Identification, Decomposition and Analysis of Dynamic Large-Scale Structures in Turbulent Rayleigh-Bénard Convection

by

Philip John Sakievich

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2017 by the
Graduate Supervisory Committee:

Yulia Peet, Chair
Ronald Adrian
Kyle Squires
Marcus Herrmann
Eric Kostelich

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

The central purpose of this work is to investigate the large-scale, coherent structures that exist in turbulent Rayleigh-Bénard convection (RBC) when the domain is large enough for the classical "wind of turbulence" to break down. The study exclusively focuses on the structures that from when the RBC geometry is a cylinder. A series of visualization studies, Fourier analysis and proper orthogonal decomposition are employed to qualitatively and quantitatively inspect the large-scale structures' length and time scales, spatial organization, and dynamic properties. The data in this study is generated by direct numerical simulation to resolve all the scales of turbulence in a 6.3 aspect-ratio cylinder at a Rayleigh number of 9.6×10^7 and Prandtl number of 6.7. Single and double point statistics are compared against experiments and several resolution criteria are examined to verify that the simulation has enough spatial and temporal resolution to adequately represent the physical system.

Large-scale structures are found to organize as roll-cells aligned along the cell's side walls, with rays of vorticity pointing toward the core of the cell. Two different large-scale organizations are observed and these patterns are well described spatially and energetically by azimuthal Fourier modes with frequencies of 2 and 3. These Fourier modes are shown to be dominant throughout the entire domain, and are found to be the primary source for radial inhomogeneity by inspection of the energy spectra. The precision with which the azimuthal Fourier modes describe these large-scale structures shows that these structures influence a large range of length scales. Conversely, the smaller scale structures are found to be more sensitive to radial position within the Fourier modes showing a strong dependence on physical length scales.

Dynamics in the large-scale structures are observed including a transition in the global pattern followed by a net rotation about the central axis. The transition takes

place over 10 eddy-turnover times and the subsequent rotation occurs at a rate of approximately 1.1 degrees per eddy-turnover. These time-scales are of the same order of magnitude as those seen in lower aspect-ratio RBC for similar events and suggests a similarity in dynamic events across different aspect-ratios.

DEDICATION

In memory of Jim Miner.

Ever the engineer and forever a friend.

ACKNOWLEDGMENTS

First and foremost, I wish to acknowledge my parents Mark and Gina Sakievich. With out your love, patience, support and guidance I would never have come close to where I am today. All children owe their parents a great debt for giving them life, but I owe you one greater for the quality of life you have bestowed upon me. Next, I wish to acknowledge and thank my loving wife, Annie, for her patience and support over the past four years. You have been with me through every up and down during this process, and my love for you has grown exponentially in this time period.

I would also like to acknowledge my advisors and academic mentors Dr. Yulia Peet and Dr. Ronald Adrain. You have both been incredibly kind, patient and supportive of me as I have worked to understand the nuances of thermal convection and turbulence. I have enjoyed working with both of you, and I appreciate the way you have helped me to develop as a researcher. Dr. Peet thank you for seeing the potential in me and offering me a research position at the end of my master's degree. Dr. Adrian, thank you for your generosity to me and my growing family. Your wisdom, mentorship and kindness have been greatly appreciated. I have enjoyed our many long discussions. Under you're tutelage I have come to appreciate the value of declaring precise mathematical definitions and expressing equations in their continuous form. I have also learned an incredible amount from you in arenas that extend well beyond the realm of academia. Your boundless optimism and kindness have provided lessons I will never forget.

My other committee members have also played a large roll in reaching this position. Dr. Herrmann, your exceptional courses inspired me to pursue CFD and taught me to love the power and beauty of numerical methods. Dr. Squires, you have been a part of my academic career since my very first semester at ASU over 9 years ago, and I have

always found your instruction and feed back to be concise, well posed and incredibly insightful. Dr. Kostelich your course introduced me to one of my favorites subjects, high-performance computing, and you have been a continual resource and support to me during my research. I am truly fortunate to have such an amazing committee where every member has played a foundational roll in shaping my academic course and bringing me to my current position.

I would also like to acknowledge the amazing graduate students that I have been privileged to work with over the years. Brandon Merrill, YiQin Xu, Daniel Coxe and my other colleagues in the Integrative Simulations and Computational Fluids Lab, you have been great colleagues, and I have enjoyed working with everyone of you. I would like to give a special thanks to Tanmoy Chatterjee for the long and engaging conversations, and your keen insight into turbulence. You have a brilliant mind and I am certain you will continue to do great work in the future. I am also thankful for my colleagues in the Laboratory for Energetic Flow and Turbulence: Liuyang, Heather and Rafeed. You have been great friends, and I have been inspired by your work.

I also wish to recognize the many other friends and family members who have supported and encouraged me over the years. There are too many people to thank individually, but I acknowledge that a large portion of my success is due to the support system and positive influences in my life. In the coming years I hope to pay forward the kindness and consideration that have been shown to me by so many individuals.

Finally, I would like to acknowledge U.S. National Science Foundation Grants CBET-1335731 and CMMI-1250124, XSEDE allocations TG-ENG140002 and TG-CTS150039, and the Arizona State University Dean's Fellowship (2013/2017-MAE-105) for supporting my work.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

This is a numerical study of turbulent Rayleigh-Bénard convection (RBC) in cylinders when the aspect-ratio is large enough to permit several persistent, three-dimensional roll-cells. RBC studies are most commonly conducted in rectangular and cylindrical domains, but this study will exclusively focus on the structures that form in cylindrical domains. The emphasis of this work is on understanding the spatial and temporal properties of these large-scale of motions within the flow field. In many ways this is a companion study to the experimental work conducted by Richard Fernandes [22]. The boundary conditions, spatial domain and physical properties of the flow field in this work are set to mirror his experiments. Utilizing fully resolved numerical simulations in this study provides additional insight into the spatial structure of the turbulent flow field that Fernandes was not able to capture with experimental techniques. This chapter introduces the high-level research questions that are guiding this work, provides background information on the physics, and describes the organization for the rest of the document.

1.1 Research Questions

Thermal convection, which is often referred to as natural convection, plays an important role in atmospheric science, astrophysics and several engineering applications. Thermal convection occurs when unstable thermal stratification induces fluid flow through buoyancy forces. When the forces generated by buoyancy are smaller than

the diffusive forces all heat is transported through the fluid via molecular diffusion. However, once the buoyant forces become sufficient enough for a flow to be generated the rate of heat transfer increase dramatically.

At the onset of convection the flow is laminar, but a transition to turbulence occurs as the strength of the internal buoyancy forces grow. Several different forms of thermal convection exist [2], but the most popular form to study is Rayleigh-Bénard convection (RBC). Turbulent RBC is considered an ideal problem for investigating the complex phenomenon of turbulent thermal convection because the simple boundary conditions make it manageable to study experimentally and numerically without sacrificing the core complexity of thermal convection. An illustration of the domain and boundary conditions for a standard RBC cell are provided in figure 1.

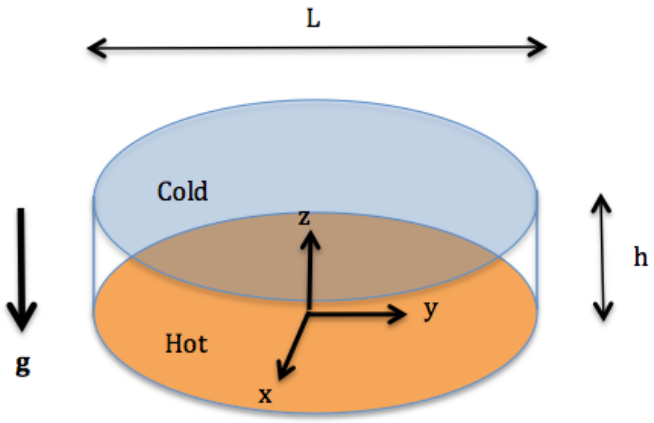


Figure 1: Illustration of the domain and boundary conditions that are associated with Rayleigh-Bénard convection

The production of turbulence by buoyancy in RBC is one of the more easily conceptualized mechanisms among all of the canonical flows. However, the analysis of turbulent RBC is still very complicated, and many of the tools that are used to study the canonical shear flows do not fully cross over to the study of RBC. For example,

canonical shear flows such as pipe flow, channel flow and flow over a flat plate can be analyzed with only one inhomogeneous direction. This reduces the complexity of analysis significantly because the turbulent fluctuations and mean flow can be analyzed with respect to only one spatial variable.

Ideally, turbulent RBC should only have a single inhomogeneous direction as well. In this case the convection cell would have a fixed height and the horizontal boundaries would extend toward infinity. The aspect-ratio (Γ), or the ratio of the horizontal and vertical length scales would be infinite and the flow would only be inhomogeneous in the vertical direction. The majority of RBC applications which are well represented by wide horizontal layers [2], and so the infinite Γ case is of significant interest, but creating a domain to approximate $\Gamma = \infty$ is not straight forward. Since this scenario is impossible to create experimentally most investigations of RBC are performed with sidewalls. Adding side walls makes the analysis more complicated because it gives the flow a minimum of two inhomogeneous direction, it adds in additional physics through wall effects, and it bounds the lowest frequencies that can be seen inside the convection cell [7]. Through direct numerical simulations with horizontal periodicity the wall effects can be removed, but the frequency limitation is still present. Within the limit of Γ approaching infinity there must come a point where the central region of an RBC cell will become free from finite Γ effects but, this Γ has yet to be identified.

An additional problem that occurs in the analysis of turbulent RBC is identifying an appropriate mean with which to perform Reynolds decomposition. True Reynold's decomposition requires an infinite time average which is equal to the infinite ensemble average, or in other words the mean should be statistically stationary. In the canonical shear flows a mean can be found by averaging over the homogenous directions. For RBC where $\Gamma = \infty$ the flow can be averaged across horizontal layers in the same

manner, and the resulting mean flow must be zero for all velocity components at all spatial locations. This is because the buoyancy force is parallel to the vertical axis of the convection cell. However, in the case where Γ is limited due to side walls planar averaging can no longer be considered a completely valid method for determining the stationary mean flow, and it can no longer be assumed that the mean flow will be zero at all points in the domain. This leads to the first major research question of this study:

1) What is the best estimate for an infinite time averaged field for Rayleigh-Bénard convection in cylindrical cells?

Answering this question provides a basis to quantify the length and time-scale of the large-scale structures which have very long-life cycles and can mistakenly be considered part of the mean flow field.

There are two reasons why this study is being restricted to cylindrical cells. The first is because the azimuthal direction is periodic in cylindrical domains. This makes the mathematical analysis more tractable, and it only restricts the horizontal motions of large scale structures to the radial direction allowing for a closer approximation to the infinite Γ case than a geometry with corners. The second reason is that the majority of simulations and experiments in RBC are performed in cylindrical domains.

The next item of interest in this study is to identify the coherent structures in the field. Coherent structures can be defined by repeated patterns in the flow that last for a significant but finite amount of time. A significant amount of time in this context is a coherence time several times larger than the localized temporal scales. Since these structures have a finite life cycle it is impossible for them to reside in the mean flow field and they must reside completely in the fluctuating field. This is why

it is important to first identify a proper mean field that is robust and accounts for any instance of the field that may be seen in an ensemble.

Coherent structures are important because they occur on various scales of the flow and encapsulate the principle mechanisms of turbulence in an idealized model that is physical, visual and intuitive. This fundamental level of understanding and observation provides an important foundation for data analysis, as well as advancing theories and models. This leads to the next major research question of this study:

2) What are the properties of the large-scale coherent structures in Rayleigh-Bénard convection when the domain is a moderate aspect-ratio cylinder?

The emphasis on large-scale structures is because they tend to be the least homogeneous and most energetic structures in the flow field. These properties make them an attractive choice for characterizing and describing the properties of turbulent flows. Additionally, there is quiet a bit of evidence that the large-scale structures are created by assembly of small-scale structures [3, 4, 76]. A more complete discussion of this matter is provided later in the chapter.

While coherent structures are of great interest, they are often hard to identify and characterize. A common technique for overcoming these issues is to decompose the flow field (or portions of the flow field) into a set of linearly independent modes. Modal decomposition is a mathematically sound technique that is used across a wide range of applications. There are several different forms of modal decomposition that can be used such as Fourier, Chebyshev, Legendre, proper orthogonal decomposition, dynamic mode decomposition, among others. The modes have a mathematical definition that filters the structure in the flow field into narrow bands where the energy can be quantified. However, a looming question persists when modal decomposition is used to describe a stochastic, chaotic systems such as turbulence: are the modes well aligned

with the physical properties of the flow field such as coherent structures? If the modes are not well aligned with the physical properties of the flow then their results may not provide any additional insight into the physics. Understanding the strengths and limitations of modal analysis leads to the final research question for this study:

3) What physical insights can be obtained about the large-scale structures through a modal representation of the flow field inside the cylindrical domain?

This work utilizes Fourier analysis and proper orthogonal decomposition to extract physical insight from the flow field with varying success. The modal analysis results are contained in the later chapters of this document.

1.2 Novel Contributions

One of the important requirements of this work is to provide a novel contribution to the fields of science and engineering. This section outlines the contributions this work provides to these fields, and specifically the RBC community. A list of these contributions is provided below.

1. Provides an original, high-fidelity, direct numerical simulation that has the following unique characteristics
    - It represents one of the few studies with moderately high aspect-ratio
    - It contains one of the longest RBC direct numerical simulations in this turbulent regime with a total run time of 3054 free-fall times, or approximately 100 eddy-turnovers
    - It is one of the very few numerical studies to compare velocity measurements with experiments

2. Addresses the impact of improper averaging on moderate aspect-ratio RBC that has not previously been evaluated.
3. To the best of the author's knowledge, it is the first numerical study that utilizes resolved azimuthal Fourier modes to quantify the length and time scales associated with the large-scale structures, and the spatially variance of the integral time-scale within the flow field.
4. To the best of the author's knowledge, it is the first study to verify that moderate aspect-ratio RBC experiences global dynamics in the large-scale structures including a net rotation that is similar to the rotation events observed in unit aspect-ratio RBC.

1.3 Rayleigh-Bénard Background

1.3.1 General Information

Classical RBC occurs when a fluid is heated from below and cooled from above in a uniform manner, and the vertical axis is aligned with the pull of gravity (see figure 1). RBC is considered to be one of the canonical turbulent flows, and it has been a subject of interest in the thermal turbulence community for decades. A large portion of recent research has focused on how the global statistics and mean profiles of RBC scale throughout the turbulent regime. Noticeably less work has been published on the characterization and life cycle of the coherent structures, despite the fact that production of turbulence by buoyancy in RBC is one of the more easily conceptualized mechanisms among all of the canonical flows. One of the most detailed descriptions of turbulent RBC structures is provided by Zocchi *et al.* [76]. They describe five

characteristic structures: plumes, thermals, waves, swirls and a large-scale circulation (LSC). The plumes are either thermal columns that rise out of the thermal boundary layers or sheets near the boundary layer. The sheet-like plumes have a tendency to merge when they are in close proximity to one another [57]. Shishkina *et al.* [66] performed an extensive computational study of the sheet-like plumes to provide detailed descriptions of their geometric and physical characteristics.

