^{n+1} = \mathcal{V}_p^n$. Here, $\mathcal{W}_p^n$ is the *details space* which corresponds to the *averages space* $\mathcal{V}_p^{n+1}$; these two vector spaces are orthogonal to each other and their sum is the averages space of the higher scale [18]. As a result, we can compose any averages space $\mathcal{V}_p^n$ as the sum of a single averages space $\mathcal{V}_p^0$ along with a sequence of details $\mathcal{W}_p$:

$$\mathcal{V}_p^n = \mathcal{V}_p^0 \oplus \mathcal{W}_p^0 \oplus \mathcal{W}_p^1 \oplus \cdots \oplus \mathcal{W}_p^{n-1} \tag{3.5}$$

Now, we can define the multiscaling functions that span $\mathcal{V}_p^n$ and the multiwavelet functions that span $\mathcal{W}_p^n$ given a basis for the spaces $\mathcal{V}_p^0$ and $\mathcal{W}_p^0$. Assuming $\phi_0, \ldots, \phi_{k-1}$ spans $\mathcal{V}_p^0$, then any space $\mathcal{V}_p^n$ has the following for a basis, obtained by dilating and translating the original basis:

$$\phi_j^{nh}(\tau) = c_h \phi_j(\mathcal{T}_{\Omega_c \to \Omega_h} \tau) \tag{3.6}$$

The operator $\mathcal{T}_{\Omega_c \to \Omega_h}$ maps the coordinates of the actual element to a bi-unit triangle and

τ refers to the coordinates. c_h is a dilation factor:

$$c_h = \sqrt{\frac{|\Omega_c|}{|\Omega_h|}} = 2^n \tag{3.7}$$

A similar construction is available for $\mathcal{W}_p^n$, with the basis $\psi_{l,0}, \dots, \psi_{l,k-1}$, where in two dimensions, $l = 1, \dots 3$

$$\psi_{l,j}^{nh}(\tau) = c_h \psi_{l,j}(\mathcal{T}_{\Omega_c \to \Omega_h} \tau) \tag{3.8}$$

As previously discussed, the ϕ_j functions are the multiscaling functions and the $\psi_{l,j}$ are the multiwavelet functions. By construction, the following orthonormal properties hold [63]:

$$\langle \phi_i, \phi_j \rangle = \delta_{ij}, \qquad \langle \psi_{l,i}, \psi_{l',j} \rangle = \delta_{ij}\delta_{ll'} \tag{3.9}$$

These properties extend to the multiwavelets that were constructed via dilation and translation:

$$\langle \psi_{l,j}^{nh}, \psi_{l',j'}^{n'h'} \rangle = \delta_{ij}\delta_{ll'}\delta_{nn'}\delta_{hh'} \tag{3.10}$$

Since $\mathcal{W}_p^0 \perp \mathcal{V}_p^0$ follows from the definition of $\mathcal{W}_p^0$, the basis functions ϕ_i and $\psi_{l,j}$ are also orthogonal: $\langle \phi_i, \psi_{l,j} \rangle = 0$. Therefore, a complete orthonormal basis for $\mathcal{V}_p^n$ is the set $\{\phi_j\} \cup \{\psi_{l,j}^{nh}\}$ with modes $j = 0, \dots, k-1$, dimensional connections, $l = 1, 2, 3$, scales $m = 0, \dots, n-1$ and subregions $h = 0, \dots, 2^{2m} - 1$.

3.1.2 Series Approximation

The efficient function decomposition is based on the idea that when a given function f is sufficiently smooth at some scale m', the details at any further scale $m > m'$ are small enough to be ignored, and the rest of the decomposition at all further scales can be neglected. The truncation of the decomposition based on the size of the details leads to a reduction in terms necessary to approximate the function.

In order to approximate a function on the spaces outlined above, we can project the function onto the space $\mathcal{V}_p^0$ as well as the spaces $\mathcal{W}_p^n$. The approximation of the function is then the sum of these projections. To decompose a function $f \in \mathcal{V}_p^n$:

$$f(x) \approx \tilde{f}(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k-1} s_{jl}^n \phi_{jl}^n(x) \tag{3.11}$$

Here, ϕ_{jl}^n are the scaling functions that compose the basis of each space $\mathcal{V}_p^n$, and the coefficients s_{jl}^n are found from the inner product of the basis functions and the original function $f(x)$:

$$s_{jl}^n = \int_{2^{-n}l}^{2^{-n(l+1)}} f(x)\phi_{jl}^n(x)dx \tag{3.12}$$

Of course, there is an equivalent representation using only one averages space $\mathcal{V}_p^0$ and the complementary details spaces $\mathcal{W}_p^n$.

$$\tilde{f}(x) = \sum_{j=0}^{k-1}(s_{j0}^0\phi_j(x) + \sum_{m=0}^{n-1}\sum_{l=0}^{2^m-1} d_{jl}^m\psi_{jl}^m(x)) \tag{3.13}$$

Again, the coefficients d_{jl}^m are found as the inner product of the original function $f(x)$ and the basis functions $\psi_{jl}^n(x)$, analogous to Equation 3.12.

While the total number of expansion coefficients in the averages space decomposition, Equation 3.11, and the details space decomposition, Equation 3.13, are the same, the number of significant coefficients in Equation 3.13 for a given error tolerance ϵ is different. The development of fast transforms between the two different expansions was demonstrated in [2], and allows for efficient transition between scales when necessary [3]. The advantage of using the multiwavelet expansion is that fewer significant expansion coefficients are needed, so the computational speed and efficiency of the method are increased [2]. The thresholding technique is similar to that found in [11, 63] wherein

$$||f - \tilde{f}||_\infty < \epsilon \tag{3.14}$$

3.1.3 Wavelet Thresholding

In signal denoising applications, wavelet thresholding is applicable and advanageous because the scale of the coefficients in the signal decomposition is similar to that of the noise that is clouding the signal. Considering the decomposition of a noisy signal, setting “small” wavelet coefficients that are less than a threshold $\epsilon << 1$ and reconstructing the signal effectively removes the noise and can return a continuous version of the signal without the noise. Because the noise contributes information that is the same scale as the smaller coefficients, discarding the coefficients effectively discards the noise as well, resulting in a signal is denoised. Figure 3.2 demonstrates this nicely, showing a signal, its wavelet coefficients, the thresholded coefficients, and the denoised signal in that order [35].

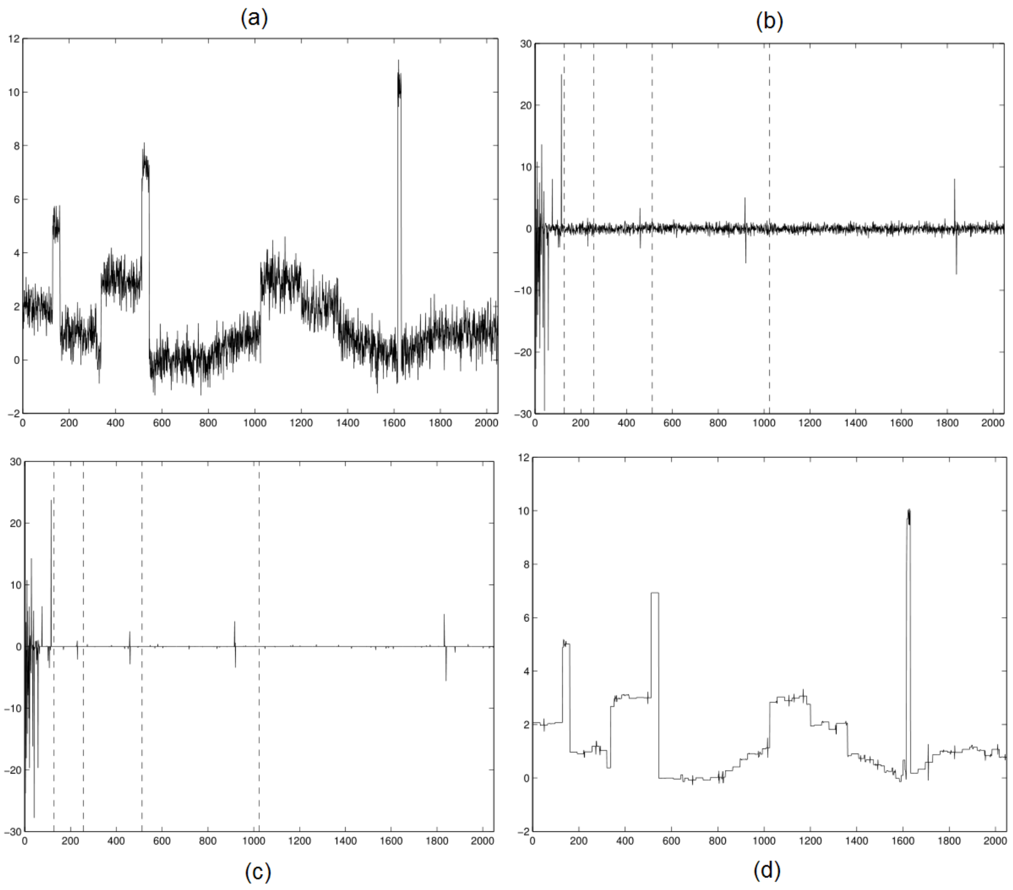


Figure 3.2: (a) The original signal is in the top left, containing stationary and white noise. (b) The signal's Haar transform. The noise is distributed evenly across the entire wavelet coefficient spectrum, and the singularities in the original signal are still distinguishable in the wavelet coefficients from the noise. (c) Wavelet coefficients after soft thresholding (d) The reconstructed signal, with significantly less noise. Additional details available in [35]

Although denoising a signal is a different application from computational fluid dynamics, the concepts are similar. There are two main methods of modifying the coefficients: hard- and soft- thresholding. In hard-thresholding, small coefficients are removed while large coefficients are unchanged. The distinction between “small” and “large” in this sense depends on the concern for reconstruction accuracy and the competing interest of computational efficiency. Soft thresholding involves shrinking the coefficients larger than the threshold, moving the entire signal towards zero in the interest of producing a continuous reconstructed signal. For data compresion in our application, hard-thresholding is the method of choice.