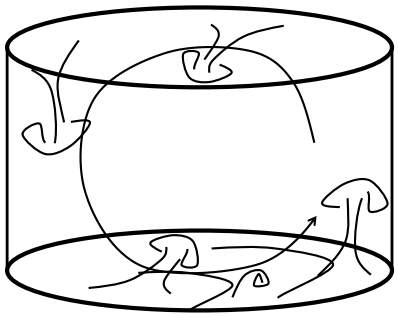


Figure 2: Illustration of the thermal life cycle as outlined by Zocchi *et al.*

Thermals are detached packets of fluid, or blobs that ascend or descend depending upon their temperature. These structures generally form when the stem of a thermal mushroom becomes too thin, and the head of the plume breaks off. Zocchi also identified another structure known as swirls where the thermal emission would curl back on itself. Models suggest that swirls arise from the same instability as plumes, but that the swirl has more shear acting in the upwind direction [65]. When plumes and thermals impact the opposing boundary layer a wave is generated that propagates out from the point of impact. Zocchi shows that these waves tend to propagate toward areas where the horizontal velocities decrease and vertical velocities dominate. The regions of large vertical motion generate LSC [58] which are sometimes described as "roll-cells."

These structures can be classified by their spatial size and coherence times. Thermal plumes, mushroom thermals and swirls can be considered small scale structures with relatively short coherence times. The roll-cells or large-scale-circulations are aptly named since they are a much larger structures with coherence times that significantly exceed those of thermal plumes. The thermal sheets can be considered an intermediate structure since they form as a result of the interaction between roll-cells and the small-scale thermals. This range of fine, intermediate and large-scale structures provides an interesting analogy to the range of coherent structures found in shear-flows. This analogy will be discussed in detail later in this chapter.

Typical RBC experiments are conducted in either cylindrical or rectangular containers where the side walls are insulated to approach the adiabatic limit and the top and bottom plates are kept at either constant mean temperature or constant mean heat flux. The investigations are conducted in a range where the Oberbeck-Boussinesq (OB) approximation can be considered valid. The OB approximation states that fluid density can be considered a linear function with respect to temperature and that all other material properties can be considered constant over the given temperature range. Under these conditions the RBC problem can be characterized by the shape of the domain and three dimensionless control parameters: Rayleigh number (Ra), Prandtl number(Pr) and aspect-ratio (Γ).

$$Ra = \frac{\beta g \Delta T h^3}{\alpha \nu} \tag{1.1}$$

$$Pr = \frac{\nu}{\alpha} \tag{1.2}$$

$$\Gamma = \frac{L}{h} \tag{1.3}$$

The Rayleigh number is comprised of the thermal coefficient of expansion (β),

gravitational constant (g), mean temperature difference between the top and bottom plates (ΔT), vertical height of the cell (h), thermal diffusivity (α) and kinematic viscosity (ν). The Prandtl number is the ratio of kinematic viscosity and thermal diffusivity and the aspect-ratio (Γ) is the ratio between a defining horizontal length (L) i.e. the diameter of a cylindrical cell, and the vertical height of the cell.

Of necessity both physical and numerical experiments are conducted in domains of finite horizontal extent i.e. finite Γ. The vast majority of experiments and numerical simulations have been conducted in low aspect-aspect ratio domains ($\Gamma \leq 2$). There are several compelling reasons for studying turbulent RBC in low Γ domains. In terms of general turbulence studies the fluid motion becomes increasingly turbulent as the Ra number is increased and the Pr number is decreased. More specifically, one of the current goals of the RBC community is to identify and characterize the transition to the "Ultimate" regime of turbulent RBC at high Rayleigh numbers. This regime was originally predicted by Kraichnan in 1963 [46] and a more recent review of the community's progress was provided by He, Funfschilling, Nobach, Bodenschatz and Ahlers in 2012 [35]. Numerically and experimentally it is less expensive to reach a higher Ra by increasing the height of the domain at a fixed diameter i.e. by decreasing Γ.

This focus on small Γ has lead to a relatively strong understanding of the heat transfer scaling prior to the ultimate regime. The Grossmann-Lohse (GL) theory [32, 31, 30, 29] has been presented as a unifying model for determining the scaling of the Nusselt (Nu) and Reynolds (Re) numbers over a large range of Ra and Pr.

$$Nu = \frac{Q_o h}{\alpha \Delta T} \tag{1.4}$$

$$Re = \frac{UL}{\nu} \tag{1.5}$$

The Nusselt number includes the kinematic heat flux (Q_o) and can be considered

the ratio of heat transported by convection and conduction. The Reynolds number depends upon the velocity and length scales that are selected. Typically h is selected as the Re length scale in RBC, but the velocity scale is not well defined since the flow is generated by internal forces. The GL theory defines its scaling predictions with a central assumption that there is only one characteristic mean velocity scale, and they use this scale to define Re. This scale is associated with a mean "wind," or a single circulation that travels across the bounding walls while leaving the core of the convection cell relatively homogeneous and well mixed. This velocity scale is often referred to as the "wind of turbulence" and it has a clear physical meaning in these small Γ cells. An excellent review of the GL theory, its successful predictive power, and the general status of turbulent RBC research can be found in the 2009 review by Ahlers, Grossmann and Lohse [5]. Clearly, a mean wind cannot occur in horizontally isotropic RBC because it would imply a preferred horizontal direction. The existence of a mean flow is a puzzling and vexing shortcoming of all finite Γ experiments. This is one identified short coming of the GL theory because it currently does not account for Γ effects in its scaling predictions.

The small Γ dominated work of the last few decades has greatly increased our understanding of thermal convection, and while there has been a steady increase in the Rayleigh numbers that are reached in experiments and simulations there is still a wide gap with respect to the enormous Rayleigh numbers that are seen in the geo and astro physical realms such as the Earth's atmosphere. This might be seen as sufficient justification to continue focusing virtually all research efforts on small aspect-ratio RBC cells, but there are compelling reasons to focus on wide aspect-ratio ($\Gamma > 2$) RBC as well. The background and justification for wide Γ studies will be discussed in the next section.

1.3.2 Large Aspect-Ratio

There are several compelling reasons to study RBC at larger Γ. First and foremost, the vast majority of RBC applications are not constrained by sidewalls, but are more appropriately modeled by wide, horizontal, fluid layers having large, but not necessarily infinite Γ [2]. This is important because adding side walls does two things. First, it adds in additional physics through wall effects, and second, it bounds the lowest frequencies that can be seen inside the convection cell [7]. Through direct numerical simulations with horizontal periodicity the wall effects can be removed, but the frequency limitation is still present. Within the limit of Γ approaching infinity there must come a point where the central region of an RBC cell will become free from finite Γ effects. However, this Γ and its relationship to Ra and Pr has yet to be identified.

When the aspect-ratio (Γ) of a convection cell is small ($\Gamma \leq 2$), a single LSC is the largest observable structure in the flow. However, when the Γ exceeds roughly 4 the LSC becomes a three-dimensional, multi-roll structure [56, 74]. This is an important observation, since the smaller scale structures are believed to be intricately tied to the roll-cells. However, little is known about the properties of the multi-roll cell because most analyses of coherent structures in RBC have been performed in small Γ domains ($\Gamma \leq 2$).

The most comprehensive study of Γ dependence for RBC in cylindrical domains was performed by Bailon-Cuba *et al.* [7]. Bailon-Cuba *et al.* studied how the heat transport and the LSC patterns vary as a function of Γ and Ra at a fixed Pr of 0.7. They observed a notable difference in the heat transfer as the LSC evolves from a single roll to a multi roll state. Bailon-Cuba *et al.* found that the global heat

transfer in the cell dips to a minimum during this transitionary regime, and that it saturates at a constant value as Γ exceeds 8. These results are in contrast to the earlier experimental work of Funfschilling *et al.* [28] where no Γ dependence was found for the global heat transport. However, there are several possible reasons why these results do not match up. The work of Funfschilling *et al.* was performed at a higher Prandtl number ($Pr = 4.38$), and was performed with a coarser sampling of Γ. Additionally, the some of the experimental results required adjustments due to the finite thermal conductivity of the test cell's walls. Therefore it is very reasonable to assume that the experiments of Funfschilling *et al.* may not have fully captured the fluctuations in Nusselt number, or that there is a Prandtl number dependence for the scaling of global heat transfer with respect to Γ.

Bailon-Cuba *et al.* [7] also observed that as Γ grows the LSC patterns favor pentagonal and hexagonal shapes and that these patterns did not display transient behavior in their analysis. However, in the more recent work of Emran and Schumacher (2015) an estimated time scale for the drift of these large-scale mean patterns was extracted from simulations of RBC in very wide aspect ratios ($\Gamma = 50$) and at relatively low Ra (5×10^5) [21]. Emran and Schumacher estimate that the characteristic time scale for the large-scale drift in the very wide Γ case is on the order of 10^3 free-fall time units which is well beyond the analysis time used in Bailon-Cuba's study.

While less is known about the physics of large aspect-ratio RBC, it has clearly been shown that important changes in the physics occur as Γ is increased. Understanding these changes and their implications are critical as we interpret the data that has already been collected, plan future experimental studies and move towards modeling large complex systems like the atmosphere or the oceans.

1.3.3 Equations and Scales

Turbulent RBC is governed by the Navier-Stokes equations, and any given instance of the flow can be characterized by Ra, Pr, Nu and Γ provided the OB approximation is valid. Virtually all analysis in RBC is performed with respect to these 4 dimensionless quantities, but their significance in the actual governing equations depends upon the scales that are selected for dimensional analysis. In this study three major scales have been used to perform analysis and they are the conduction, free fall, and Deardorff scales.

1.3.3.1 Conduction Scaling

As the title states, the conduction scales are derived from the conduction phenomenon. The scales are based off the time that it will take for heat to be transported across the layer depth and the resulting length (z_c), time (t_c) and temperature (T_c) scales are as follows:

$$z_c = h \tag{1.6}$$

$$t_c = \frac{h^2}{\alpha} \tag{1.7}$$

$$T_c = \Delta T \tag{1.8}$$

These scales are similar to the Townsend scales that Adrian *et al.* [2] analyzed. The main difference between Townsend's scales and the conduction scales defined in this section are their length scales. Townsend's length scales are based off a conduction layer thickness which is significantly smaller than the layer depth. Inside Townsend's

conduction layer thermal diffusion dominates the heat transport. While Townsend's scales are physically appropriate for analyzing conduction, and thermal boundary layers, the conduction scales defined in this section are presented because they were the first scales used in this study. The conduction scales were initially selected because the example simulations for RBC in the research code Nek5000 use this scaling.

When the Navier-Stokes equations are nondimensionalized by the Conduction scales the momentum and temperature equations take the following form:

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla P + Pr\nabla^2\mathbf{u} + RaPr\theta \tag{1.9}$$

$$\theta_t + (\mathbf{u} \cdot \nabla)\theta = \nabla^2\theta \tag{1.10}$$

In this form the diffusive and advection terms in the remain of the same order and the forcing term due to buoyancy is scaled with Rayleigh number.

1.3.3.2 Free Fall Scaling

The next set of scales to be analyzed are the so called "free fall" scales. These scales get their name from their velocity scale (w_f) which is defined as the velocity that the fluid would travel at if it were freely driven by the buoyancy force across the layer depth without any other limitations. The parallel is similar to the Newtonian description for the velocity a ball will reach if it is dropped from a given height. The velocity scale and accompanying length (z_f) and temperature (T_f) scales are defined below.

$$z_f = h \tag{1.11}$$

$$T_F = \Delta T \tag{1.12}$$

$$w_f = \sqrt{z_f \beta g T_f} \tag{1.13}$$

Additionally, a free fall time (t_f) can be defined by dividing the free fall velocity by the layer depth. The free fall scales become pertinent for the large scales as the flow becomes increasingly turbulent and viscosity plays a smaller role. When the Navier-Stokes equations are non-dimensionalized by the free fall scales the momentum and thermal equations take the following forms.

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla P + \sqrt{\frac{Pr}{Ra}}\nabla^2\mathbf{u} + \theta \tag{1.14}$$

$$\theta_t + (\mathbf{u} \cdot \nabla)\theta = \sqrt{\frac{1}{RaPr}}\nabla^2\theta \tag{1.15}$$

Equations 1.14 and 1.15 show scaling that is similar to the classic forms seen in shear flows. The Prandtl and Rayleigh number combinations reduce the role of the diffusion terms and allow the advection and forcing terms to become more dominant. It is probably a combination of this familiar form and the simple physical interpretation of the scales that has lead these scales to be one of the most popular for flow analysis in the RBC community.

1.3.3.3 Deardorff Scaling

Deardorff [18] proposed a set of scales that are similar to the free fall scales. The primary difference is that the temperature and velocity scales are defined around the kinematic heat flux (Q_o) instead of ΔT. The Deardorff length (z_D), temperature (T_D) and velocity (w_D) scales are defined as follows:

$$z_D = h \tag{1.16}$$

$$T_D = \frac{Q_o}{w_D} \tag{1.17}$$

$$w_D = (Q_o \beta g z_D)^{1/3} \tag{1.18}$$

Adrian *et al.* [2] found that vertical velocity and temperature rms profiles taken from thermal convection data collected via laboratory experiments, numerical simulations and the atmospheric measurements collapse on top of each other when they are scaled by the Deardorff scales. This universal scaling across a very wide spectrum of Rayleigh numbers makes them an ideal selection for analysis. When the Navier-Stokes equations are non-dimensionalized by the Deardorff scales they take the following form:

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla P + (\frac{Pr^2}{RaNu})^{1/3}\nabla^2\mathbf{u} + \theta \tag{1.19}$$

$$\theta_t + (\mathbf{u} \cdot \nabla)\theta = (\frac{1}{RaPrNu})^{1/3}\nabla^2\theta \tag{1.20}$$

A challenge for numerical simulations presents itself when the Navier-Stokes take the form of equations 1.19 and 1.20 because the Rayleigh and Nusselt numbers both appear in the equations. The Rayleigh number is partially defined by ΔT and is associated with Dirichlet boundary conditions. The Nusselt number relies on knowledge of the heat flux or Neumann boundary conditions. Defining both types of boundary conditions is not possible in numerical analysis since this would over define the problem. Therefore running a simulation with the Deardorff scaling requires the Rayleigh number to be predefined and the Nusselt number to be calculated in-situ or vice-versa. As the in-situ quantity varies the constants in front of the diffusions terms in equations 1.19 and 1.20 would need to be adjusted.

Even though there are challenges with running a simulation based off the Deardorff scales, their properties for scaling analysis are very useful. Richard Fernandes also used the Deardorff scales to compare experimental RBC data with atmospheric data as part of his PhD thesis [22]. His experiments are of particular interest since they were conducted at a relatively large Γ, and his results are the primary validation source for this numerical study. A brief introduction to his work is provided in the next section.