Impressive compression rates have been achieved in literature for wavelet methods. For a variety of test problems subject to the Euler equations using the MRDG method, Shelton achieved compression rates of 75% DoF in one dimension and 96% DoF in two dimensions, with predictions of even better compression in three dimensions [63]. In a review of wavelet methods for CFD, the SCALES and CVS methods achieved compression rates of over 99% DoF, outperforming the LES method [66, 56, 68, 62] from both an efficiency and accuracy standpoint. While the majority of the simulations in the later sections were not run with compression as a prime objective, a small number of tests were run with different thresholding values. Compression rates for the simplified method presented here were in the range of 75% compression. While this is significantly lower than current state of the art, it is comparable to some of the early methods in the literature, and is also subject to a number of available improvements that would increase efficiency.

3.2 The Discontinuous Galerkin Method

Discontinuous Galerkin (DG) methods [14, 32] are an interesting synthesis of ideas from finite volume and spectral element methods. The method consists of two separate halves: at higher detail, the method is high-order by design, but at a certain level of detail, the method switches to a decomposition onto computational elements, coupling these elements using a finite-volume like surface Riemann solver [41].

DG methods have been particularly suited to solution of hyperbolic systems of conservation laws such as

$$u_t + \nabla \cdot F(u) = 0 \tag{3.15}$$

where the domain of interest is $\Omega = \uplus_{k=1}^{K} D_k \subset \mathbb{R}^d$ where the D_k composing the domain are disjoint triangles or tetrahedra. The associated boundary conditions are expressed as

$$u|_{\Gamma_i}(x,t) = g_i(u(x,t), x, t), \qquad i = 1, \cdots, b$$

at the boundaries $\uplus \Gamma_i \subset \partial\Omega$. For simplicity's sake, if we assume the flux function in Equation 3.15 is linear, the weak form of 3.15 on each element D_k is

$$\begin{aligned} 0 &= \int_{D_k} u_t \varphi + [\nabla \cdot F(u)]\, \varphi dx \\ &= \int_{D_k} u_t \varphi - F(u) \cdot \nabla \varphi dx + \int_{\partial D_k} (\hat{n} \cdot F)^* \varphi dS_x \end{aligned} \tag{3.16}$$

Here, φ is a test function, and $(\hat{n} \cdot F)^*$ represents the numerical flux in the unit normal. Proceeding as in [32], the strong form of the system becomes

$$0 = \int_{D_k} u_t \varphi + [\nabla \cdot F(u)]\, \varphi dx - \int_{\partial D_k} [\hat{n} \cdot F - (\hat{n} \cdot F)^*]\, \varphi dS_x \tag{3.17}$$

In the standard DG, the solution to this equation is a vector $u^k := u_N|_{D_k}$ from the space of polynomials of maximum degree N, $P_N^n(D_k)$. As the scalar test function $\varphi \in P_N(D_k)$ is from the same basis, the method is of course of Galerkin-type. To be explicit, the solution in each cell would be a piecewise-polynomial. Representing both the solution and test function by expansion in a basis of $N_p := dim P_N(D_k)$ Lagrange polynomials L_i [69, 41], the mass, stiffness, differentiation, and face mass matrices are defined as follows.

$$M_{ij}^k := \int_{D_k} l_i l_j dx, \tag{3.18}$$

$$S_{ij}^{k,\partial v} := \int_{D_k} l_i \partial_{x_v} l_j dx, \tag{3.19}$$

$$D^{k,\partial v} := (M^k)^{-1} S^{k,dv}, \tag{3.20}$$

$$M_{ij}^{k,A} := \int_{A \subset \partial D_k} l_i l_j dS_x \tag{3.21}$$

We can re-write 3.17 using the matrices as

$$\begin{aligned} 0 &= M^k \partial_t u^k + \sum S^{k,\partial_v}[F(u^k)] - \sum_{F \subset \partial D_k} M^{k,A}[\hat{n} \cdot F - (\hat{n} \cdot F)^*], \\ \partial_t u^k &= -\sum_v D^{k,\partial_v}[F(u^k)] + L^k[\hat{n} \cdot F - (\hat{n} \cdot F)^*]|_{A \subset \partial D_k} \end{aligned} \tag{3.22}$$

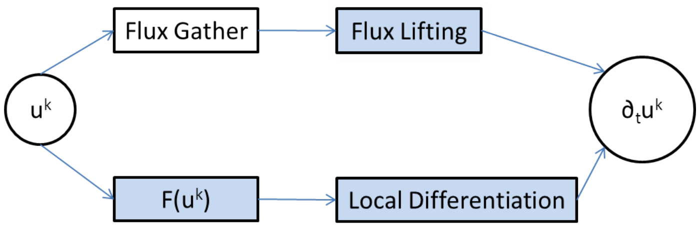


Figure 3.3: Decomposition of a DG operator into its different subtasks; operations that are element local and do not depend on global communication are highlighted in blue. More details regarding the construction and specific implementation of these operators is available in [41]

The lifting matrix L^k included in 3.22 acts on vectors of the shape $[u^k|_{A_1}, \cdots, U^k|_{A_4}]$. Here, $u^k|_{A_i}$ is the vector of the degrees of freedom on face i, and L^k performs a compositon of functions. The lifting matrix applies each face's mass matrix, embeds the facial values back into a volume vector, and finally applies the inverse volume mass matrix [41]. The implementation of the DG method is aided by the fact that DG decomposes naturally into four stages. Figure 3.3 visualizes the decomposition and in particular notes that the majority of the operations are element-wise local and are feasible without global communication. The element-wise nature of the operations simplifies the parallelization of the scheme, and also distinguishes DG from other finite element methods.

3.3 Time Stepping Method

The major focus of this work is taking advantage of the multi-scale nature of the turbulence problem in the spatial dimensions. While there are many applications that exist that have a multi-scale temporal dimension, the test cases and typical applications that we would encounter in a data center are not multi-scale in time. As a result, a simple fourth order Runge-Kutta multi-stage time-stepping method is implemented for use. As mentioned in Chapter 2, certain multi-resolution wavelet methods have also incorporated an adaptive time step, but because our focus is mainly spatial resolution, we will not be taking advantage of an adaptive time step. The details for the RK4 method are widely available, and specific details for the implementation can be found in [63].

3.4 Choice of Flux

Due to the nature of the discontinuous global space, the numerical flux function carries information across the boundaries of elements. In a one dimensional case, an interface would have a “left” element and a “right” element. While the choice of a flux function depends on the application, the flux function must satisfy identity and reflexive relationships:

$$\mathcal{F}_n(U, U) = \mathcal{F}_n(U) \tag{3.23}$$

$$\mathcal{F}_n(U^L, U^R) = \mathcal{F}_n(U) \tag{3.24}$$

When using artificial viscosity, simple averages composed of equal influence from the left (L) and right (R) sides are implemented

$$\mathcal{U} = \frac{\mathcal{U}^L + \mathcal{U}^R}{2} \tag{3.25}$$

$$\mathcal{F}_n^{av} = \frac{\mathcal{F}_n^{av}(U^L) + \mathcal{F}_n^{av}(U^R)}{2} \tag{3.26}$$

There are also other choices available for the viscous flux function in a DG scheme, each bringing their own advantages and disadvantages to the table. Zhang and Shu presented an inconsistent and weakly unstable flux method that was used as a baseline for comparison of flux methods [72, 38]. Commonly used flux choices include the Bassi-Rebay scheme (BR) [23], the local discontinuous galerkin method (LDG) [15], the Baumann-Oden scheme (BO) [7], and the Lax-Friedrichs flux [63]. The different flux schemes differ in stencil size and the related amounts of “communication” between elements - LDG and BO have tighter stencils and thus less communication than BR, for example. On the other hand, the LDG method requires a smaller timestep, making it less efficient in that sense [38]. Of course, there are a number of other potential choices that exist, some of which can be found in [44] and the references within.

In our case, the Lax Friedrichs flux has been implemented. In classical finite volume methods, the flux matching across the element interfaces was crucial in order to propagate the boundary information to the interior of the domain, and also to keep the solution within

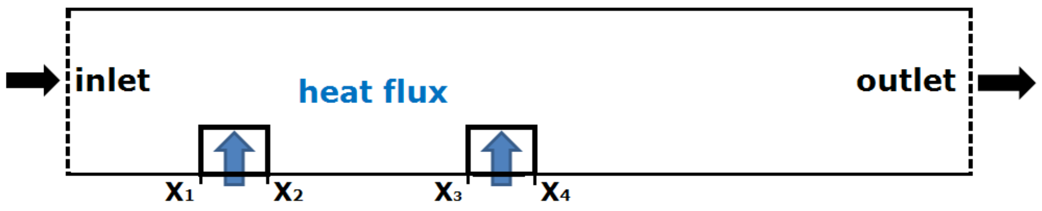


Figure 3.4: The dimensions and layout of the model problem, representing a slice of a server.

reasonable bounds. However, in the DG method, the choice of flux is less influential for two reasons. First, the element update is not entirely dependent on the interface integral, and secondly, for higher order DG methods, the discontinuities across elements tends to be increasingly small. The Lax-Friedrichs flux was chosen because it performs well compared to traditional methods and its efficiency over more complex methods [63]. The function is

$$\mathcal{F}_n = \frac{F_n(U^L) + F_n(U^R)}{2} + \frac{\Delta x}{\Delta t}\frac{U^L - U^R}{2} \tag{3.27}$$

3.5 Incompressible Navier-Stokes Equations with Heat Transfer

The problem of application is the flow of air horizontally through a server encased in an electronics rack. In practice, air flow from the front of a rack through the servers and out the back is a turbulent, incompressible, viscous, three dimensional flow. The air moves through the server and across the heated components on the motherboard due to the fans in the server that create a pressure gradient. For the purposes of this paper, we shall consider a model problem: a two dimensional simplification where we assume flow across a given slice of the server is indicative of the overall flow. The representative 2D problem that we consider is essentially pressure driven channel flow with a heat flux on bluffs on the bottom wall representing the heated components in a server such as the CPU or the RAM. The governing equations for fluid flow and heat transfer in terms of velocity and temperature fields and are derived from the conservation of mass, momentum, and energy, along with a number of constitutive relations and are presented here in vector form.

$$\begin{aligned}
&\nabla \cdot \mathbf{u} = 0 \\
&\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) - \nabla p + \mathbf{f} - \rho\mathbf{g}\beta(T - T_r) \\
&\rho C_p \frac{\partial T}{\partial t} + \rho C_p \mathbf{u}\cdot\nabla T = \nabla\cdot\kappa\nabla T + Q + \Phi
\end{aligned}$$

The unknowns are the velocity $\mathbf{u}$, the pressure p and the temperature T. $\mathbf{f}$ is the body force, not including gravity, and ρ is the density and μ is the molecular viscosity. β is the thermal expansion coefficient, and g is the acceleration due to gravity. C_p is the specific heat and constant temperature and κ is the thermal conductivity.

Since we will be considering a two dimensional forced convection case, the governing equations can be written explicitly in the following non-dimensionalized manner:

$$\begin{aligned}
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\
u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\
u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -\frac{1}{\rho}\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \\
u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} &= \frac{1}{RePr}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)
\end{aligned}$$

Here, $U = U_\infty$ scales the velocity. The length scale is L_x and the time scale is $\frac{L_x}{U}$. Re denotes the Reynolds number: $Re = \frac{\rho U L_x}{\mu}$ and $Pr = \frac{\nu}{\alpha}$ is the Prandtl number and α is the thermal diffusivity. Finally, the nondimensionalized temperature is $T = \frac{(T^* - T_1^*)}{T_2^* - T_1^*}$ where the superscript asterisk refers to the dimensional temperature.

The boundary conditions that will be applied are as follows. The velocity is zero on the top and bottom walls of the channel and the inlet velocity is uniform. The top wall is insulated and the bottom wall is mostly insulated except for two regions where the heat flux is nonzero to represent the heated components in a server. The inlet temperature will be T_R, a reference temperature. There will be two initial conditions considered. The velocity may be zero inside the channel and the temperature in the channel is T_R; physically, this denotes a state where the server is off and the fans are not pulling air across the components.

At $t = 0$, we can presume that the server is turned on and initiates the air flow. Another condition is uniform flow through the channel, as if driven by the CRAC unit; the simulation would describe the airflow as the server's fans are activated. At the outlet, we will apply the condition that the gradients of the pertinent variables are zero, implying that the perturbations to the system are far from the outlet.

$$u(x, y = 0) = 0; \quad v(x, y = L_y) = 0; \tag{3.28}$$

$$\frac{\partial u}{\partial x}(x = L_x, y) = 0, \quad \frac{\partial v}{\partial x}(x = L_x, y) = 0 \tag{3.29}$$

$$\frac{\partial T}{\partial y}(x, y <= 0.25) = \begin{cases} 0 & \text{if } x \notin [x_1, x_2] \cup [x_3, x_4] \\ 1 & \text{if } x \in [x_1, x_2] \cup [x_3, x_4] \end{cases} \tag{3.30}$$

$$\frac{\partial T}{\partial y}(x, y = L_y) = 0, \quad \frac{\partial T}{\partial x}(x = L_x, y) = 0 \tag{3.31}$$

$$u(x, y, t = 0) = 0, \quad v(x, y, t = 0) = 0, \tag{3.32}$$

$$T(x, y, t = 0) = T_{ref} \tag{3.33}$$

Here, the regions $[x_1, x_2]$ and $[x_3, x_4]$ denote the portions of the channel where the heat flux is nonzero on the bottom wall. The length of the channel is L_x and the height of the channel is L_y.

CHAPTER IV

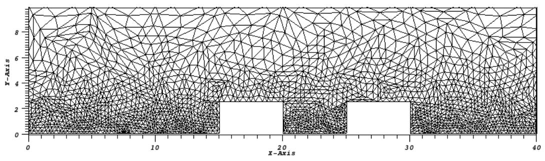
SIMULATION RESULTS

A number of different simulations have been run for comparison purposes. Although it is difficult to present an exhaustive battery of test cases due to the cost of solving the problem, the results are promising. The simulations include the DG method using Legendre polynomials as the basis, and a simplified version of the full MRDG method, as well as various methods used to verify the accuracy and precision of the MRDG method. ANSYS® FLUENT simulations are also provided as a benchgmark for comparison.

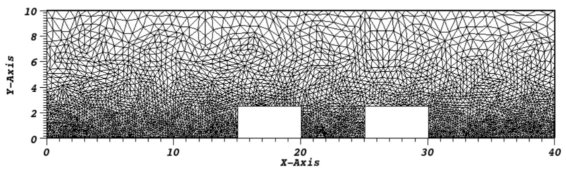
4.1 Mesh Refinement Comparison

The mesh used in the majority of the simulations by the MRDG and DG code is displayed in Figure 4.1a, and the refined mesh is displayed in Figure 4.1b. Mesh generation for the MRDG simulation is accomplished by MeshPy and PyMetis [40].

The initial velocity profile condition for this test case was uniform flow throughout the channel: $u(x, y, 0) = 1, v(x, y, 0) = 0$. The simulation represents the development of the flow over nine seconds, and as seen in Figure 4.2, the simulation displays many of the same qualities at both mesh refinement levels, and does not deviate significantly from the refined mesh. It may be noted here that the initial condition of a uniform flow that is propagated evenly throughout the channel is a slightly unrealistic condition for the velocity profile.

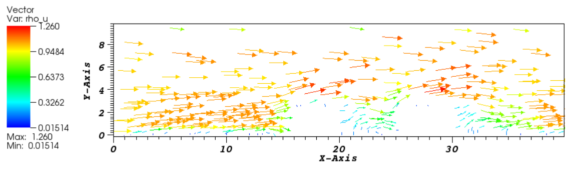


(a) Mesh of channel with bluffs

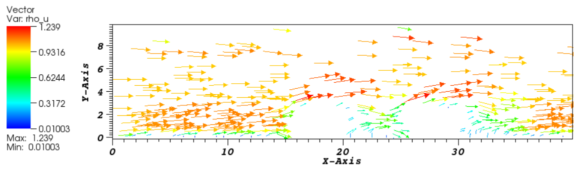


(b) Refined mesh of channel

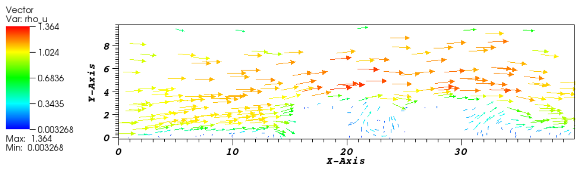
Figure 4.1: A comparison between the basic mesh used in the simulations and the refined mesh used in this section for verification.



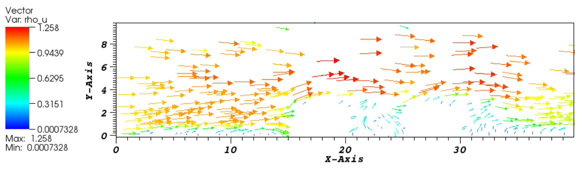
(a) Coarse mesh: t = 2.9s



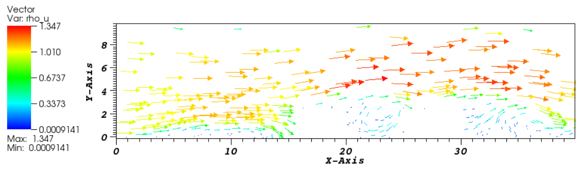
(b) Refined mesh: t = 2.9s



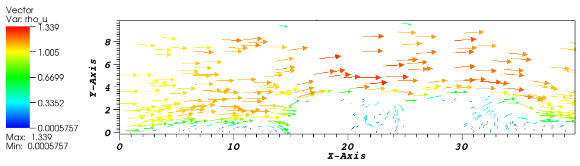
(c) Coarse mesh: t = 5.8s



(d) Refined mesh: t = 5.8s



(e) Coarse mesh: t = 8.7s



(f) Refined mesh: t = 8.7s

Figure 4.2: Comparison of velocity vectors between basic mesh and a refined mesh for a transient simulation.