1.3.4 Experiments of Fernandes

Fernandes [22] performed a series of turbulent RBC experiments in a $\Gamma = 6.3$ test cell with Ra ranging from $10^5 - 10^9$. These experiments used particle image velocimetry (PIV) to explore the spatial structure of turbulent RBC with water as the working fluid ($Pr \approx 4.0 - 6.7$). The walls of the test cell had a viewing window cut into them so that two PIV cameras could be used to capture images. These cameras were placed in a side-by-side configuration to extend the field of view inside the test cell. This configuration was only able to capture two dimensional velocity data. A diagram illustrating the field of view with respect to the total diameter of the test cell can be found in figure 3. Further details on the experimental apparatus can be found in Fernandes' thesis.

It was noted in the previous section that Fernandes used the Deardorff scales for his analysis of the flow. However, he defined a length scale z^*:

$$z^* = 0.5 \times z_D \tag{1.21}$$

Fernandes used this length scale to facilitate easier comparison with atmospheric data since this was also part of his thesis work. His definitions of the other dimensionless variables (Ra, Nu, Γ) are equivalent to those provided in this work.

One of the most unique characteristics of Fernandes' experiments was the extremely long averaging times that were used. Fernandes' statistics were generated from sets of statistically independent, two-dimensional snapshots which were obtained over very long averaging times. These snapshots were collected in groups of 15 with ∼30-40 seconds between each snapshot. The heat source was periodically turned off for 30 minutes to an hour between sets of snapshots, then turned back on and allowed to

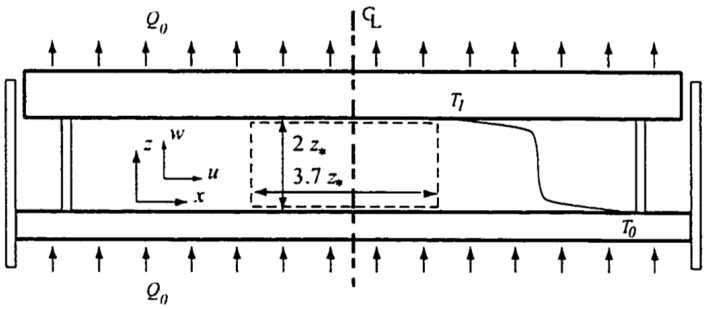


Figure 3: Illustration of the experimental device and the field of view for velocity measurements ($z^* = 0.5z_D$). Image reproduced from Adrian *et al.* (1986) with permission from the author.

settle for 4-12 hours. This perturbing of the flow was done to keep the large-scale-motions from developing any preferential direction so the horizontal planes could be considered statistically homogeneous in the cell's core. The total time for each of these experimental runs was on the order of hundreds of hours. The long averaging times and periodic perturbations employed in these experiments provide one of the closest approximations to an infinite time average in turbulent RBC. The single and double point statistics along with the mean data from Fernandes work is serving as a resources for validation for this study which is found in chapter 2.

1.3.5 Analogy to Shear Flows

Coherent structures occur on various scales of turbulent flow and they encapsulate the principle mechanisms of turbulence in an idealized model that is physical, visual and intuitive. This fundamental level of understanding and observation provides an important foundation for data analysis, as well as advancing theories and models. The study of coherent structures has become a large part of the field of turbulence because

of these reasons. A strong analogy can be drawn between the coherent structures that are observed in shear flows to those that are seen in turbulent RBC.

In shear flows it has been documented that small scale vortices very near the wall often take the form of hair pins [73]. Adrian *et al.* [3] has shown that as the hair pin vortices move down stream they tend to grow and spawn additional hair pins in their wake. Adrian classifies these hairpins as small scale structures see figure 4 (1a). These vortices combine into large scale structures as they align along a central axis and form a packet of hairpins [3]. Hairpin packets are classified as large scale structures because they can grow to significant size, and are one explanation for the bulges that are often seen in the instantaneous turbulent boundary layer profile [4]. An illustration of this is provided in figure 4(2a) These hairpin packets also have a tendency to self align and when they do noticeable low speed streaks occur along the central axis' of these structures [4]. This occurs because the hairpin rotation is such that the direction of flow between its legs opposes the direction of mean flow. High speed streaks are also observed in the boundary layers which are thought to occur between the series of low speed streaks as a consequence to the rotation between hairpins traveling in parallel. "Very large-scale motions" (VSLM) are characterized by structures that exceed the characteristic boundary layer length scales by a factor of 3 in the stream wise direction [45]. Adrian [4] has suggested that the VLSMs could be composed of a series of large hairpin packets that have self aligned see figure 4 (3a). Guala *et al.* [34] showed that in pipe flow these VSLM's can actually grow to lengths exceeding 3 pipe diameters. In this sense they are certainly "very-large" and can be considered some of the largest scales in the flow.

Turbulent RBC also has a rich set of coherent structures that span a wide range of length and time scales. On the smallest scales thermal plumes and sheets will rise

out of the boundary layers to transport heat across the vertical domain, see figure 4 (2a). At the onset of convection these thermal emissions are on the order of the layer depth, but as the level of turbulence increases they become smaller and increase in number. These small scale thermal structures tend to congregate in groups. One possible cause for this congregation is because the dynamics of opposing thermals from the opposite side of the convection cell impacting the boundary layer herds the thermals into concentrated areas. In areas where the density of thermal structures leaving the boundary layer is significantly high the plumes collectively cross the full layer depth and form a large scale up or down draft, see figure 4 (2b). These structures bear a strong resemblance to the large scale structures of shear flows because they are comprised of a series of small scale structures that self organize into a structure with greater coherence times and length scales. Additionally these large scale structures interact with one another generating a pattern of large scale circulations, see figure 4 (3a). Emran and Schumacher [21] have shown that these large scale patterns contain a dynamic scale, but the dynamics occur on such a long time scale that they have yet to be fully observed or understood. The dynamics and patterns of these large scale motions make a compelling analogy to the VLSM that have been identified in shear flows because they represent a structure that evolves on scales that are significantly greater than the large scale motions themselves.

This striking analogy hints at a universal nature for the life cycle of turbulence. In this life cycle small scales congregate to form larger structures and these larger structures form even larger structures still. The large scales also influence the small scales which forms a self perpetuating cycle as the different scales influence one another. The large scale genesis is dependent upon the small scale organization and the small scale organization is dependent upon the large scale interactions. Studying turbulent

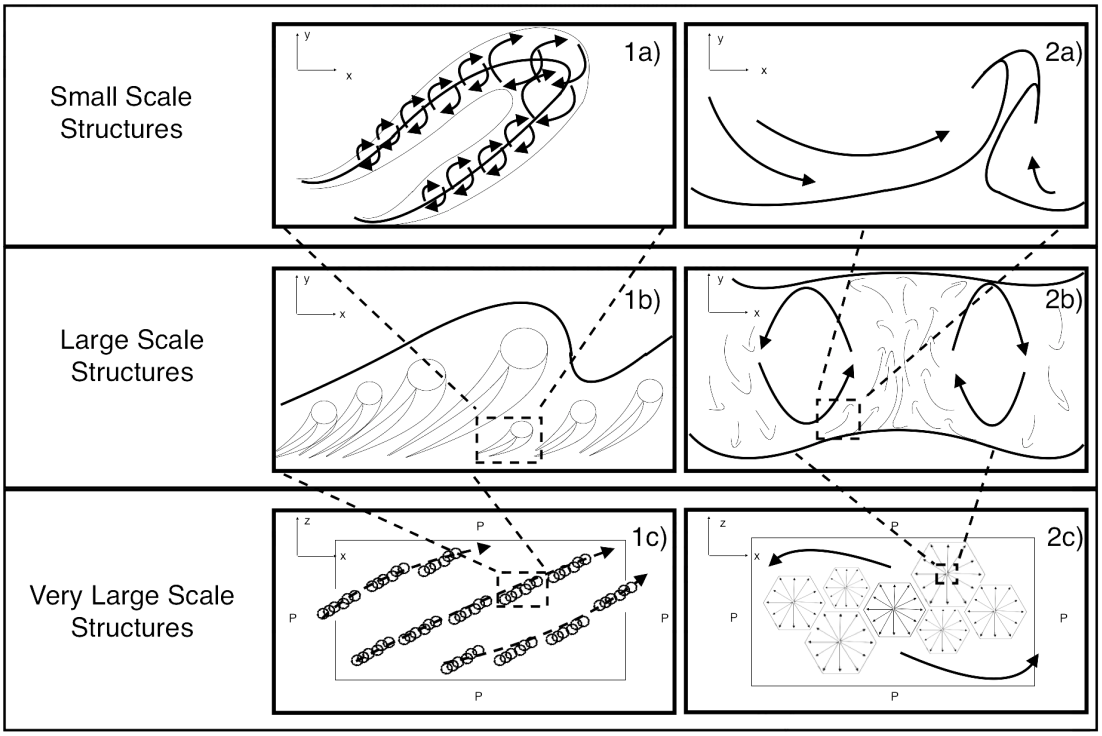


Figure 4: Comparison between the small scale (a), large scale (b) and very large scale (c) coherent structures seen in shear flows (1) and Rayleigh-Bénard convection (2).

RBC with this cycle in mind may help us to uncover a universal relation that can be applied to all turbulent flows.

1.4 Document Outline

This chapter (chapter 1) is focused on providing the background motivation for this work. Chapter 2 is also an introductory chapter, but its emphasis is on introducing the numerical methods and post processing techniques that are used throughout this study. The purpose of chapter 2 is to outline the numerical procedures and

document the development work that has been performed to generate the results in the proceeding chapters. The source code that accompanies the author's development work is provided in Appendix B.

Chapter 3 is containts the text from a previously published work by the author and two members of the committee in the International Journal of Heat and Fluid Flow [60]. It provides validation with experiments for the computational work and outlines the persistent large-scale structures that are seen in the flow field. In terms of the research questions presented earlier in this chapter, chapter 3 is predominantly directed towards laying the ground work for answering questions 1 and 2. Approval from the other authors to include the text from the journal paper in this document is provided in Appendix C.

Chapter 4 is primarily focused on answering the first research question of this work, what is the best estimate of the infinite time average? The chapter is written to address the broad need in the turbulence community to use careful averaging procedures to ensure the optimal estimation of the Reynolds average is obtained. RBC data is presented as an example for when care must be applied to the averaging techniques.

Chapter 5 is dedicated to studying the azimuthal decomposition of the flow field via Fourier modes. The physical interpretation of the modes is investigated by studying the average energy distribution across the modes, the temporal characteristics and evolution of the modes, and by comparing the spatial structure of the modes with total flow field. Affect of the inhomogeneous r and z directions are also investigated by studying how the energy content changes across these spatial dimensions.

Chapter 6 directly builds upon chapter 5 by performing proper orthogonal decomposition (POD) on the Fourier modes. The goal of this chapter is to see if additional

physically relevant structure can be drawn from performing a decomposition across the two inhomogeneous directions. A discussion on the merits and variables of POD as a tool for interpreting physics is also provided. Appendix A is a dictionary of POD mode visualizations that is provided as a companion to chapter 6. The number of POD modes is very large and only results from a small selection are presented in chapter 6. Chapters 5 and 6 are directed toward answering the second and third research questions and chapter 7 is dedicated to summarizing the findings and conclusions of the work.

Chapter 2

NUMERICAL METHODOLOGY

The flow fields that are being analyzed in this study are supplied from direct numerical simulation (DNS) of the Navier-Stokes equations. DNS simulations resolve all the length and time scales in the fluid domain and do not employee any form of modeling while computing the flow field. There are several numerical methods available for solving the Navier-Stokes equations including finite difference, finite volume, global spectral methods and the spectral element method. Finite difference and global spectral methods work on computational grids where the nodes must be ordered and spaced in a specific manner for the methods to work i.e. they require structured grids. Finite element, finite volume and spectral element methods work with unstructured grids where the computational grid can be subdivided into smaller cells, or elements which allows for added geometric flexibility. In this study the spectral element method is used to compute the flow field using the research code Nek5000. The cylindrical geometry in this work makes it difficult to use global spectral methods such as Chebyshev, or a Fourier Chebyshev combination. Global spectral methods are better suited for box domains and one such example in RBC is the classic DNS performed by Kerr in 1996 [44]. Adapting the computational methods into cylindrical coordinates leads to a large spatial resolution disparity between the center of the domain and the outer region, and this disparity increases quadratically with domain size. The spectral element method provides a higher-order accuracy and geometric flexibility. These qualities make SEM the truly ideal numerical method to perform this study's DNS computations in moderate aspect-ratio cylinders. This chapter serves

as a primer for the spectral element method, and it provides an overview of the post processing methodologies and routines that were developed specifically for this work.

2.1 Introduction to the Spectral Element Method

Nek5000 is based on high-order spectral element discretization and is highly-scalable on massively-parallel computers [25, 42]. It supports incompressible and low Mach number fluid dynamics (more than 200 published scientific papers), heat transfer, magneto-hydrodynamics (MHD), combustion [43, 6], and has been recently extended to include linear elasticity [54]. Nek5000 is maintained through an *svn* and *git* repositories and is freely available for download through the *svn checkout* command, or github. The repository contains all the source files and subroutines, as well as auxiliary tools for pre/post processing and several benchmark examples.

The spectral element method (SEM) combines the geometric flexibility of finite elements with the exponential convergence of global spectral methods. This is accomplished by solving equations with the continuous Galerkin method. Galerkin's method is a form of the method of weighted residuals. The method of weighted residuals requires the equation's variables to be approximated with a set of trial functions. The inner product of the trial functions with another set of functions known as the test functions produces an error norm that can then be minimized. With Galerkin's method the trial and test functions are the same and are often referred to as the shape or basis functions. Some restrictions exist for the basis functions when Galerkin's method is employed. The basis functions must be orthogonal to one another at the sampling points i.e. element nodes, and they must sum to 1 between the nodes.

Lagrange interpolating polynomials exhibit these characteristics and are a common choice for basis functions.

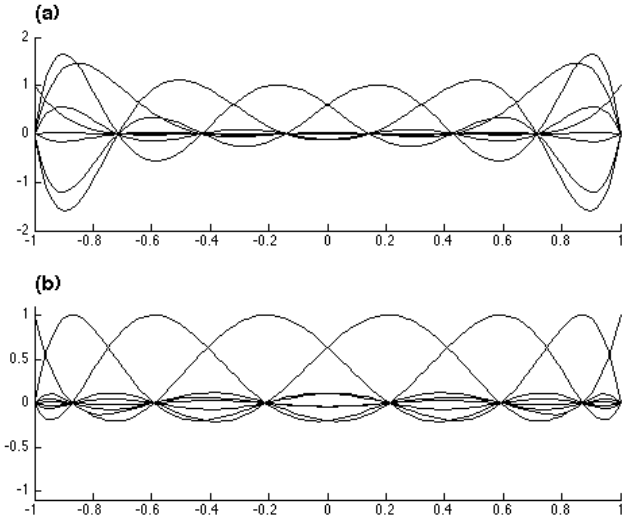


Figure 5: Distribution of 7th order basis functions using equally spaced points (a) and the Gauss-Lobatto-Legendre distribution (b)

The most crucial differentiator between SEM and the more traditional Galerkin formulation of the finite element method is that SEM can support high order polynomial basis functions and remain stable. Any polynomial basis function must have p+1 nodes to represent it, where p is the polynomial order. The traditional finite element method places these nodes at equally spaced locations. This works for low order polynomials, but as the order is increased the basis functions become highly oscillatory near the element boundaries as seen in figure 5. This leads to computational problems with the solutions becoming unstable. SEM uses a non-uniform distribution of nodes which clusters nodes toward the boundaries to alleviate this problem. As a result SEM solvers are able to reach significantly higher order polynomials than the traditional

finite element formulations. This polynomial or p-type refinement is what allows SEM to obtain convergence that is similar to global spectral methods.