However, initializing the flow in this manner avoids the transient period and transitions much faster to the steady state where the flow features are more obviously discernable and can be more easily compared, so for the purpose of mesh refinement comparison, this accelerated initialization makes sense. Looking at the maximum velocity in each plot, we notice that the maximum velocity in the different phases of the simulation remains within a few percentage points. For example, comparing the maximum velocities at $t = 2.9s$, we see that the coarse mesh presents $u_{max,coarse} = 1.26$ compared to the refined mesh maximum of $u_{max,refined} = 1.239$; the discrepancy is 1.67%. While the percentage difference is approximately 8% at the middle time interval, nearing the end of the simulation the discrepancy between the maximum velocities decreases further to 0.5%, which is encouragingly small. The Reynolds number of these flows was Re = 1600.

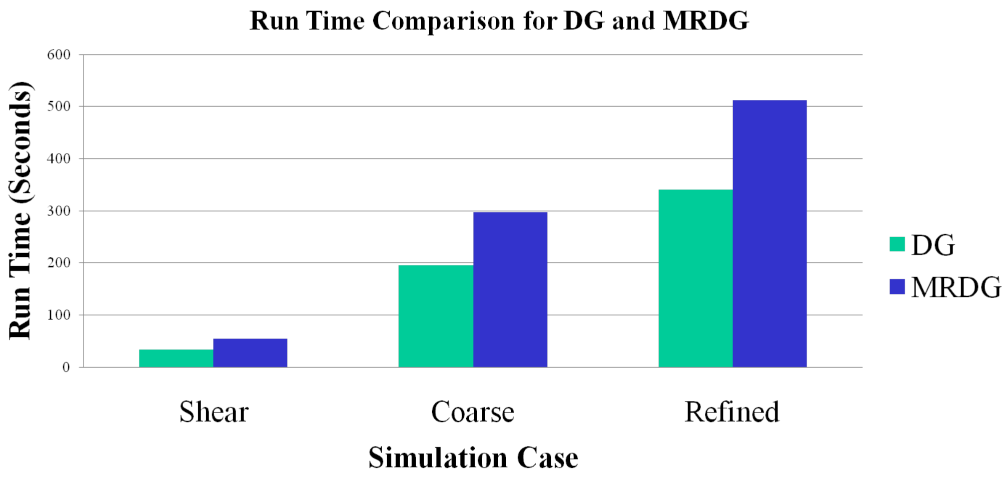


Figure 4.3: A comparison between the run times of the two different methods: DG and the MRDG method. The shear case refers to the laminar flow described in Section 4.2 shows only a small difference in run time, but for the more complicated channel bluff flow, the discrepancy is exacerbated.

4.1.1 Runtime and Accuracy Comparisons

A small sample size run time comparison was performed to examine the difference in runtime between the original DG method and the MRDG method. Looking at a comparison of the run time between the two methods, we see that there is a negligible difference for the very simple shear flow case discussed initially for verification purposes. However, when running the coarse mesh and the refined mesh, there is a more significant advantage for the DG method. While it is discouraging that the MRDG method's runtime is not superior, the run time is still on the same order of magnitude and there is definitely a lot of room for optimization of the code in order to improve the run time.

To expand on the optimization theme for the MRDG method, the *hedge* code is composed of files written in two programming languages: Python and C++. As with all programming languages, each of the two have their advantages and disadvantages. Python is a high-level language that relieves the programmer from dealing with the details of the actual execution of the code. Topics like memory addresses and allocation, garbage collection (discarding unused objects, like variables or instances of classes) and registers are left to the compiler to handle in exchange for a language that is more usable and readable. On the other hand, C++ is an intermediate-level language that grants access to efficiency-increasing methods at the exchange of user-facing complexity. For the existing *hedge* code, the bottlenecks in the Python code that would slow down the simulations have been rewritten in the faster C++ language. The result is a code that provides the user-friendly nature of Python with a small, fast core written in C++ to decrease the run-time. However, the adjustments I made to the program for the purposes of these simulations were all done in Python, as it is an easier language in which to code. Transferring the most frequently called functions and heavy mathematical operations from Python to C++ would have a definite effect on the MRDG run-time. In addition, there are further optimizations with Python such as memoization and unique imports that can be introduced to improve the run-time of the MRDG method. Moreover, there are a number of Python modules such as *hotshot*, *profile*, and *timeit* that help profile and time code. Being able to see where the the code runs slowest would allow one to achieve the greatest amount of optimization [19].

There also are a number of hardware specific available optimizations in the code that would make it a much more competitive code in terms of compute load and execution time. The Hedge code includes optional modules that allow for improvement of the code. The first of which is the boostmpi, which is a Message Passing Interface (MPI) wrapper for Python. Boostmpi provides access to high-performance communication between processes for parallel computing. In the absence of parallel computing architecture, one could also utilize the PyCUDA module with an Nvidia CUDA-enabled graphics card. CUDA, as previously mentioned, is Nvidia's foray into general-purpose computation on graphics processing units. The natural fit between GPUs and fluid flow simulations has been discussed previously, and the PyCUDA module would provide a number of distinct advantages, the foremost of which being an increase in speed with no affect on the accuracy of the method. Introducing these and other optional modules in hedge have been estimated to add at least additional 20%

speed increase [41].

4.2 Simple Steady State Flow

The next test case is a very simple flow that has an exact solution. Fully developed laminar pipe flow has an exact solution far from the entrance where the laminar flow has developed fully. Here, we can use the analytic solution as a benchmark for the MRDG method. For this case, there is an exact solution which can be used as accuracy verification:

$$\begin{aligned} u(x, y, t) =& y^2 \\ v(x, y, t) =& 0 \end{aligned}$$

The flow produced by the MRDG exhibits the expected features: a maximum velocity in the center of the channel tapering to zero at the walls where no-slip is enforced. The velocity profile for the solution to the Navier Stokes equation in this flow is shown in Figure 4.4, where what's shown is the bottom half of the pipe flow. The top of the graph shown would be a symmetry line and the plot could be mirrored above to create an entire picture of the laminar pipe flow. The L^2 error for the MRDG simulations was consistently and repeatably on the order of 10^{-5}, with a short run time of around thirty seconds. Solving the analogous problem using ANSYS® FLUENT grants a very similar plot shown in Figure 4.5, where again the symmetry of the system was used to simplify the equation. Conceptually mirroring the graph above creates the entire pipe and shows the parabolic flow profile. For a pipe diameter of $0.2m$ and a viscosity of $0.002\frac{kg}{m \cdot s}$, the Reynolds number for these cases was $Re = 100$. The images in Figure 4.4 and Figure 4.5 are of the same simulation and the similarity between them serves as a form of verification of the MRDG method.

4.3 Steady State Channel Flow

One of the test cases that was simulated was a long term simulation to see how the flow would develop. In the simulations with no time-dependent boundary conditions, steady state was achieved at approximately 50s, determined qualitatively by examining the flow contours

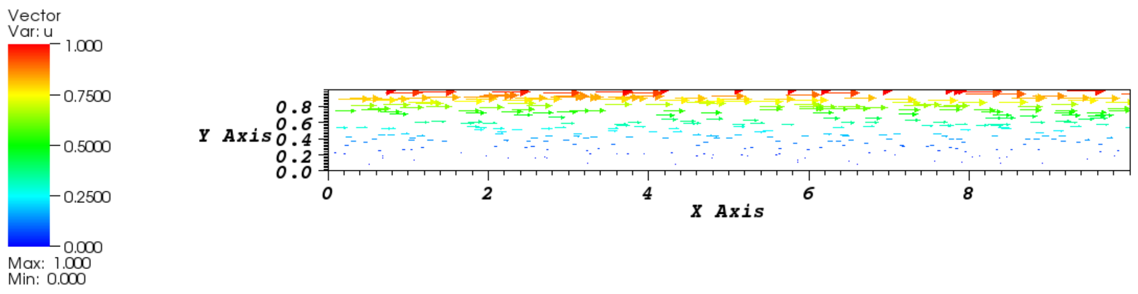


Figure 4.4: Vector plot of horizontal velocity; laminar flow modeled using the MRDG code. These results agree with [32].

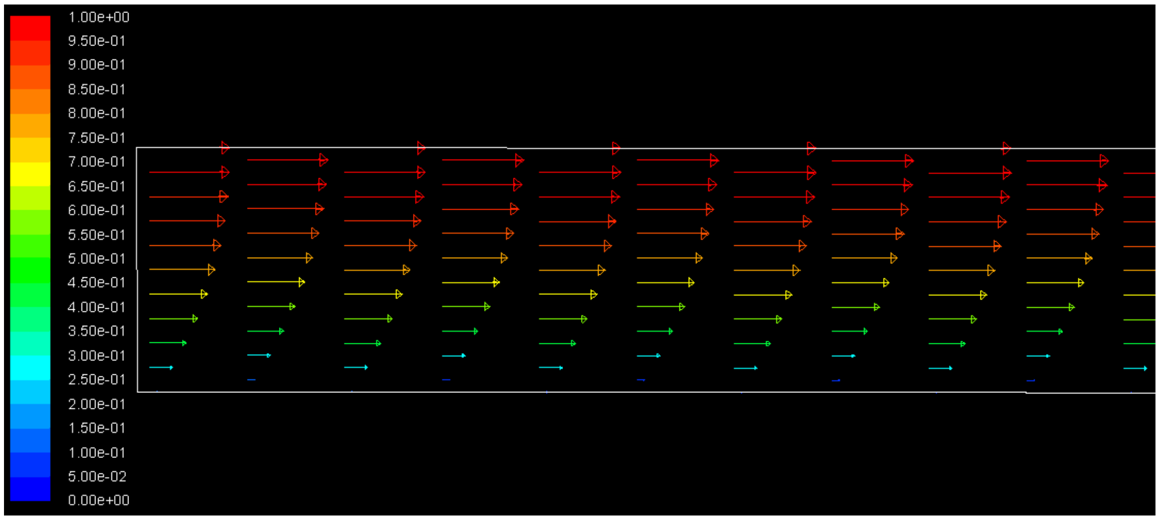
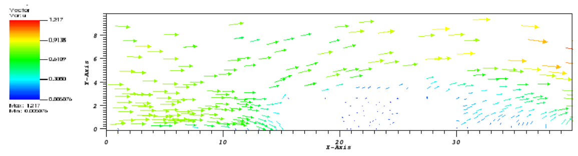