2.1.1 Galerkin Method and Weak Formulation

It was stated in the previous section that Galerkin's method is a flavor of the method of weighted residuals. In Galerkin's method variables are represented by trial function which is comprised of a characteristic set of basis functions and a set of coefficients. The method of weighted residuals constructs an error norm by taking the inner product of the trial functions with another set of functions known as the test functions. The minimization of this norm leads to the approximate solution of the equations that are being solved. In the Galerkin method the trial and test functions are constructed from the same basis. The definitions and examples will be presented with only 1 spatial dimension in this section for simplicity's sake.

$$u^{\delta} = \sum_{i=0}^{N} \hat{u}_i \theta_i(x) \rightarrow \text{Trial Function} \tag{2.1}$$

$$v^{\delta} = \sum_{j=0}^{N} \hat{v}_j \theta_j(x) \rightarrow \text{Test Function} \tag{2.2}$$

$$(F(u^{\delta}), v^{\delta}) = \int_{\Omega} F(u^{\delta}) v^{\delta} d\Omega = 0 \tag{2.3}$$

The δ super script in equations 2.1 and 2.2 signifies the basis representation of the functions which is an approximation of the exact function. Evaluating the inner product of the trial and test function requires integration over the solution domain (see equation 2.3) and this process is greatly simplified through numeric integration. Numeric integration is ideal since it can be handled discretely and it has a high level of accuracy. To make the process uniform for all types of problems the integration is

performed in local coordinates that range from -1 to 1 and the problem is transformed into its actual (or global) coordinates by a simple change of coordinates. Numeric integration is performed by multiplying the function at set of given set of points in the local coordinates by an accompanying set of weights and then summing the results. The characteristic integration points and weights are often referred to as the quadrature. In the SEM the integration points and the trial function coefficients are defined at the same locations.

Consider the 1D Poisson equation with a constant right hand side (G) and homogeneous Dirichlet boundary conditions:

$$\nabla^2 u(x) = G, x \in a : b \tag{2.4}$$

Taking the inner product of both sides with respect to the test function leads to the following:

$$(\frac{d^2u}{dx^2}, v) = (G, v) \tag{2.5}$$

$$\int_a^b \frac{d^2u}{dx^2} v(x) dx = \int_a^b v(x) G dx \tag{2.6}$$

The derivative on the left hand side of the equation can be recast into the weak form through integration by parts.

$$v\frac{du}{dx}|_a^b - \int_a^b \frac{du}{dx}\frac{dv}{dx} dx = \int_a^b v(x) G dx \tag{2.7}$$

An additional property of the test function that was not mentioned previously is that it will have a value of zero at all Dirichlet boundary conditions. This is because the exact solution is known at Dirichlet boundary conditions, and so a way to ensure that the inner product is zero at these locations is to require the test function to be zero at these points. If this problem had Nuemann boundary conditions then this term would not completely disappear. With this understanding the boundary term in

2.7 can be set to zero. Now the problem can be expressed in its discrete form and transformed to local coordinates.

$$-\int_{-1}^{1}\sum_{i=0}^{P}\sum_{j=0}^{P}\hat{u}_i\hat{v}_j\frac{d\theta_i(\xi)}{d\xi}\frac{d\theta_j(\xi)}{d\xi}\frac{d\xi}{dx}d\xi=\int_{-1}^{1}\sum_{j=0}^{P}\hat{v}_j\theta_j(\xi)G\frac{dx}{d\xi}d\xi \tag{2.8}$$

The value P in equation 2.8 is the polynomial order of the basis that is being used. To cast the problem in a completely discrete form the continuous integrals must now be converted into discrete numeric summations with weights from the selected quadrature.

$$-\sum_{k=0}^{P}\sum_{i=0}^{P}\sum_{j=0}^{P}w_k\hat{u}_i\hat{v}_j\frac{d\theta_i(\xi_k)}{d\xi}\frac{d\theta_j(\xi_k)}{d\xi}\frac{d\xi_k}{dx}=\sum_{k=0}^{P}\sum_{j=0}^{P}w_k\hat{v}_j\theta_j(\xi_k)G\frac{dx}{d\xi_k} \tag{2.9}$$

The final step that will be shown in this section is to reorder the sums and cancel the test function coefficients.

$$-\sum_{k=0}^{P}\sum_{i=0}^{P}w_k\hat{u}_i\frac{d\theta_i(\xi_k)}{d\xi}\frac{d\theta_j(\xi_k)}{d\xi}\frac{d\xi_k}{dx}=\sum_{k=0}^{P}w_k\theta_j(\xi_k)G\frac{dx}{d\xi_k} \tag{2.10}$$

Equation 2.10 represents the most general form of the solution through the Galerkin method. Further simplification requires specific knowledge of the basis functions. After applying the boundary conditions it can be cast into a matrix form with P-1 equations and P-1 unknowns and solved using standard numerical techniques.

This section is meant to serve as a primer to the basic, continuous Galerkin method and additional topics such as handling multiple elements, various boundary condition combinations, the solution to other ordinary and partial differential equations, and advanced topics such as the discontinuous Galerkin formulation can be found in any of the numerous texts devoted to finite element theory. One final comment on the implementation of the Galerkin method in SEM is that solutions are guaranteed to be C0 continuous. This means that solutions are guaranteed to be continuous

across element boundaries for the primary variables, but not the primary variables' derivatives. When higher order polynomials are implemented in the SEM method the solution can converge towards continuous derivatives significantly faster than if the solution were calculated with lower order polynomials. This is one of the significant advantages of SEM.

2.2 Post Processing Techniques

This section of the chapter focuses on the post processing techniques that are utilized throughout the work. The applications and analysis that accompany these techniques are not the emphasis of this section because those subjects are thoroughly discussed in the following chapters. This section serves primarily as a source to document the development work and implementation of the post processing techniques. It is a high level guide to the implementation of the methodologies that are used throughout the work, and the source code is also provided in Appendix B. The source code is also hosted on github at https://github.com/psakievich/DissertationCode and freely available to the public.

2.2.1 Global Mapping and Projection Operations

The initial and foundational step for post processing in this work is a global projection from Nek5000's computational grid (unstructured, cartesian coordinate system) to a structured grid in cylindrical coordinates. Computationally advancing the Navier-Stokes equations in the cartesian coordinate system with an unstructured grid is favorable from a mesh resolution stand point, but the natural frame of reference for

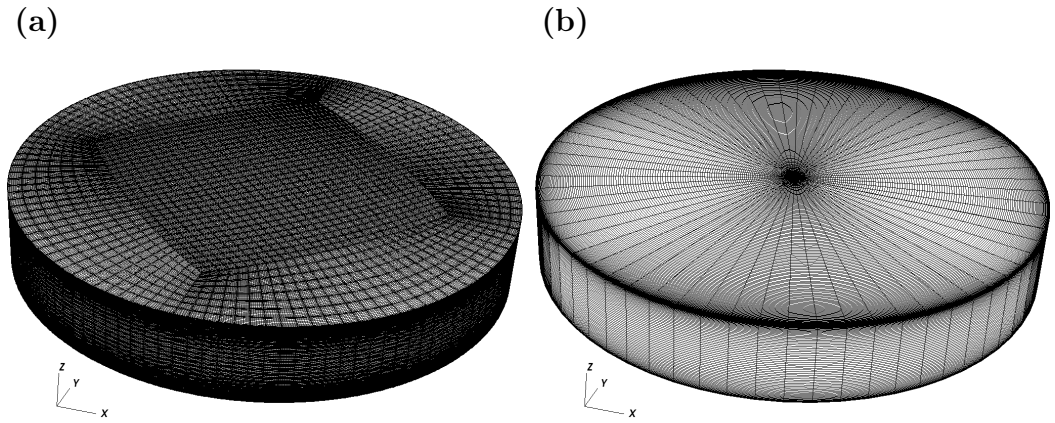


Figure 6: Images of the grid used to calculate the flow field (a) and a sample grid for post processing in cylindrical coordinates (b).

analysis of the physics is in cylindrical coordinates. Performing a projection operation between the two coordinate systems in post-processing mode ensures that the data is from a fully resolved field, even if the regions of the cylindrical coordinate system are not sampled at a fine enough rate for DNS computations. An example of grids from each of the coordinate systems is provided in figure 6.

Development of the global projection routines occurred in three stages. The first stage was to sample points at various r, θ and z locations over a small portion of the domain, specifically the central region of the cell. A simple matlab script was developed to write the desired points to an ASCII file that can be read by Nek5000's native spectral interpolation routine *hpts*. *hpts* reads the ASCII file, divides the points between processors, performs a global search to find the specified points, and then stores the processor, element, and local coordinate locations of each of the points. Storing this locating data allows the routine to efficiently interpolate the data with the same order of accuracy as the underlying basis function, and communicate across

processors via the highly scalable crystal router algorithm [25]. This matlab based method is sufficient for sampling small portions of the domain, but it is ill-suited for interpolating across the entire domain because *hpts* writes data to a single file in ASCII format. As the desired number of points and the number of snapshots both increased this format was deemed unsuitable for continued analysis.

The second stage was to modify the native *hpts* routine and the subroutines called within *hpts* so that the results were output in binary format with Nek5000's native input/output (IO) formats *.fld* and *0.f*. The original routines were copied into a separate file, and renamed with the prefix *ps* to signify modifications by the author. Defining a series of independent routines ensured that the old routines could still be called and all dependencies within the source code were maintained. These routines are documented in Appendix B in the file PhilFuncs.f.

The second stage routines still required an ASCII input file, but the output became a series of independent binary files that could be post processed with the open source visualization software Visit [15]. However, post processing with other software packages and detailed scripting with these routines was not available without substantial additional development. This isbecause the Nek5000 output files are not widely supported, and do not follow the conventions of other, more standard, file formats. Further more, the Nek5000 format is designed to output unstructured grids with higher-order elements that have the same number of points in each principle direction. This proved to be very restrictive when trying to output data that was sampled to capture physics, and further complicated other operations such as Fourier transforms in the azimuthal direction. Several efforts were made to allow for non-uniform sampling within an element, but on each attempt the errors propagated further into the source code. Eventually it was determined that the best course of

action would be to incorporate an entirely separate IO format with wide support across multiple software platforms.

After researching the options it was decided that the Visualization Tool Kit (VTK) library [63] would provide the most flexibility, power and portability for the interpolated datasets. VTK is a mature, open source, C++ library that is primarily used to analyze scientific data. The VTK file formats are supported by a wide range of post processing tools, and large library of file conversion functions are available to translate VTK files into other formats (HD5, ensight, etc). VTK file formats also include support for many data types including rectilinear, curvilinear, and unstructured grids. Thus developing an interface between VTK and Nek5000 lays the foundation for many additional geometries that extend beyond the cylindrical grids in this study. Another major advantage of the VTK library is that wrappers have been developed to support additional programming languages such as Java, Python and TCL. Access to scripting languages such as Python and TCL expand the post processing possibilities and reduce the development time needed to generate post processing routines. Developing the VTK interface also involved a sizable set of Fortran routines on the Nek5000 side. These additional routines serve as an interface between interpolation storage variables in Nek5000, and the C++ VTK routines. The details and rational of this development project are documented in the next section.

2.2.2 Merging Nek5000, FFTW and VTK

In addition to adding in IO support through VTK, another key objective was to integrate fast Fourier transforms (FFT's) in the cylinder's azimuthal direction. In fact, the main reason the high resolution global projections are required in this work

is to facilitate the use of spatial FFT's. These two objectives were resolved during a single development project because they are interrelated, and a description of the resulting data pipeline for processing datasets is provided below.

Projections between coordinate systems and Fourier transforms are accomplished by decomposing the domain into azimuthal rings so that Fourier transforms can be performed within the Nek5000 subroutines. The files that contain these subroutines are MYFFT, IntPntsFFT.f, psVtkOutput.cpp, and psVtkOutput.h, and all of these routines are included in Appendix B. MYFFT contains a list of parameters that the other routines use at compile time to determine the size of the arrays, the number of processors to use when performing FFT's, the azimuthal resolution, and the number of FFT's each processor will perform. IntPntsFFT.f contains the fortran subroutines that are used for the interpolation process, FFT setup and execution as well as the companion declarations for the C++ routines in the psVtkOutput files. The code that actually executes the FFT operations is the FFTW library [27] whose specific documentation can be found online.

A high level description for how the fortran subroutines within IntPntsFFT.f operate is provided below.

1. Declare the global sampling based on the domain description in MYFFT
2. Assign the portion of the domain to each processor, and declare the points
3. Find the points in the global domain using the base routines from *hpts*
4. Perform the interpolation
5. Set up the FFTW plan
6. Transform the velocity and spatial points to their respective cylindrical representation
7. Perform the FFT

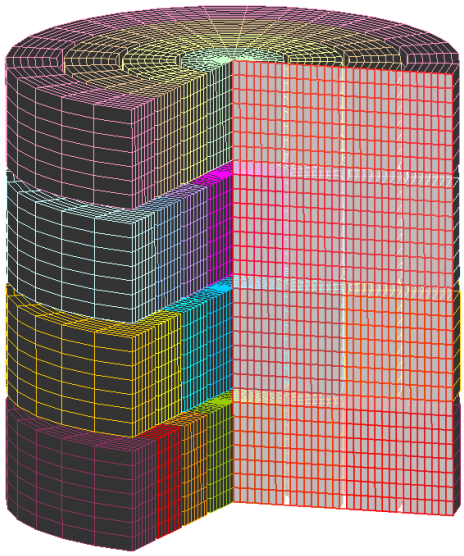


Figure 7: Visual representation of the domain decomposition and wave number assembly process. Each colored azimuthal ring is stored and operated on by an individual processor and the light red plane illustrates the assembly across all processors for a particular wave number.

8. Gather each wave number to a single processor
9. Call C++ routines to output a two-dimensional plane of complex Fourier coefficients

A visualization of the domain decomposition and wave number reconstruction process is provided in figure 7. Reassembling the decomposed data into a series of files containing the Fourier coefficients for a given wave number generates smaller individual files for linearly independent datasets. This facilitates highly parallel batch processing of the data, and it partitions the data to a size where multiple files can easily be analyzed on a laptop. The routines can also be utilized to output the spatial decomposition of the domain by having each processor export the azimuthal ring in an individual VTK file.

The C++ routines in the psVtkOutput files use VTK's object oriented framework

to receive pointers to the array storing the complex Fourier coefficients for a given wave number and another array containing the spatial locations. Individual objects are created to store the points and Fourier coefficients, with individual objects allocated for real and complex portions of the Fourier coefficients. These objects are populated with the actual values, and then high level objects for the overall grid and file writer are created. The grid is populated with the data, which is followed by the writer being populated with the grid and then finally the file is written to disk. A second set of routines is also provided for outputting the raw, untransformed data. These routines are independent of the geometry and so they can reused for any structured grid in future studies.

2.2.3 Python Scripting

One of the major benefits of using the VTK library is the ability to directly access and process the data within Python. Python has a very large number of libraries, and incredible versatility. It can be used to perform shell scripting, perform calculations in parallel (via mpi4py), file manipulation, scientific plotting and a myriad of other applications.

A key python tool that is used heavily throughout this work is the modred library [10]. Modred is an open source library for performing modal decomposition and analysis such as proper orthogonal decomposition (POD), dynamic mode decomposition (DMD), balanced POD (BPOD) and many others. The library is abstracted in such a way that any dataset can be used to perform these analysis as long as the user defines a class that supports the fundamental linear algebra operations of vector addition, vector scalar products and an inner product between two vectors. This

abstraction has proven useful for lots of additional post processing routines beyond the framework of modal analysis. Two modules containing modred vector classes were developed for this work and are documented in Appendix B: MrImaginaryVtk and MrRealVtk. As the names suggest, these modules are differentiated by their ability to handle complex or real datasets.

Python has been utilized for a large chunk of the post processing in this work, and there are additional modules for performing the inverse Fourier transforms on the VTK files that contain Fourier coefficients, calculating statistics and integral time scales from the Fourier coefficients, and POD calculations. Samples of these calculations are also provided in the appendix, however no additional details will be provided in this section since the formulas that define these quantities receive detailed discussions in later chapters.

Chapter 3

LARGE-SCALE THERMAL MOTIONS OF TURBULENT RAYLEIGH-BÉNARD CONVECTION IN A WIDE ASPECT-RATIO CYLINDRICAL DOMAIN

Abstract

The large-scale structures that occur in turbulent Rayleigh-Bénard convection in a wide-aspect-ratio cylindrical domain are studied by means of direct numerical simulation. The simulation is performed in a 6.3 aspect-ratio cylindrical cell with a Rayleigh number of 9.6×10^7 and Prandtl number equal to 6.7. Single-point and double-point statistics compare well against experimental results under nearly identical conditions. Large-scale thermal motions with coherence times exceeding 20 eddy-turnovers ($\sim$600 free-fall time units) are seen in the instantaneous fields. Temporally filtering them by integrating over approximately one eddy-turnover time scale reveals a clear pattern consisting of seven discrete thermal structures: three warm, rising sectors, three cool, falling sectors and a single plume of warm, rising fluid that wanders around the center of the cylindrical cell. Smoothing over still longer times (10 and 20 eddy turn-over time scales) yields a clear hub-and-spoke pattern of warm and cool sectors in a dominantly 120 degree periodic pattern separated by concentrations of radial vortex lines (the spokes) plus a nearly circular plume at the center of the test section (the hub). The similarity of the patterns in the instantaneous fields and the long-time smoothed fields demonstrates long persistence of these structures, a defining characteristic of coherent structures in turbulence. The warm and cool sectors are intimately linked with conical roll-cells rotating about the spokes, and these circulations

are likely the analogs of the 'wind of turbulence' found in low-aspect-ratio RBC experiments.

3.1 Introduction

Thermal convection plays an important role in many natural and engineered systems. Of the many different forms of thermal convection Rayleigh-Bénard convection (RBC) is by far the most studied [1, 4, 2]. It is one of the canonical turbulent flows, and it has been a subject of interest in the thermal turbulence community for decades. RBC occurs when a fluid between parallel, horizontal planes is heated from below and cooled from above by horizontally uniform boundary temperatures or heat fluxes. Despite this seemingly simple configuration many outstanding questions regarding the nature of turbulent RBC remain unanswered.

Recent research on RBC has focused on how the global statistics and mean profiles in small aspect-ratio cells scale throughout the turbulent regime. Less work has been done on the characterization and life cycle of the coherent structures, despite the fact that production of turbulence by buoyancy in RBC is one of the more easily conceptualized mechanisms among all of the canonical turbulent flows. One of the most detailed descriptions of turbulent RBC structures in low aspect-ratio cells is provided by Zocchi, Moses and Libchaber [27]. They describe five characteristic structures: plumes, thermals, waves, swirls and a large-scale circulation (LSC).

Plumes are either thermal columns or sheets that rise out of the near-wall thermal boundary layers. The sheet-like plumes have a tendency to merge when they are in close proximity to one another [17]. Shishkina and Wagner [22] performed an extensive

computational study of the sheet-like plumes to provide detailed descriptions of their geometric and physical characteristics.

Thermals are transient detached packets of fluid, or blobs that ascend or descend depending upon their temperature. These structures begin as small protrusions from the near-wall viscous-conductive layer and assume a characteristic mushroom shape as they grow. Zocchi *et al.* [27] also identified another structure known as swirls where the thermal emission curls back on itself. Models suggest that swirls arise from the same instability as thermals and plumes, but that the swirl has more shear acting in the upwind direction [21]. When plumes and thermals impact the boundary layer of the opposing wallside, a wave is generated that radiates out from the center of impact. The work of Zocchi *et al.* [27] shows that these waves tend to propagate toward areas where the horizontal velocities decrease and vertical velocities dominate. The regions of large vertical motion generate LSC [18] which are sometimes described as "roll-cells." Aside from the waves, similar motions have been observed in relatively wide aspect-ratio (Γ=width/height) convection cells by Adrian, Ferriera and Boberg [1] among others.

When the aspect-ratio of a convection cell is small ($\Gamma \leq 2$), a single LSC is the largest observable structure in the flow. However, when Γ exceeds roughly 4 the LSC becomes a three-dimensional, multi-roll structure as noted by du Puits, Resagk and Thess [16], as well as Xia, Sun and Cheung [25]. This is an important observation, since the smaller scale thermals and plumes are believed to be advected horizontally by the roll-cells. Little is known about the properties of the multi-roll cell because most studies of coherent structures in RBC have been concerned with small Γ domains ($\Gamma \leq 2$).

Thus far, the most comprehensive study of Γ dependence for RBC in cylindrical

domains was performed by Bailon-Cuba, Emran and Schumacher [3]. In this work they studied how the heat transport and the LSC patterns vary as a function of Γ and Rayleigh number (Ra) at a fixed Prandtl number (Pr) of 0.7. They observed a notable difference in the heat transfer as the LSC evolves from a single roll to a multi roll state. They also observed that as Γ grows the LSC patterns favor pentagonal and hexagonal shapes. In a more recent work Emran and Schumacher [9] estimate the time scale for the drift of these large-scale mean patterns from simulations of RBC in very wide aspect ratios ($\Gamma = 50$), but at relatively low Ra (5×10^5). Their estimate of the characteristic time scale for the large-scale drift in the very wide Γ case was on the order of 10^3 free-fall time units, well beyond the computational analysis time in their work or Bailon-Cuba's [3] study.

The goal of this paper is to elucidate the geometrical structure and organization of LSCs in a cylindrical domain having moderately wide aspect-ratio ($\Gamma = 6.3$) This is accomplished by direct numerical simulation (DNS) using a spectral element code. Among the various methods of numerically simulating turbulent RBC we prefer DNS over RANS, LES, and hybrid RANS-LES [14, 13, 6, 24, 26] because the governing equations of DNS do not depend on semi-empirical parameters. We use temporal filtering on several different time-scales to extract the large, coherent structures from an otherwise chaotic turbulent field. We study a single, but representative case at $\Gamma = 6.3$, Rayleigh number $= 9.6 \times 10^7$ and Prandtl number$= 6.7$, values that permit highly resolved DNS so detailed that the nature and organization of turbulence can be observed with accuracy throughout the entire domain. The parameters for this study were selected to facilitate direct comparison of statistics with the experiments of Fernandes [10].

The paper is organized as follows: Section 2 presents the numerical methodology,

Section 3 presents comparisons of double- and single-point statistics with the experiments of Fernandes [10], Section 4 introduces the flow structures, Section 5 describes their organization and temporal coherence more carefully by applying temporal filters to the flow-fields, and in Section 6 we draw the conclusions.

3.2 Numerical Methodology

Direct numerical simulation (DNS) of RBC in a cylindrical cell with an aspect-ratio ($\Gamma = width/height$) of 6.3 was performed using the spectral element code Nek5000. Nek5000 is a highly-parallelizable, well-vetted code for solving the incompressible, Navier-Stokes equations, and it currently has over 225 world-wide users [11]. The non-dimensional form of the Boussinesq equations for thermal convection is:

$$\nabla \cdot \mathbf{u} = 0, \tag{3.1}$$

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}}\nabla^2 \mathbf{u} + \theta \hat{z}, \tag{3.2}$$

$$\theta_t + (\mathbf{u} \cdot \nabla)\theta = \frac{1}{\sqrt{RaPr}}\nabla^2 \theta, \tag{3.3}$$

where $\mathbf{u}$, p and θ are velocity, pressure and temperature. Equations 3.1-3.3 were scaled spatially by the height of the cell (h), thermally by the temperature difference between the top and bottom plates (ΔT), and the velocity was scaled by the "free-fall velocity,"

$$w_f = \sqrt{hg\beta\Delta T}. \tag{3.4}$$

The Rayleigh (Ra) and Prandtl (Pr) numbers in equation 2 are defined as:

$$Ra = \frac{\beta g \Delta T h^3}{\alpha \nu}, \tag{3.5}$$

$$Pr = \frac{\nu}{\alpha}, \tag{3.6}$$

where β, g, α and ν are the coefficients of thermal expansion, gravitational constant, thermal diffusivity and kinematic viscosity respectively. The parameters of this simulation were selected to allow direct comparison with the experiments of Fernandes [10]. Fernandes conducted a series of experiments in a 6.3 Γ test cell with cylindrical side-walls. These experiments used water as the working fluid with Prandtl number ranging from 4.0 to 6.7 and Rayleigh numbers ranging from $5.8 \times 10^7 - 1.1 \times 10^9$. Our simulation was conducted with Prandtl and Rayleigh numbers of 6.7 and 9.6×10^7 respectively. The simulation's boundary conditions were no-slip at all walls, constant temperature at the top and bottom plates and zero heat-flux along the side walls.

In our simulation Ra was gradually ramped from the edge of the turbulent regime ($Ra = 5 \times 10^5$) to the target Rayleigh number, and then allowed to reach a stable, fully-developed state before data was collected. The simulation was judged to be fully developed after 8 eddy-turnovers because the volume-averaged, kinetic energy in the cell began oscillating about a steady value. The eddy-turnover time for the roll-cells (t_ϵ) was estimated as the time it takes for a particle to cross the layer-depth twice:

$$t_\epsilon = \frac{2h}{< w_{rms} >_V} \tag{3.7}$$

where $< w_{rms} >_V$ is the volume-averaged, vertical, r.m.s. velocity. For reference, each eddy-turnover is roughly 30 free-fall time units (h/w_f). The DNS data used for analysis in this paper spanned approximately 20 of these eddy-turnover's, or 615 free-fall time units, and accounts for approximately 41 minutes in dimensional time. Brown, Nikolaennko and Ahlers [5] identified a random reorientation of the LSC

in a unit Γ cell that occurred on a time scale of approximately 10 eddy-turnovers. Fernandes [10] also identified coherence in the large-scale structures that met or exceeded 10 eddy-turnovers. Based off these observations 10 t_ϵ was determined to be a scale of interest, and in the context of this paper it is treated as a medium, or intermediate temporal scale.

3.2.1 Numerical Resolution

The spatial domain was discretized with hexahedral elements and a marginal amount of biasing toward the upper and lower plates was applied to the element distribution. The spectral element method (SEM) used in this simulation also applies a Gauss-Lobatto-Legendre (GLL) quadrature which clusters points inside each element toward the boundaries and greatly improves resolution at the walls. Ninth order polynomials were used for the quadrature resulting in roughly 44 million grid points. Fernandes calculated the Kolmogorov length for this scenario to be approximately $1.2 \times 10^{-2}h$ and our simulation's grid had 5 points within this range at the wall. We also determined that this grid satisfies the spatial resolution criteria of Grötzbach [12]. The temporal resolution for each time step was approximately $t_\epsilon \times 10^{-4}$ with a corresponding CFL range of $\sim 0.6 - 0.7$.

We have conducted an *a-posteriori* analysis to evaluate the resolution of our results utilizing the techniques outlined by Scheel, Emran and Schumacher [20]. Scheel *et al.* [20] performed the majority of their analysis using Nek5000, and provided several specific methods for determining adequate resolution in SEM simulations of RBC [20]. One of their methods evaluates the vertical profiles of kinetic energy dissipation (ϵ) and thermal dissipation (ϵ_T) which are defined below:

$$\epsilon = \frac{1}{2}\sqrt{\frac{Pr}{Ra}}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)^2 \tag{3.8}$$

$$\epsilon_T = \sqrt{\frac{1}{PrRa}}(\nabla \theta)^2 \tag{3.9}$$

Profiles of these two scalar quantities are a good metric for judging convergence in SEM simulations because they are comprised of the derivatives of primary variables $(\mathbf{u}, \theta)$. SEM results are guaranteed to be C0 continuous, or continuous in the primary variables, within the elements and across element boundaries. The derivatives of these terms, or secondary variables are guaranteed to be continuous within the elements, but not across element boundaries [8]. Thus, discontinuities in the derivatives across element boundaries are commonly used to detect under-resolved simulations. Profiles for both dissipation terms are presented in the top two panels of figure 8. To illustrate smoothness and continuity of the present computations, the dissipation profiles and element boundaries are shown in the near-wall region, where the gradients are largest. These profiles were generated through area and time averages in the same manner as Scheel *et al.* [20]. The smoothness of the dissipation profiles testifies that the simulations are well resolved according to this criterion.

Another methodology, initially proposed by Bailon-Cuba *et al.* [3] and utilized by Scheel *et al.* [20], is to define height dependent Kolmogorov (η_k) and Batchelor (η_B) scales by averaging over horizontal planes and time.

$$< \eta_K(z) >_{A,t} = \frac{Pr^{3/8}}{Ra^{3/8}} < \epsilon(z)^{-1/4} >_{A,t} \tag{3.10}$$

$$< \eta_B(z) >_{A,t} = \frac{1}{Pr^{1/8}Ra^{3/8}} < \epsilon(z)^{-1/4} >_{A,t} \tag{3.11}$$

From here the original Grötzbach criterion can be modified to the following [3, 20]:

$$\frac{\Delta z(z)}{< \eta(z) >_{A,t}} < \pi \tag{3.12}$$

where $\Delta z(z)$ is the vertical gird spacing and η is either the Batchelor or Kolmogorov scale. We have plotted this criterion for both the Kolmogorov and Batchelor scales in the lower right panel of figure 8 where it can be seen that we are adequately resolved according to this criterion.