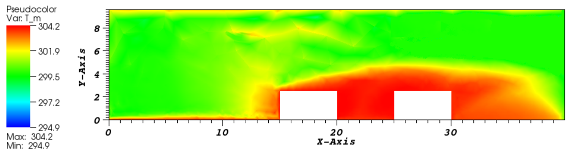
Figure 4.5: Vector plot of velocity in fully developed laminar pipe flow; laminar flow modeled using ANSYS® FLUENT code.

over time. Both the velocity and the temperature fields are provided for examination in Figure 4.6. Here, the comparison between a standard DG method and the enhanced MRDG method is exposed. While both methods perform at a similar computational speed, the MRDG method was able to resolve a higher degree of detail and information in the system. This is particularly evident in the temperature contours, which are much more detailed in the MRDG method. The initial conditions here were a reference temperature throughout the channel and the inlet flow also has the same reference temperature of 300 K. At $t = 0$, the bluffs were assigned a heat flux of $Q_1 = 50\frac{W}{cm^2}$ and $Q_2 = 75\frac{W}{cm^2}$. These fluxes are smaller than average for CPUs but higher than average for other components like RAM [13]. The velocity profile was zero throughout the channel, and at the inlet there was a uniform flow with a velocity of $1\frac{m}{s}$. These boundary conditions represent the setting where the server and the CRAC unit are both off for $t < 0$ and not generating heat or driving air through the server. At $t = 0$, both the server and the CRAC unit are turned on. A more realistic simulation would take into account the fact that the flow would not instantly begin to flow, and the heat from the computer components would not instantly emit the full heat flux load, but for a steady state simulation these details can be somewhat ignored as they have smaller influence on the long-time development of the flow.

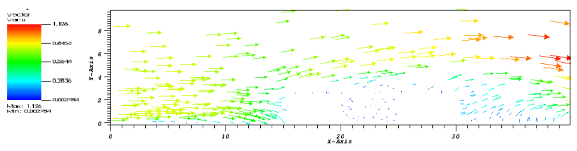
Looking at the thermal profiles of a long run that was allowed to reach steady state, we see that the results from the DG method are very comparable to the MRDG method. The profiles look similar and the maximum temperatures attained in each case are very close, to the point where the percent difference between them would be very small. As expected, you can see the flow carrying the heat away from the bluffs and also how the heat generated by the first bluff actually extends backwards along the flow due to the difficulty of flowing over that sharp corner. There is also some evidence of the lower quality of the DG solution when looking in the area above the first bluff. In the thermal profile produced by the DG, there are a few jagged edges in the profile that don't correspond to any physical phenomena in that region. In comparison, the MRDG thermal profile does not have these inaccuracies and produces a smooth profile that is more accurate.



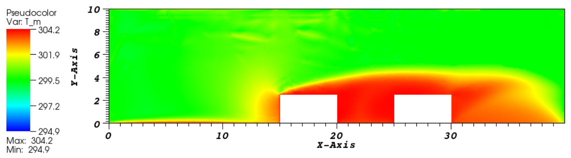
(a) DG Vector Plot



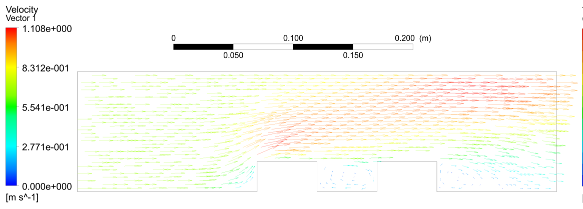
(b) DG Temperature Plot



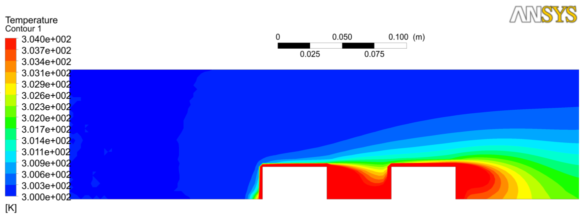
(c) MRDG Vector Plot



(d) MRDG Temperature Plot



(e) FLUENT Vector Plot



(f) FLUENT Temperature Plot

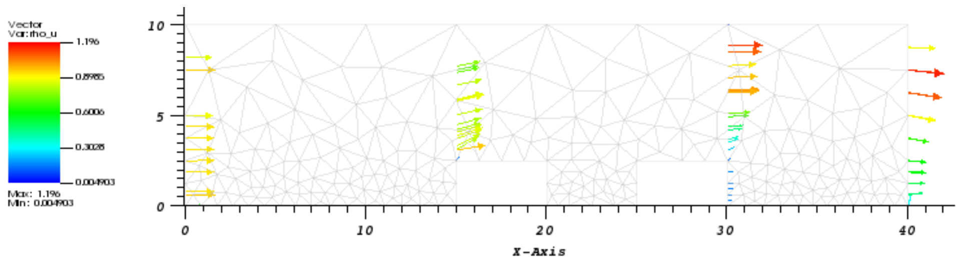
Figure 4.6: Steady state channel flow with bluffs after 50s of flow development; comparison between DG and MRDG shows an advantage for the MRDG method. The results generated using ANSYS ® FLUENT are shown in the bottom row.

The velocity profile and temperature profile for this case that was generated using ANSYS ® FLUENT are displayed in Figure 4.6. Although the legend color scheme is different for these plots, the scale is actually the same and the results from the MRDG are very comparable to those displayed in the figure. In order to achieve this steady state version in FLUENT, instead of letting the transient simulation run until steady state was achieved, the "steady state" option was simply used in the solution generation. In terms of comparisons between the three different simulations (DG, MRDG, and FLUENT), there are a few features that are notable. First, the FLUENT plot in Figure 4.6e most obviously shows that there is an area of increased velocity above the bluffs and leading towards the exit, caused by the uniform flow encountering the stagnation point at the sharp top left corner of the initial bluff. This increased velocity is somewhat missing from the DG simulation in Figure 4.6a but is much more apparent in the MRDG counterpart in Figure 4.6c. All three velocity simulations are qualitatively similar otherwise, with an initial slow region and little to no flow between the bluffs or in the bottom right region of the flow, where the bluffs would have blocked the majority of the velocity. Quantitatively, the DG method overestimated the maximum velocity by 9% as compared to the MRDG method and the FLUENT simulation: $1.21\frac{m}{s}$ (DG) vs $1.12\frac{m}{s}$ (MRDG) and $1.11\frac{m}{s}$ (FLUENT), lending weight to the accuracy of the MRDG method.

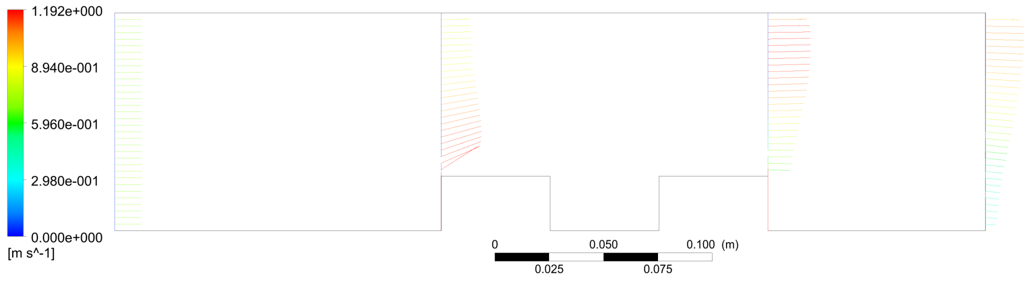
The thermal profiles show some discrepancy when comparing the three different methods. Both the DG and MRDG method seem to overestimate the influence of the heat flux compared to the FLUENT simulation. In Figure 4.6b and 4.6d, there is a region of high temperature that is carried significantly far away from the bluffs to form a cloud surrounding the bottom half of the channel. In comparison, Figure 4.6f has a thermal profile that indicates that the highest temperature air, the red region, does not travel so freely through the channel, and in fact the high temperature region is smaller and more condensed than as prediced by the MRDG methods. While the FLUENT profile does contain a "cloud" of higher temperature being pulled away from the bluffs, it is at a much lower temperature than seen in the previous Figures. From a practical standpoint, erring on the side of predicting a hotter temperature is favorable in the sense that one would rather prefer to over-cool the servers and assuredly avoid heat damage; if the simulation predicted temperatures that were too low, one might inadvertently provide too little cooling and risk damage to the servers. That being said, the deficiency in the accuracy of the thermal profile is definitely a priority for further improvement. Quantitatively, the maximum temperatures in all three methods are actually very close, around 304 K in all of the different cases.