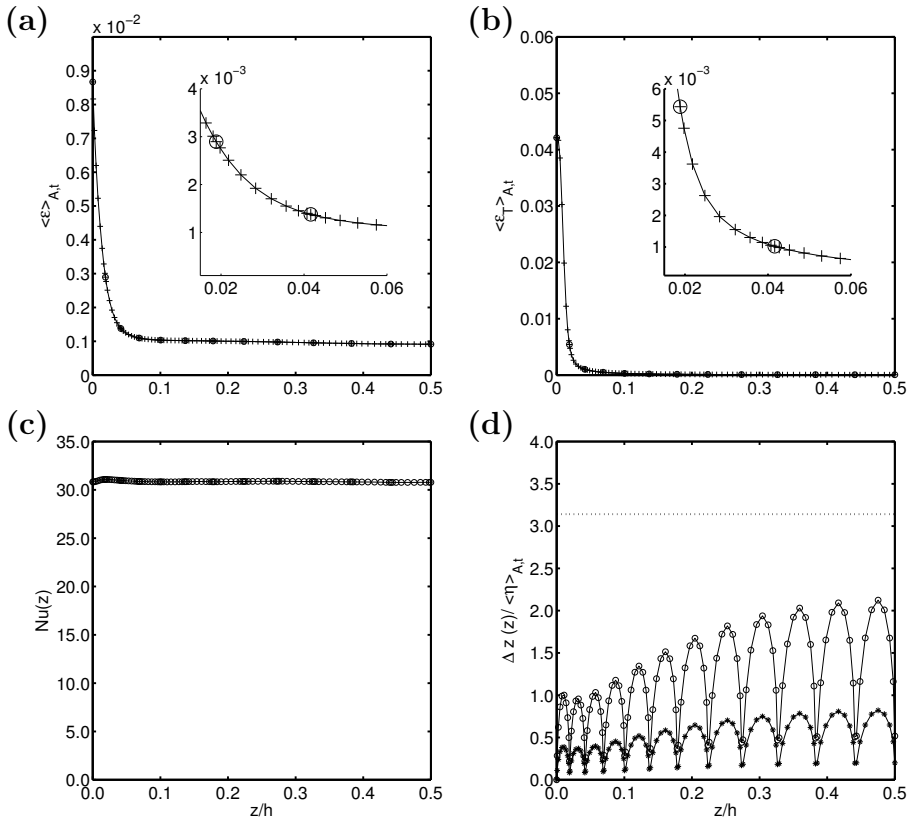


Figure 8: Resolution tests for this simulation: kinetic energy dissipation profile (a), thermal dissipation profile (b), horizontal plane Nusselt number profile (c), vertical length scales: Kolmogorov (stars) and Batchelor (circles) (d). All time averages were calculated from 205 instantaneous snapshots with 3 free-fall times between each snapshot. Open circles in (a) and (b) are placed at the element boundary locations.

The final resolution criterion that we wish to comment on is convergence of the Nusselt number which can be defined on individual planes (eq. 3.13 below), by volumetric averaging of the kinematic heat flux (eq. 3.14 below) , or by balancing dissipation inside the convection cell (eqs. 3.15 and 3.16 below) [20].

$< Nu(z) >_z \pm \sigma$	Nu_{bot}	Nu_{top}	Nu_{Q_o}	Nu_{ϵ}	Nu_{ϵ_T}
30.89 ± 0.08	30.83	30.85	30.87	30.83	30.12

Table 1: Nusselt number calculations in row order: average planar Nusselt number ($Nu(z)$) and the associated standard deviation, Nu at the bottom plate, Nu at the top plate, Nu from volume averaged kinematic heat flux, Nu from volume averaged kinematic dissipation, Nu from volume averaged thermal dissipation

$$Nu(z) = \sqrt{RaPr} < w\theta >_{A,t} - \frac{\partial < \theta >_{A,t}}{\partial z} \tag{3.13}$$

$$Nu_{Q_o} = 1 + \sqrt{RaPr} < w\theta >_{V,t} \tag{3.14}$$

$$Nu_{\epsilon} = 1 + \sqrt{RaPr} < \epsilon >_{V,t} \tag{3.15}$$

$$Nu_{\epsilon_T} = \sqrt{RaPr} < \epsilon_T >_{V,t} \tag{3.16}$$

$Nu(z)$ is plotted in the lower left panel of figure 8, and a summary of the different calculations of Nu is included in table 1.

These results show that we have good agreement between the different methodologies for computing Nu and that the majority of these methodologies give a Nu within 0.2% of each other. The only outlier is Nu_{ϵ_T} which is about 2.5% lower than the others.

3.3 Comparison with Experiments

This section compares two-point and single-point statistics computed from the DNS data with the corresponding quantities found in the experiments of Fernandes [10]. Quantitative comparisons of the velocity and temperature fields of RBC simulations and experiments have been rare to date, and we hope that the present comparison and discussion will provide useful guidelines for future studies. The agreement one should expect from this comparison is tempered by certain differences inherent to

DNS and experiment, despite our best efforts to numerically recreate the environment of Fernandes' experiment. First, DNS yields data from every grid point in the domain, but over a temporal duration that is limited by the cost of computation. In contrast, experiments yield data from a limited subset of points in the volume, but they can be performed over very long durations. These factors lead to different time averaging protocols. Second, differences between the data sampling domains and assumptions concerning horizontally homogeneous statistics lead to different spatial averaging protocols. Third, the experimental and DNS data were non-dimensionalized using different, albeit related scales for velocity and temperature. Fourth, although both flow fields occur in geometrically similar cylindrical domains with constant temperature horizontal boundaries, differences exist between the ideal, mathematically perfect boundary conditions of the DNS and the realistic, imperfect thermal boundary conditions of the experiment.

3.3.1 Differences between Experimental and Numerical Data Collection Protocols

In the following sections we discuss the foregoing differences and the steps taken to minimize their effects on the statistical comparison.

3.3.1.1 Differences in Temporal Averaging

Fernandes' [10] statistics were generated from a set of 300 statistically independent snapshots which were obtained over very long averaging times. These snapshots were collected in groups of 15 with $4z_*/w^*$ ($\sim$30-40 seconds) between each snapshot in a group ($z_* = 0.5h$ is half the layer depth, w^* is the Deardorff velocity scale precisely

Group	O(Ra)	Pr	Free-Fall Times	Γ
Kerr (1996) [15]	10^7	0.7	116	6.0
Shishkina & Thess (2009) [23]	$10^8 - 10^9$	4.38	290	1.0
Bailon-Cuba *et al.* (2010) [3]	$10^7 - 10^9$	0.7	$\sim 80 - 300$	0.5-12.0
Scheel et al. (2012) [19]	$10^5 - 10^8$	0.7	~ 480	1.0
Scheel et al. (2013) [20]	$10^6 - 10^9$	0.7	~ 75	1.0 & 3.0
Emran & Schumacher (2015) [9]	10^5	0.7-10	~ 600	50
Current Work	10^8	6.7	615	6.3

Table 2: A select list of RBC DNS studies. Total averaging times are listed in the 4th column.

defined later). The heat source was periodically turned off for 30 minutes to an hour between sets of snapshots, then turned back on and allowed to settle for 4-12 hours. This perturbing of the flow was done to keep the large-scale-motions from developing any preferential direction. The data between the perturbations was treated as independent realizations of the order of O(1000) free-fall times in duration and resulted in an averaging procedure that included temporal and ensemble averaging.

At each Ra the total procedure took hundreds of hours which is O(10^5) free-fall time units. Recreating a DNS data set that spans a similar amount of time and/or incorporates a similar number of realizations would be prohibitively expensive. For comparison, table 2 displays the total runtimes of a selection of the representative numerical simulations in RBC to illustrate this wide gap. The next section illustrates how the additional spatial data available through DNS can help narrow this gap.

3.3.1.2 Differences in Spatial Dimensions

As previously mentioned, we designed our DNS to match the experiments of Fernandes [10] as close as possible, and we define the spatial domain for both studies in cylindrical coordinates as Ω_F.

$$\Omega_F(r,\phi,z) : r \in 0 : 3.15h; \phi = 0 : 2\pi; z \in 0 : h \tag{3.17}$$

However, Fernandes did not utilize the entire volume for his velocity measurements. Instead, he chose to take measurements in the center of Ω_F far enough from the side walls so he could apply the assumption of horizontal homogeneity to the velocity statistics [10]. Fernandes obtained velocity measurements through the use of two-dimensional, particle image velocimetry (PIV) at a vertical plane in the center of the cell with a field of view 1.85 h in length and h in height. We shall refer to this as the experimental window Ω_E.

$$\Omega_E(r,\phi,z) : r \in 0 : 0.925h; \phi = 0, \pi; z \in 0 : h \tag{3.18}$$

One notable advantage that our DNS data set has over the experimental results of Fernandes is a third spatial dimension from which to draw data. Fernandes [10] applied horizontal-line averages on his 2D PIV data sets to examine the velocity statistics in the plane Ω_E. To utilize the extra dimension in the DNS results we defined a sub-volume within Ω_F that can be generated by rotating Ω_E about the central axis:

$$\Omega_{SV}(r,\phi,z) : r \in 0 : 0.925h; \phi = 0 : 2\pi; z \in 0 : h \tag{3.19}$$

We then projected the instantaneous DNS results within Ω_{SV} onto evenly spaced points in cylindrical coordinates, and performed horizontal, planar averages to the data to mimic the line-averaging performed by Fernandes. For example, the mean vertical velocity averaging procedure we used in Ω_{SV} is defined as follows:

$$< w(z) >_{A,t} = \frac{1}{N_T N_P} \sum_{i=1}^{N_T} \sum_{j=1}^{N_P} \frac{2r_j}{R} w(r_j, \phi_j, z_j, t_i) \tag{3.20}$$

where N_T is the number of instantaneous snapshots in time, N_P is the number of points in the plane, the subscript A stands for the "area-averaged", and $R = 0.925\,h$

is the radius of a circular cross-section of Ω_{SV}. We also collected statistics along horizontal planes that spanned the full DNS domain (Ω_F) using numerical averaging procedures native to Nek5000.

Another notable difference in the DNS and experimental data sets is in the horizontal velocity components. The DNS calculations were performed in cartesian coordinates where u and v correspond to the x and y axis. However, the domain is cylindrical and so cartesian velocity components do not properly reflect the periodicity found in the cylindrical domain. Additionally, since Fernandes collected 2D velocity fields in a plane intersecting the central axis of Ω_{SV} the horizontal component in his data sets can be interpreted as a radial velocity component. Based on these two observations we chose to compare radial velocity statistics from the DNS with the horizontal velocity statistics Fernandes published. The DNS radial velocity component (u_r) was extracted from the two horizontal velocity components using the simple transformation in eq. 3.21.

$$\phi = \tan^{-1}(x/y)$$
$$u_r(x,y,z,t) = u(x,y,z,t)\cos\phi + v(x,y,z,t)\sin\phi \tag{3.21}$$

In the remainder of the paper, radial velocity will be implied for our DNS results whenever horizontal velocity is referred to.

3.3.1.3 Differences in Scaling

It should be noted that Fernandes non-dimensionalized variables using the Deardorff scales of velocity and temperature [7], and a vertical length equal to one-half the layer depth ($z_* = 0.5h$). The excellent scaling properties of Deardorff's scales have been shown to collapse, to within $\pm 15\%$, the profiles of second and third order turbulence

moments measured in laboratory experiments at $Pr = 6.8$, $Ra = 10^7$ onto atmospheric measurements of convective boundary layers at $Pr = 0.7$ and $Ra \sim 10^{14} - 10^{16}$ by Adrian, Ferreira and Boberg [1]. Correlation of the scaled data over such a huge range is testimony to the power of this scaling. Moreover, the non-dimensional variables are O(1), indicating that the scales themselves are representative of the physical variables. The Deardorff velocity and temperature scales are defined below, where Q_o is the kinematic heat flux whose units are $[K \cdot mm/s]$. Q_o can be determined by dividing the standard heat flux at the wall (H_o $[W/mm^2]$) by the density (ρ $[kg/mm^3]$) and specific heat (c_p $[J/(kg \cdot K)]$).

$$Q_o = H_o/(\rho c_p) \tag{3.22}$$

$$w^* = (\beta g Q_o h)^{1/3} \tag{3.23}$$

$$\theta^* = \frac{Q_o}{w^*} \tag{3.24}$$

However, the DNS procedure used the "free-fall" scales during the calculations because they provide better accommodation for the constant temperature boundary conditions. To provide a one-to-one comparison of velocity statistics with Fernandes and to take advantage of the compelling properties of Deardorff scales discussed above, all computational velocity data presented in this paper was rescaled to the Deardorff scales. This was accomplished by multiplying the DNS field by w_f and then dividing by w^*. w^* was determined by calculating Nu in the simulation and then scaling it against Fernandes values of Nu and w^* at this Ra (9.6×10^7). These were $Nu = 27.9$ and $w^* = 2.5mm/s$ respectively [10].

The simulated value of Nu for $Ra = 9.6 \times 10^7$ (see table 1) was approximately 10% higher and lead to $w^* = 2.6mm/s$. Our simulation shows well converged results (see Section 2.1), and Fernandes demonstrated that heat loss through the side walls

was kept to under 2% for his experiments [10], so it seems that the usual suspects of in sufficient numerical resolution and heat loss through the side walls are not to blame. One possible culprit for the discrepancy in Nu is the differences between experimental and numerical boundary conditions. The no-slip velocity boundary conditions were identical between the two cases, but the thermal boundary conditions were not. The numerical simulation explicitly enforced constant temperature boundaries while the experiment maintained an average constant heat flux through resistive heaters. Additionally, the simulation provided sidewalls that are perfectly insulated while the experiment had sidewalls with finite thermal conductivity.

3.3.2 Statistics Results

3.3.2.1 Two-Point Statistics

In this section the two-point, correlation functions of the velocity field for vertical separation are compared against the published results of Fernandes in figure 9. This statistic is a useful mechanism for analyzing similarity in the spatial structure of the velocity field, and it is a good metric to judge how the organization of our DNS velocity field compares with the experiment's.

Visual inspection of figure 9 shows the contour shapes and amplitudes between the results are qualitatively in good agreement. This indicates that the large-scale velocity structure in the numerical field matches the experiment within the specified sub-volume Ω_{SV}. The two-point correlation does not tell us that the velocity structures have the same azimuthal orientation or vertical direction, but rather that their shape and size are very similar. We note that in the DNS contours the horizontal velocity

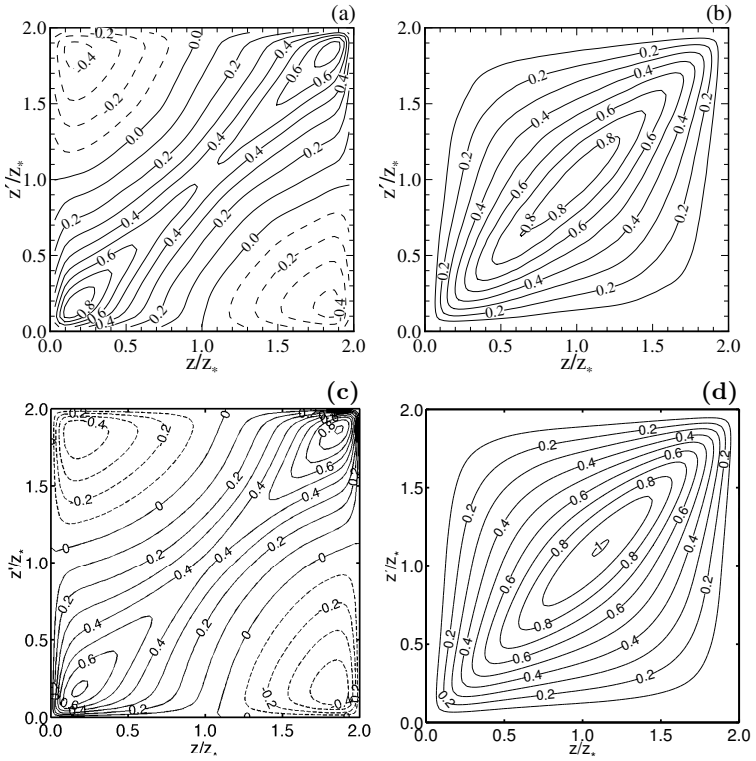


Figure 9: Contour plot of the vertical, two-point correlation of the u-velocity component (left) $< u_r(z)u_r(z') >_{A,t}$ and w-velocity component $< w(z)w(z') >_{A,t}$(right), all values normalized by $w*^2$: (a) and (b) experimental results at $Ra = 2 \times 10^8$ (averaged within Ω_E)($Ra = 2 \times 10^8$ is the closest Ra to our simulations, for which two-point correlation is presented in the experiments), (c) and (d) DNS at $Ra = 9.6 \times 10^7$ (averaged within Ω_{SV})

correlations are slightly larger at the vertex (2,2) and the experimental results are slightly larger around the origin (0,0). We also observe a similarity in the shape of the contours at the opposing corners in the experimental and DNS plots. This indicates that the DNS' horizontal velocity structure within Ω_{SV} is similar to a reflection across the mid-plane of the velocity structure observed in the experiments. A more quantitive comparison of velocity statistics is provided through the single-point statistics in the next section.

3.3.2.2 Single-Point Statistics

In this section the single-point vertical and horizontal velocity statistics as well as the temperature fluctuations are compared against those of Fernandes [10]. Separate profiles are presented for DNS statistics that are averaged on horizontal planes that span the sub-volume Ω_{SV} and the entire volume Ω_F. We also note that the r.m.s velocity profiles in this section are the full velocity components, and not the fluctuations. This was done to match the results of Fernandes [10] who assumed a zero-mean velocity field for his r.m.s. calculations.

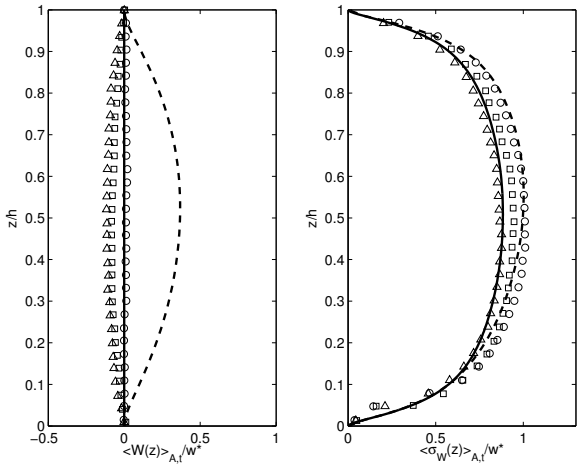


Figure 10: Ensemble and horizontally averaged mean (left) and r.m.s. (right) vertical velocity profile normalized by $w*$: DNS: $Ra = 9.6 \times 10^7$, averaging domain Ω_F (–) and averaging domain Ω_{SV} (- -); Experiment: $Ra = 6 \times 10^7$ ($\triangle$), $Ra = 2 \times 10^8$ ($\square$), $Ra = 1 \times 10^9$ ($\circ$)

Figure 10 displays the mean and r.m.s. profiles for the vertical velocity component. The r.m.s vertical velocity profile within Ω_{SV} over predicts the magnitude by $\sim 11\%$ when compared to the experimental results. This over prediction is based on a pointwise comparison between the numerical results and a profile interpolated between

the experimental data sets at $Ra = 5 \times 10^7$ and 2×10^8 to the DNS Ra number. Additionally, a notable bias is present in the upper half of the numeric profile. However, when the horizontal averaging is extended to span all of Ω_F the profile's symmetry about the mid-plane improves and the profile's magnitude reduces to within 5% of the expected value. A similar trend is seen in the mean vertical velocity profiles.

The mean vertical velocity profile from sampling points within Ω_{SV} is dramatically larger than if averaged over the full domain Ω_F (which is identically zero as expected). It is also larger than the experimental profiles which were averaged over Ω_E, and it has a sign difference. The difference in sign between the experimental and numerical results we attribute to the lack of preferred direction in RBC systems. There is nothing in the geometry, boundary conditions or governing equations to give preference to a positive or negative vertical velocity making up-drafts and down drafts of equal probability in the LSC. The sign difference between the experimental and numerical profiles suggests that while our DNS results clearly indicate a central up-draft, a central down-draft has likely prevailed in the experiments. This sign difference is also consistent with the reflection of structure we observed in the two-point correlations for the horizontal velocity components (figure 9).

The trends in r.m.s. horizontal velocity profiles in figure 11 are similar to the trends in vertical velocity profiles. The r.m.s profile from within Ω_{SV} is slightly larger than the profile averaged across the entire domain, but both of these profiles are very close to the experimental results at similar Ra. We also see a continuation of the reflection for the horizontal velocity structure that was observed in the two-point statistics. The experimental profiles have slightly larger peaks near the lower boundary, and the numerical profiles have larger peaks toward the top boundary. This follows the trend

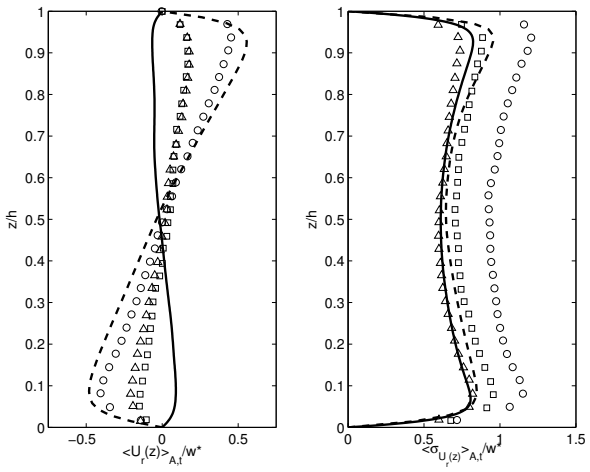


Figure 11: Ensemble and horizontally averaged mean (left) and r.m.s. (right) horizontal velocity profiles normalized by $w*$: DNS: $Ra = 9.6 \times 10^7$, averaging domain Ω_F (–) and averaging domain Ω_{SV} (- -); Experiment: $Ra = 6 \times 10^7$ ($\triangle$), $Ra = 2 \times 10^8$ ($\square$), $Ra = 1 \times 10^9$ ($\circ$)

seen in figure 9 and supports our observations regarding the vertical profiles displayed in figure 10.

The mean horizontal velocity profiles have a slightly different behavior than their vertical velocity counterparts. Inside Ω_{SV} the mean takes a similar shape and direction when compared to the experimental profiles. The mean inside Ω_{SV} is notably larger in magnitude than the expected profile that lies between experimental profiles at $Ra = 5 \times 10^7$ and 2×10^8 in the upper half of the domain ($z/h \geq 0.5$), and the overshoot is similar in magnitude to the mean vertical velocity profile in figure 10. When the horizontal averaging is extended to span Ω_F the mean horizontal profile changes direction and substantially reduces in magnitude. We attribute this change to the organization of the LSC in the moderately large Γ domain. This structure will be reviewed extensively in the next section.

Finally, we present statistics from the temperature field in figure 12. Fernandes

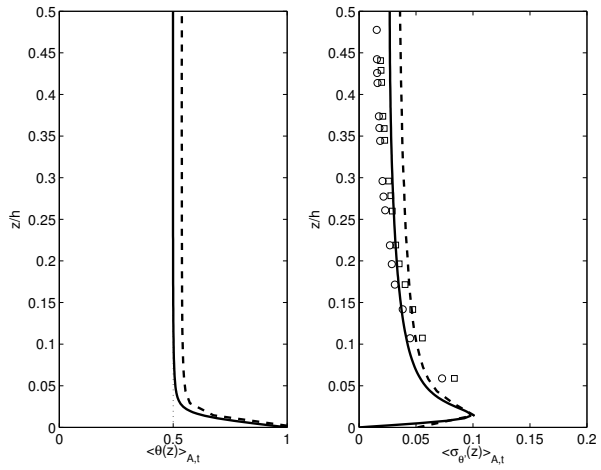


Figure 12: Ensemble and horizontally averaged mean temperature profile (left) and r.m.s temperature fluctuations (right) normalized by ΔT: DNS: $Ra = 9.6 \times 10^7$, averaging domain Ω_F (–) and averaging domain Ω_{SV} (- -); Experiment: $Ra = 2 \times 10^8$ ($\square$), $Ra = 5 \times 10^8$ ($\circ$)

didn't provide mean temperature profiles to compare against, but the slightly higher bulk temperature in the Ω_{SV} mean profile is consistent with the mean, vertical velocity profile from figure 10. This indicates a net updraft of warm fluid within Ω_{SV}. When the horizontal averaging is extended to the entire domain the mean profile's bulk temperature value returns to the expected value of 0.5 outside of the thermal boundary layer. The r.m.s. profiles for the temperature fluctuations are consistent with this interpretation, and the simulated r.m.s. profiles display slightly larger magnitudes than experimental profiles in the bulk of the flow, corresponding to a preferential central up-draft (higher temperatures) in the simulations as opposed to a probable slight down-draft (lower-temperatures) in the experiments.

Overall, we find that the statistics from our simulation compare well with the experimental results of Fernandes. We see good agreement in shape, direction, and magnitude for the various profiles despite only having a single realization of the flow field and a much smaller averaging time. The profiles tend to have larger magnitudes

when they are constructed from averages within Ω_{SV}. However, when the averaging domain is increased to span Ω_F the profiles are, for the most part, within $\pm \sim 5\%$ of the experimental results.

3.4 Structure of the Flow

The remainder of the paper focuses on analyzing the large-scale thermal structures and their relationship to the pattern of the large-scale circulation. Figure 13 displays several cut planes of the instantaneous temperature field. The color scale of the temperature field spans $\pm 5\%$ of ΔT to highlight the structures within the bulk region where the large-scale structures reside.

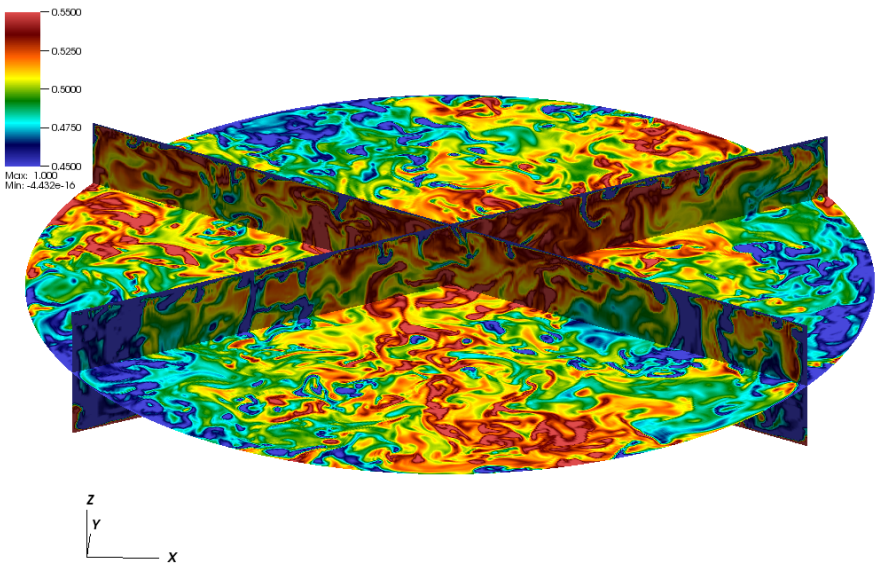


Figure 13: Instantaneous temperature data scaled between $0.45 - 0.55\Delta T$ at the horizontal cut plane at $z/h = 0.5$ and vertical cut planes at 30°, 150° from the x-axis

Large concentrated areas of warm and cool fluid can be discerned in the instantaneous temperature field on the horizontal plane in figure 13. The vertical planes in figure 13

show that the warm and cool regions consist of concentrations of smaller structures (plumes and sheets) crossing the bulk layer. The mid-plane is a favorable location to investigate structures that span the entire layer depth because the mean horizontal velocity components are minimal, and the up (warm thermal) and down (cool thermal) motions have comparable strength due to symmetry about the mid-plane.

Figure 14 shows a more detailed view of the mid-plane to better illustrate the concentrated thermal areas. The left panel is from the same time instance displayed in figure 13, and the right panel displays results 5.9 t_ϵ (117 free-fall times) later to illustrate that these thermal concentrations can be observed for extended periods of time. The similarities between these two instances suggest that an underlying large-scale structure exists, but it's hard to discern the structures precise form with such a high level of noise from the back ground turbulence.

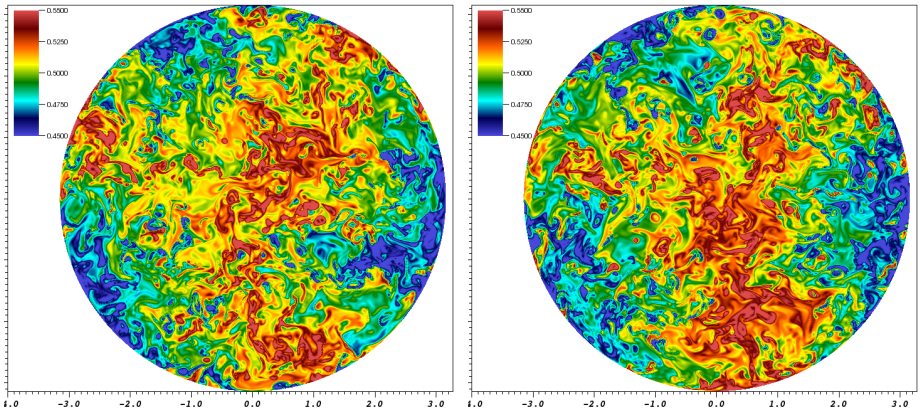


Figure 14: Instantaneous temperature data in the $x-y$ plane at z/h=0.5 for two different time instances $5.9t_\epsilon$ apart.

3.5 LSC Extraction by Smoothing in Time

In the following section we use running, temporal averages to filter noise from the flow field and reveal properties about the large-scale structures for relatively large Γ, turbulent RBC. This averaging procedure smooths the signal and it is defined as:

$$< u(x,y,z) >_t = \frac{1}{T}\int_{t_o}^{t_o+T} u(x,y,z,t)dt \tag{3.25}$$

where T is the averaging period or filter width, and t_o is the starting point in time for each filtering operation. In this section we will use the terms "averaging," "filtering" and "smoothing" interchangeably since the purpose of our running average procedure is to remove small scale fluctuations and preserve the larger scale structures who have longer life cycles.