4.3.1 Cross Section Comparisons

In order to elucidate the comparisons between the MRDG method and the FLUENT simulations, comparisons at certain cross sections through the channel flow will be displayed. Both the velocity vectors and the thermal contours are shown in Figure 4.7 and Figure 4.8. The cross sections are taken at the entrance of the flow, at $x = 0$, at the left corner of the first bluff $x = 0.15$, the right corner of the second bluff at $x = 0.25$, and at the exit of the channel at $x = 0.40$. Comparing the velocity profiles at the different cross sections, we see that they match up very well. The initial cross section, taken at the entrance of the channel,



(a) MRDG Vector Cross Sections

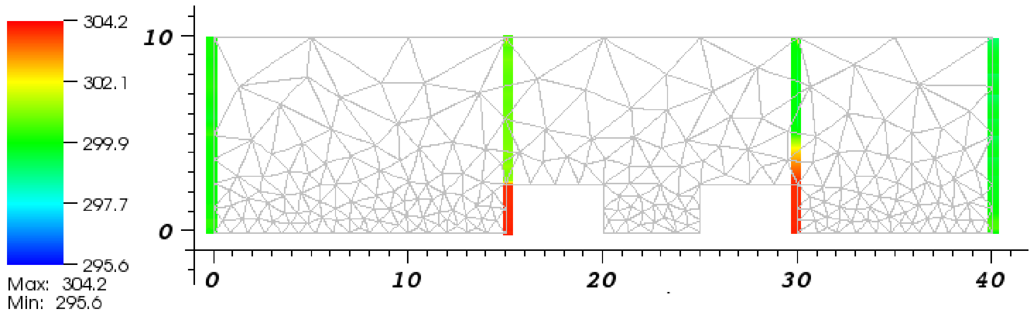


(b) FLUENT Vector Cross Sections

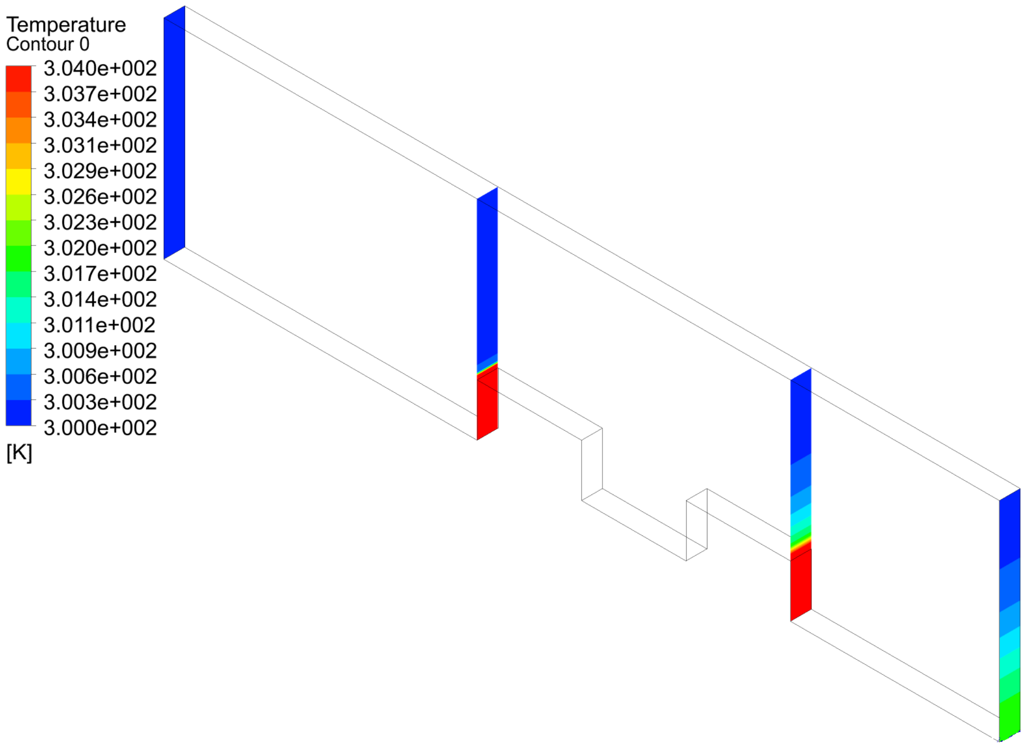
Figure 4.7: Comparisons of the velocity profiles and temperature contours taken at different cross-sections throughout the channel. The simulations shown are produced by the MRDG simulation as well as FLUENT for simulation verification.

is similar in both versions: uniform flow. At the corner of the first bluff, we again see a similar profile. Starting at the bottom of the profile and moving to to the top, we see that at the bottom the majority of the velocity vectors have a slight positive y-axis component, with the x-axis component being strongest near the bluff and dying off closer to the top of the channel. At the third cross section at the end of the second bluff, the profiles are again similar. Near the bottom of the channel there is little to no flow due to the back of the bluff being a poor area for velocity development, and then moving higher and higher up the channel, we see a steady increase in velocity, the shape of which is mirrored in both Figure 4.7a and Figure 4.7b. Finally, at the exit of the channel, near the bottom area the flow velocity is low in both simulations, and the profiles mirror each other again. Moving up towards the top of the channel, we see a velocity increase before reaching the very top where there is a slight decrease in velocity. Again, these profiles are very similar in shape and magnitude.

Considering the temperature profiles, we see that the MRDG simulation is very well



(a) MRDG Thermal Contour Cross Sections

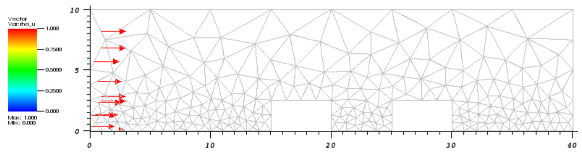


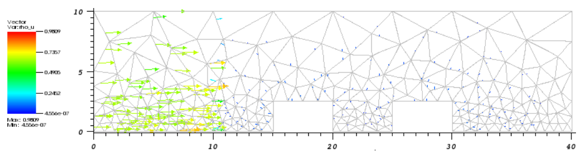
(b) FLUENT Thermal Contour Cross Sections

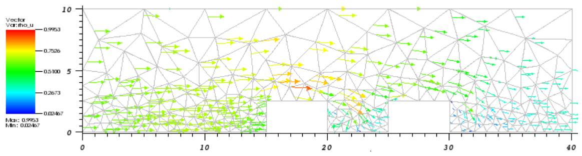
Figure 4.8: Comparisons of the temperature contours taken at different cross-sections throughout the channel.

matched to the FLUENT simulation. Note that in Figure 4.8a, the "cross section" has been widened for ease of viewing - the data for each location has simply been stretched over a wider x-range to make it easier to see, but each "bar" in the graph only contains data from a single x location, not an interval as the graph may seem to depict. Looking at the cross sections in a similar fashion as before (left to right, bottom to top), starting with the entrance cross section we see a similar base temperature around 300 K with not much deviation. At the left corner of the first bluff, of course the bluff itself has a higher temperature, shown in red in both graphs, but the rest of the channel is essentially the same temperature as the entrance; both the MRDG and the FLUENT simulations show that the top half of the channel is not receiving much influence from the heated bluff below. Considering the fact that the velocity does not have a strong vertical component in this region, it makes sense that the upper half of the channel is not yet affected by the heated bluff below.

At the third cross section located at the right corner of the second bluff, we see a bit of a discrepancy in scale, although the general profile is similar. At this point in the channel, both simulations depict the heated section to be taller than the bluffs themselves, moving a bit above the bluff and heating part of the channel. However, the MRDG simulation overestimates the influence of the heated bluffs as compared to the FLUENT simulation, where the heated region stays closer to the bluffs themselves. For the upper half of the channel, the profiles are very similar; this discrepancy is only in a small part of the channel and is exacerbated by the fact that the legends are colored differently. Note that in the FLUENT simulation, there is a region of yellow, green, and light blue in this area: a range of 302 K to 300 K. While the MRDG simulation has a range of orange, yellow, and green, looking at the legend we see that these colors actually correspond to a range of 303 to 300 K. While the colors seem to be significantly different, the actual numerical discrepancy is relatively small. Finally, at the last cross section taken at the exit of the channel, the MRDG simulation predicts a similar profile as the entrance of the reference temperature. In the FLUENT simulation, the bottom portion of the channel is slightly warmer, but the majority of the exit profile is also at the reference temperature.


(a) The initial condition is uniform inlet flow at t=1 s


(b) The flow is mainly in the first third of the channel at t = 5s


(c) The flow has pushed across the majority of the channel at t = 10s

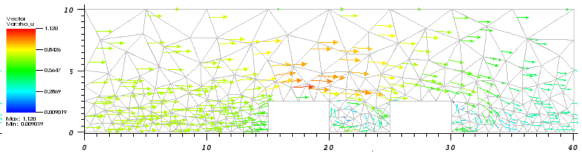

(d) The inlet uniform flow has propagated across the majority of the channel at t = 13s

Figure 4.9: Transient channel flow with a different starting condition.

4.4 Transient Channel Flow

An area that is currently in development is a transient version of the MRDG that incorporates a different initial condition. When the server is turned on and the CRAC units are activated in the data center, the initial condition would be no velocity in the channel and a uniform velocity at the inlet that propagates through the channel. Each of the displays is a time shifted simulation of the flow with the mesh in the background for clarity purposes.

Looking at the progression of the flow over time, its possible to see how the inlet uniform velocity forces its way through the channel. Initially, as mentioned, the velocity profile is $u(x, y) = 0$ throughout the channel and a uniform velocity profile at the left inlet. Figure 4.9a shows these condition with a few moments after $t = 0$ and shows that the flow has moved into the channel. In Figure 4.9b we see the flow has begun to proceed through the initial third of the channel, spreading out and covering more and more of the channel. In Figure 4.9c, the flow has moved across the majority of the channel, and a few key features have started to appear that are seen in the steady state simulation. First, above the first bluff, there is a region of higher velocity that can be seen in Figure 4.6 as well in both the MRDG and FLUENT simulations. However, unlike the steady state simulations, the flow at this location above the first bluff is still directed somewhat downwards, whereas in the previous simulations, the flow was directed up and away from the bluffs. Also, vortices have begun to form between the bluffs and after the second bluff, another feature that is echoed in the steady state simulations. Finally, in Figure 4.9d, we see the resolution of some of the discrepancies from Figure 4.9c. The vortices are more fully developed and apparent, and the flow above the first bluff is beginning to move more upwards, to match up with what we would expect to see when connecting the end of the transient flow to the steady state simulations.

CHAPTER V

CONCLUSION

We have presented a solution method for unsteady incompressible flows and associated heat transfer that directly addresses the tradeoff between speed and accuracy. The efficient function representation inherent to the multi-resolution approximation transfers to the composite MRDG method wherein the solution drives the adaptation and compression. The DG method serves as a high order computational tool with enough flexibility to take advantage of the multi-resolution framework. While the method presented here is still in its infancy, the literature has proven its effectiveness and a first effort at implementing the method returned results that agreed with previous efforts [2, 63, 41].

5.1 Future Work

Future work includes applying the MRDG method to increasingly realistic model problems, incorporating more aspects of an actual server into the model as well as considering entire racks and multi-rack domains in the simulation. Extending the method to three dimensions is theoretically straightforward and would be the ultimate goal, but even in the current two dimensional implementation, there are a number of optimizations that can still improve the code.

The sample problems that were described here are very limited in terms of their real-world applicability, not to mention that they are a far cry from an entire data center simulation. Introducing more realistic boundary conditions for the simple 2D example could be the first step in improving the model. Making the heat flux a funcion of the distance from the channel inlet could represent the introduction of hotspots on the server components. In addition, having heat flux come from the sides of the bluff would also be a closer representation of real heat generating components, instead of assuming that the heat flux only comes from one face. It may also be relevant to consider the heat flux from the components of the server in the rack above or below. Changing the actual boundary itself to more accurately reflect the server geometries would also be of use. Incorporating the internal fans of the server, for example, would definitely have an impact on the flow, as does the presence of various wires and other internal components.

5.1.1 Extension to 3-D

After handling the various details that are critical to an accurate flow representation in 2D, one of the next goals after having presented the usefulness of the method in 2D is obviously the extension to 3D, which would provide the capability for more realistic, and thus more useful, simulations and flow predictions. In the current *hedge* framework, there are already built-in functions that produce three dimensional meshes, so from a meshing standpoint, it would simply be a matter of describing the geometry of the model and passing that to the discretization code within *hedge*. Implementing a grid-adaptive multi-wavelet basis would be a very involved process, although some of the framework for such an extension exists [63]. From an operator standpoint within *hedge*, it would also be necessary to verify that none of the simplifications to the governing equations relied on any 2D specific details, and implementing any changes that arose as a result.

5.1.2 Method Applicability

Many of the advantages discussed here have been geared directly towards the solution of a distinct problem: turbulent fluid dynamics and heat transfer. In particular, certain features of turbulent flow such as its intermittent nature and self-organization into coherent structures have been targetted as areas where classical methods were not taking advantage of efficient representations. When considering the simulation of laminar flow, obviously some of these flow features for which the MRDG was designed will not be present. However, the strengths of the hybrid method come through here, as the high order, accurate nature of the DG method still applies in a laminar setting. While some of the multiresolution advantages might not be present in a laminar simulation, the DG method is still able to provide high order results for laminar simulations, as is evident in literature about DG methods being used for laminar flows [28, 47, 39, 43]. Because the design of the method was intended for turbulent flow, the MRDG method may not be as feasible for laminar flow as a method that was specifically designed for laminar flow. However, the MRDG method should be expected to perform reasonably well in a laminar situation because of the inclusion of the DG method. The MRDG would be most advantageous in a multi-scale turbulent problem because of the nature of the turbulent decomposition is much more efficient in a multiresolution decomposition than a classic polynomial decomposition. Also, for a problem like an entire data center simulation where the domain is complex and the length scale is across several orders of magnitude, running the MRDG method on an a parallel infrastructure would be a significant improvement over other methods. In an effort to increase the robustness of the code, future work should also include work in the line of introducing laminar-specific optimizations where possible so that the code is able to better encompass a wider range of flow environments.

Again, the overarching goal is to be able to produce a method that allows for real-time simulation of data centers. Simulating each server is just the first step in that goal, and so future work would obviously include the simulation of data centers. Initially, while developing the simulation, an idealized version of a data center would be used - the canonical example of a CRAC unit and two rows of four server racks would be the starting point. Verifying that the simulation works and moving towards more complicated versions of the canonical example would be the next step. Combining the different models wherein the servers within each rack are modeled and their contributions are taken into account in the overall room simulation would be the ultimate goal, at which point the goal would be to reduce the simulation's computational complexity in order to reduce the execution time. By controlling those parameters, it would be possible to simulate the entire data center and be able to predict to problems and react to them to preserve the integrity of the data center.

REFERENCES

[1] Alam, J. M., Kevlahan, N. K. R., and Vasilyev, O. V., "Simultaneous space-time adaptive wavelet solution of nonlinear parabolic differential equations," *J. Comput. Phys.*, vol. 214, pp. 829–857, May 2006.

[2] Alpert, B., Beylkin, G., Gines, D., and Vozovoi, L., "Adaptive solution of partial differential equations in multiwavelet bases," *Journal of Computational Physics*, vol. 182, no. 1, pp. 149 – 190, 2002.

[3] Archibald, R., Evans, K. J., Drake, J., and White, J. B., "Multiwavelet discontinuous galerkin-accelerated exact linear part (elp) method for the shallow-water equations on the cubed sphere," *Monthly Weather Review*, vol. 139, pp. 457 – 473, 2010.

[4] Bacry, E., Mallat, S., and Papanicolaou, G., "A wavelet based space-time adaptive numerical method for partial differential equations," *Mathematical Modelling and Numerical Analysis*, vol. 26, pp. 793–834, 1992.

[5] Basdevant, C., Holschneider, M., and Perrier, V., "Traveling wavelets method.," *C. R. Acad. Sci. Ser. I Math.*, vol. 310, pp. 647–52, 1990.

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

[7] Baumann, C. E. and Oden, J. T., "A discontinuous hp finite element method for convection–diffusion problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 175, no. 3-4, pp. 311 – 341, 1999.

[8] Belady, C. L., "In the data center, power and cooling costs more than the it equipment it supports," *Electronics Cooling*, vol. 13, February 2007.

[9] Berg, J., *Wavelets in physics.* Cambridge University Press, 2004.

[10] Bertoluzza, S. and Naldi, G., "A wavelet collocation method for the numerical solution of partial differential equations," *Applied and Computational Harmonic Analysis*, vol. 3, no. 1, pp. 1 – 9, 1996.

[11] Beylkin, G., Cheruvu, V., and Prez, F., "Fast adaptive algorithms in the non-standard form for multidimensional problems," *Applied and Computational Harmonic Analysis*, vol. 24, no. 3, pp. 354 – 377, 2008.

[12] Bindal, A., Khinast, J. G., and Ierapetritou, M. G., "Adaptive multiscale solution of dynamical systems in chemical processes using wavelets," *Computers and Chemical Engineering*, vol. 27, pp. 131–142, 2003.

[13] Bowers, M. B. and Mudawar, I., “High flux boiling in low flow rate, low pressure drop mini-channel and micro-channel heat sinks,” *International Journal of Heat and Mass Transfer*, vol. 37, pp. 321–332, Jan. 1994.

[14] Cockburn, B. and Shu, C.-W., “Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws ii: General framework,” *Mathematics of Computation*, vol. 52, no. 186, pp. 411–435, 1989.

[15] Cockburn, B. and Shu, C.-W., “The local discontinuous galerkin method for time-dependent convection-diffusion systems,” *SIAM J. Numer. Anal.*, vol. 35, no. 6, pp. 2440–2463, 1998.

[16] Cohen, A., Kaber, S. M., Müller, S., and Postel, M., “Fully adaptive multiresolution finite volume schemes for conservation laws,” *Math. Comput.*, vol. 72, pp. 183–225, January 2003.

[17] Collis, S. S., “Discontinuous galerkin methods for turbulence simulation,” in *In Proceedings of the 2002 Center for Turbulence Research Summer Program*, pp. 155–167, 2002.

[18] Coult, N., “Introduction to discontinuous wavelets,” in *Discontinuous Galerkin methods: theory, computation, and applications* (Cockburn, B., Karniadakis, G., and Shu, C., eds.), Lecture notes in computational science and engineering, ch. 2, pp. 301–308, Springer, 2000.

[19] Documentation, P., “Python speed,” August 2011.

[20] Domingues, M. O., Gomes, S. M., Roussel, O., and Schneider, K., “An adaptive multiresolution scheme with local time stepping for evolutionary pdes,” *J. Comput. Phys.*, vol. 227, pp. 3758–3780, April 2008.

[21] Domingues, M. O., Gomes, S. M., Roussel, O., and Schneider, K., “Space–time adaptive multiresolution methods for hyperbolic conservation laws: Applications to compressible euler equations,” *Appl. Numer. Math.*, vol. 59, pp. 2303–2321, September 2009.

[22] Durbin, P. A. and Pettersson-Reif, B. A., “Statistical theory and modeling for turbulent flows,” *Wiley & Sons, Ltd. 2nd ed.*, Jan. 2011.

[23] F., F. B. 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, no. 2, pp. 267–279, 1997.

[24] Farge, M., Goirand, E., Meyer, Y., Pascal, F., and Wickerhauser, M. V., “Improved predictability of two-dimensional turbulent flows using wavelet packet compression,” *Fluid Dynamics Research*, vol. 10, pp. 229–250, 1992.

[25] Farge, M., Kevlahan, N., Perrier, V., and Éric Goirand, “Wavelets and turbulence,” *Proceedings of the IEEE*, vol. 84, pp. 639–669, April 1996.

[26] Farge, M. and Schneider, K., “Coherent vortex simulation (cvs), a semi-deterministic turbulence model using wavelets,” *Flow, Turbulence and Combustion*, vol. 66, pp. 393–426, 2001.

[27] Farge, M., Schneider, K., and Kevlahan, N., "Non-gaussianity and coherent vortex simulation for two-dimensional turbulence using an adaptive orthogonal wavelet basis," *Physics of Fluids*, vol. 11, pp. 2187 – 2201, August 1999.

[28] Ferrer, E. and Willden, R., "A high order discontinuous galerkin finite element solver for the incompressible navier-stokes equations," *Computers & Fluids*, vol. 46, no. 1, pp. 224 – 230, 2011. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[29] Goldstein, D. E., *Stochastic Coherent Adaptive Large Eddy Simulation Method.* Ph.D. thesis, University of Colorado, 2004.

[30] Group, S. V. L., "Data center energy forecast: Final report.." White paper, June 2008.

[31] Harten, A., "Adaptive multiresolution schemes for shock computations," *J. Comput. Phys.*, vol. 115, pp. 319–338, December 1994.

[32] Hesthaven, J. S. and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications.* Springer Publishing Company, Incorporated, 2007.

[33] Hewlett-Packard, "Hp proliant bl2x220c g5 server blade." Data Sheet, April 2008.

[34] Holmström, M. and Waldén, J., "Adaptive wavelet methods for hyperbolic pdes," *J. Sci. Comput.*, vol. 13, pp. 19–49, March 1998.

[35] Jansen, M., "Wavelets and wavelet thresholding," in *Noise Reduction by Wavelet Thresholding*, vol. 161 of *Lecture Notes in Statistics*, pp. 9–45, Springer New York, 2001. 10.1007/978-1-4613-0145-5_2.

[36] Karniadakis, G. and Sherwin, S. J., *Spectral/hp element methods for CFD.* Oxford University Press, illustrated ed., 1999.

[37] Kevlahan, N. K.-R. and Vasilyev, O. V., "An adaptive wavelet collocation method for fluid-structure interaction at high reynolds numbers," *SIAM J. Sci. Comput.*, vol. 26, pp. 1894–1915, June 2005.

[38] Kirby, R. M. and Karniadakis, G. E., "Selecting the numerical flux in discontinuous galerkin methods for diffusion problems," *J. Sci. Comput.*, vol. 22-23, pp. 385–411, June 2005.

[39] Klaij, C., van der Vegt, J., and van der Ven, H., "Space-time discontinuous galerkin method for the compressible navier-stokes equations," *Journal of Computational Physics*, vol. 217, no. 2, pp. 589 – 611, 2006.

[40] Klockner, A., "Hedge: Hybrid 'n' easy discontinuous galerkin environment," 2009.

[41] Klockner, A., *High-Performance High-Order Simulation of Wave and Plasma Phenomena.* PhD thesis, Brown University, May 2010.

[42] Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., and Fasih, A., "PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation," Tech. Rep. 2009-40, Scientific Computing Group, Brown University, Providence, RI, USA, Nov. 2009.

[43] Landmann, B., Kessler, M., Wagner, S., and Krmer, E., "A parallel, high-order discontinuous galerkin code for laminar and turbulent flows," *Computers & Fluids*, vol. 37, no. 4, pp. 427 – 438, 2008. Turbulent Flow and Noise Generation.

[44] Laney, C. B., *Computational gasdynamics*. Cambridge University Press, 1998.

[45] LeVeque, R. J., "Python tools for reproducible research on hyperbolic problems," *Computing in Science and Engineering*, vol. 11, pp. 19–27, 2009.

[46] Li, B. Q., *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Computational Fluid and Solid Mechanics, Springer, 2006.

[47] Liu, H. and Xu, K., "A runge-kutta discontinuous galerkin method for viscous flow equations," *Journal of Computational Physics*, vol. 224, no. 2, pp. 1223 – 1242, 2007.

[48] Liu, Y., Cameron, I. T., and Wang, F. Y., "The wavelet-collocation method for transient problems with steep gradients," *Chemical Engineering Science*, vol. 55, pp. 1729 – 1734, 2000.

[49] Lomtev, I. and Karniadakis, G. E., "A discontinuous galerkin method for the navier-stokes equations," *Int. J. Numer. Meth. Fluids*, vol. 29, pp. 587–603, 1999.

[50] Luo, H., Baum, J. D., and Löhner, R., "A discontinuous galerkin method based on a taylor basis for the compressible flows on arbitrary grids," *Journal of Computational Physics*, vol. 227, pp. 8875–8893, 2008.

[51] Mallat, S. G., "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, 1989.

[52] Müller, S. and Stiriba, Y., "Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping," *J. Sci. Comput.*, vol. 30, pp. 493–531, March 2007.

[53] Myers, M., Holmes, P., Elezgaray, J., and Berkooz, G., "Wavelet projections of the kuramoto-sivashinsky equation i. heteroclinic cycles and modulated traveling waves for short systems," *Phys. D*, vol. 86, pp. 396–427, September 1995.

[54] Niemann, J., "Hot aisle vs. cold aisle containment." White paper, 2008.

[55] Nvidia, "Nvidia cuda reference manual," 2007.

[56] Okamoto, N., Yoshimatsu, K., Schneider, K., Farge, M., and Kaneda, Y., "Coherent vortex simulation: application to 3d homogeneous isotropic turbulence," in *Advances in Turbulence XII* (Eckhardt, B., ed.), vol. 132 of *Springer Proceedings in Physics*, pp. 759–762, Springer Berlin Heidelberg, 2009.

[57] Rastigejev, Y. A. and Paolucci, S., "Wavelet-based adaptive multiresolution computation of viscous reactive flows," *International Journal for Numerical Methods in Fluids*, vol. 52, no. 7, pp. 749–784, 2006.

[58] Ren, X. and Xanthis, L. S., "[']les fleurs du mal' ii: A dynamically adaptive wavelet method of arbitrary lines for nonlinear evolutionary problems–capturing steep moving fronts," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 37-40, pp. 4962 – 4970, 2006. John H. Argyris Memorial Issue. Part I.

[59] Rivière, B., *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation.* Frontiers in Applied Mathematics, Philadelphia, PA: SIAM, 1st ed., 2008.

[60] Schneider, K., Farge, M., Pellegrino, G., and Rogers, M. M., "Coherent vortex simulation of three-dimensional turbulent mixing layers using orthogonal wavelets," *Journal of Fluid Mechanics*, vol. 534, pp. 39–66, July 2005.

[61] Schneider, K. and Farge, M., "Adaptive wavelet simulation of a flow around an impulsively started cylinder using penalisation," *Applied and Computational Harmonic Analysis*, vol. 12, no. 3, pp. 374 – 380, 2002.

[62] Schneider, K. and Vasilyev, O. V., "Wavelet Methods in Computational Fluid Dynamics," *ANNUAL REVIEW OF FLUID MECHANICS*, vol. 42, pp. 473–503, 2010.

[63] Shelton, A., *A Multi-Resolution Discontinuous Galerkin Method for Unsteady Compressible Flows.* PhD thesis, Georgia Institute of Technology, August 2008.

[64] Somani, A., "Advanced thermal management strategies for energy efficient data centers," Master's thesis, Georgia Institute of Technology, December 2008.

[65] Stan, F., "Discontinuous galerkin method for interface crack propagation," *Int. J. Mater. Form.*, vol. Suppl 1, pp. 1127–1130, 2008.

[66] Stefano, G. D., Vasilyev, O. V., and Goldstein, D. E., "Localized dynamic kinetic-energy-based models for stochastic coherent adaptive large eddy simulation," *Physics of Fluids*, vol. 20, no. 4, p. 045102, 2008.

[67] van Rossum, G., "Python reference manual," 1995.

[68] Vasilyev, O. V., Stefano, G. D., Goldstein, D. E., and Kevlahan, N. K. R., "Lagrangian dynamic sgs model for stochastic coherent adaptive large eddy simulation," *Journal of Turbulence*, 2008.

[69] Warburton, T., "An explicit construction of interpolation nodes on the simplex," *Journal of Engineering Mathematics*, vol. 56, pp. 247–262, 2006. 10.1007/s10665-006-9086-6.

[70] Wirasaet, D. and Paolucci, S., "Application of an adaptive wavelet method to natural-convection flow in a differentially heated cavity," *ASME Conference Proceedings*, vol. 2005, no. 47330, pp. 499–511, 2005.

[71] Yuan, L. and Shu, C.-W., "Discontinuous galerkin method based on non-polynomial approximation spaces," *J. Comput. Phys.*, vol. 218, no. 1, pp. 295–323, 2006.

[72] Zhang, M. and Shu, C.-W., “An analysis of three different formulations of the discontinuous galerkin method for diffusion equations,” *Mathematical Models and Methods in Applied Sciences*, vol. 13, pp. 395–413, 2002.