3.5.1 Short-Time Filtering (T=1 t_ϵ)

When the temperature field is averaged over a period of 1 t_ϵ seven discrete thermal concentrations begin to emerge. These large-scale thermal structures can be identified by studying the left panel of figure 15. Six of the seven thermals are connected to the cell's side walls, and are alternately arranged by temperature. The seventh thermal is a warm updraft located in the central region of the cell, but noticeably off center. A significant portion of the central thermal resides within the volume where we collected the sub-domain statistics (figures 9-12). This central thermal structure partially explains the differences between statistics in the sub-domain (Ω_{SV}) and those that spanned the entire domain (Ω_F) because it would create a bias for warm fluid and up flow within Ω_{SV}.

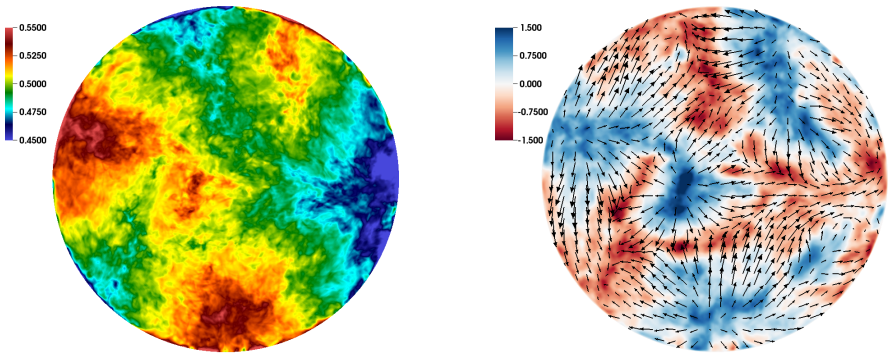


Figure 15: Thermal field at the mid-plane scaled from $0.45 - 0.55\Delta T$ (left); vertical velocity field at the mid-plane scaled between -1.5 & 1.5 w* with velocity vectors in the $x - y$ plane at z/h=0.92 superimposed (right); all results are time averaged over the 1st t_ϵ of data

Coupling between the large-scale thermal field and the velocity roll-cells is illustrated by comparing the spatial locations of the mean velocity up/down drafts and the vector field connecting these drafts. The right panel in figure 15 displays the vertical velocity component at the mid-plane to illustrate the location of the mean vertical drafts, and a vector field at the plane $z = 0.92h$, corresponding to the local maxima in the mean horizontal velocity profile (figure 11), to show how the drafts are interconnected by flow near the boundaries. A comparison of the left and right panels in figure 15 clearly shows that the warm thermals represent velocity sources and the cool thermals represent velocity sinks from the top-down perspective. Connecting the warm and cool thermals creates a complete roll-cell. This basic method can be used to conceptualize the 3D multi-roll-cell pattern in a simple and intuitive manner.

The short-time-averaged temperature fields in figure 16 can be used to interpret the fluctuation level still present in the large-scale structures after the short-time averaging. These instances were taken 3 t_ϵ apart to increase the dynamic effect through static visualizations. Very little change is observed in the thermal field which shows

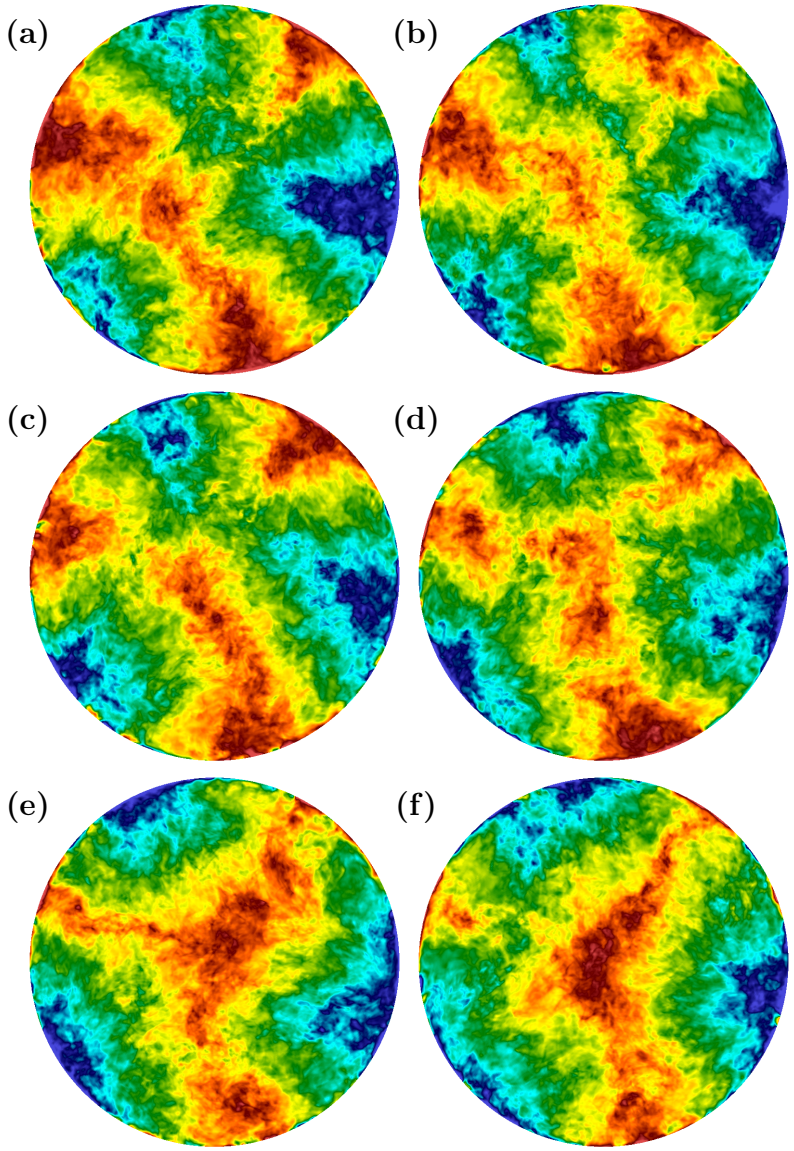


Figure 16: Temperature field averaged over one t_ϵ at $t_o = 2\ t_\epsilon$(a), 5 t_ϵ(b), 8 t_ϵ(c), 11 t_ϵ(d), 14 t_ϵ(e), and 17 t_ϵ(f) in the $x-y$ plane at $z/h = 0.5$ scaled between 0.45-0.55ΔT, the color panel is the same as in figure 8 (left)

that the large-scale thermals are relatively stationary over the averaging time of this data set. The up and down drafts near the walls show little change in spatial location, however there is some noticeable fluctuation in their individual sizes.

Qualitatively, we observe the central thermal in figure 16 moving from left to right during the time series. It is difficult to be certain if the structure is approaching a steady state, or if a secondary fluctuation with even slower dynamics is dominating the flow. However, we can be certain that the smoothing from this short time average does not bring the field to a steady state, and that a pattern of seven large-scale thermals is persistent across the entire range of our data set.

3.5.2 Medium-Time Filtering (T=10 t_ϵ)

In this section we increase our averaging time to 10 t_ϵ (300 free-fall times) to smooth out the fluctuations that were still present after short-time averaging. This "medium" averaging time is effectively half of our data set resulting in two instances to observe, see figure 17.

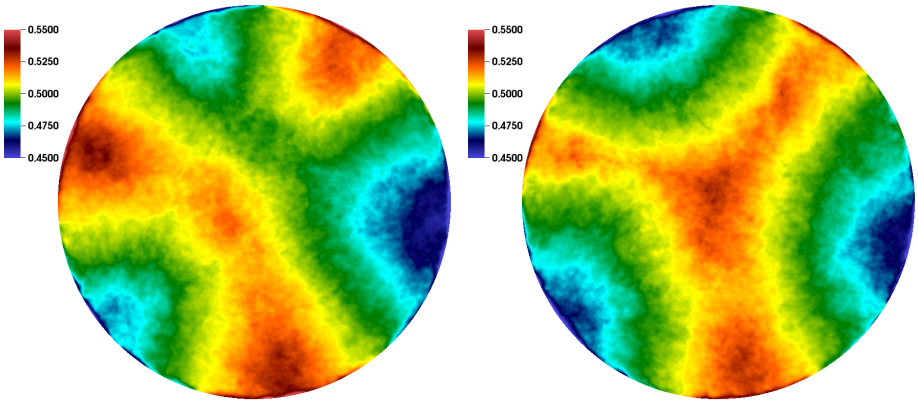


Figure 17: Temperature field averaged over 0-10 t_ϵ (left) and 10-20 t_ϵ (right) in the $x-y$ plane at $z/h = 0.5$ scaled between 0.45-0.55ΔT

The results in figure 17 show a striking set of triangular patterns for the large-scale thermal field that are manifest at the mid-plane. The seven discrete thermals that were identified in the previous section have clearer boundaries and are now visibly

obvious. From figure 17 we see the emergence of a dominant low order mode with 120°, azimuthal periodicity. There are several interesting departures from pure periodicity that can be observed in medium-time average data such as the central thermal residing off the central axis and the size variation amongst the thermals along the outer wall. The significant change in the central thermal location between the two panels in figure 17 illustrates that the central plume still moves around the center at these time scales.

However, the large-scale organization that is revealed through 10 t_ϵ of temporal smoothing is similar for both instances. Figure 18 illustrates how the large-scale organization can be interpreted as a hub and spoke pattern with roll-cells forming between alternating thermals around the outer wall and vortex lines located between the thermals. The central, uprising thermal creates in-plane, azimuthal vorticity due to the shearing effects, manifested as a circular vortex ring around it. This vorticity acts to reinforce the interaction of the large-scale thermal with the side thermals of opposing temperature and to strengthen the global large scale motions observed in the simulations.

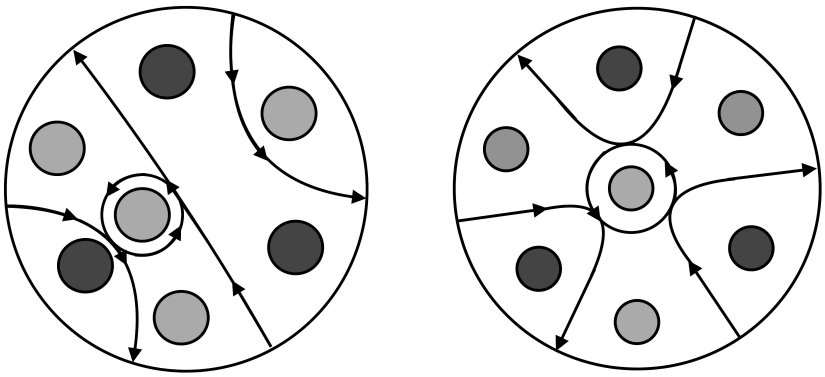


Figure 18: Conceptual diagrams of the large-scale organization in the flow field averaged over the 0-10 t_ϵ (left) and 10-20 t_ϵ (right). The light circles represent updrafts, the dark circles represent downdrafts and the vectors represent vortex lines.

The right panels in figures 17 and 18 represent a symmetric pattern, the one which

will also likely persist over longer averaging times. The left panels in figures 17 and 18 illustrate the strong effect the central thermal can have on the velocity structure. When the location of the central thermal moves off the center axis it comes closer to two other thermals with similar temperatures. This shift breaks some of the symmetry, enhances the interaction between the central thermal and the nearest thermals of opposing temperature, and can cause the other two vortex lines to break and reform as drawn in the left panel of figure 18.

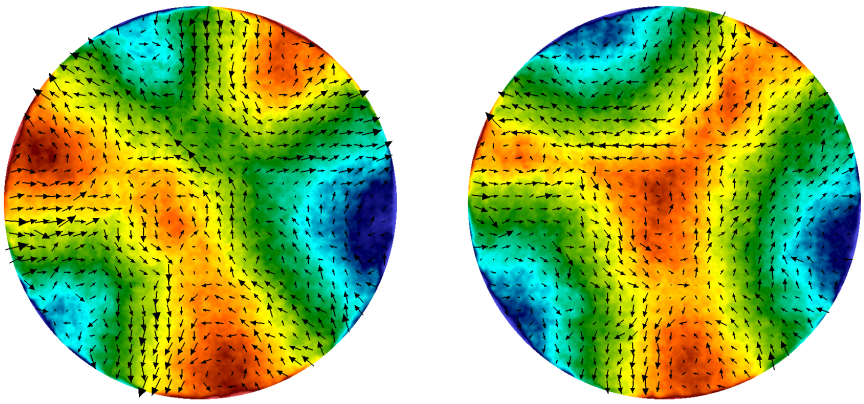


Figure 19: In plane vorticity plotted on top of temperature in the flow field averaged over the 0-10 t_ϵ (left) and 10-20 t_ϵ (right) in the $x-y$ plane at $z/h = 0.5$ (temperature scaled between 0.45-0.55ΔT, the color panel is the same as in figure 10)

To test the validity of the concepts presented in figure 18 we computed the in-plane vorticity at the mid-plane for each of these realizations and the results are displayed in figure 19. The lines of vorticity in figure 19 verify that the spokes in the temperature field between large-scale thermals are indeed regions of strong vorticity, and that the central thermal has a vortex encircling it which interacts with the spokes. These similarities give weight to the conceptual basis of the right hand diagram in figure 18. However, the conceptual diagram fails to replicate the fact that the warm regions are narrower than the cool regions. This can be remedied by recognizing that the width

of the vortex loops need not be equal, which would amount to squeezing of the vortex lines closer to the warm thermals (because the central thermal is an updraft) that reside in the outer region of the convection cell, in the right diagram of figure 18.

It is recognized that the pattern of large scale motions is most likely a function of Γ and that additional patterns may exist at other Γ's and in other domains. However, there is a strong resemblance between the pattern in the left panel of figure 17 and the pattern shown by Bailon-Cuba *et al.* [3] at similar values of Γ and Ra even though Pr was smaller by a factor of 10. During the initialization stage of this simulation we incrementally increased Ra and allowed the initial transients to settle after each discrete jump in Ra. During this process we qualitatively observed a transition in the large-scale thermals organization from a pattern that is more indicative of a single roll cell to the multi-roll cell state we have presented. From this we speculate that the characteristic size of the large-scale thermals will vary with Ra; however, this concept warrants a more thorough investigation.

3.5.3 Long-Time Filtering (T=20 t_ϵ)

We conclude our investigation of the large-scales structures at moderately large Γ by evaluating the full time averaged field from our DNS data set. Figure 20 provides a time average over the complete 20 t_ϵ. Figure 20(a) displays multiple cut planes of the average field to show the level of symmetry of the structure. The angles of the 3 vertical cut planes were chosen in an attempt to maximize intersection between large-scale thermals that are positioned directly across from one another in the cell. We observe that thermals directly across from one another are not perfectly aligned. Careful observation of figure 20(b) will show that the central thermal is still not

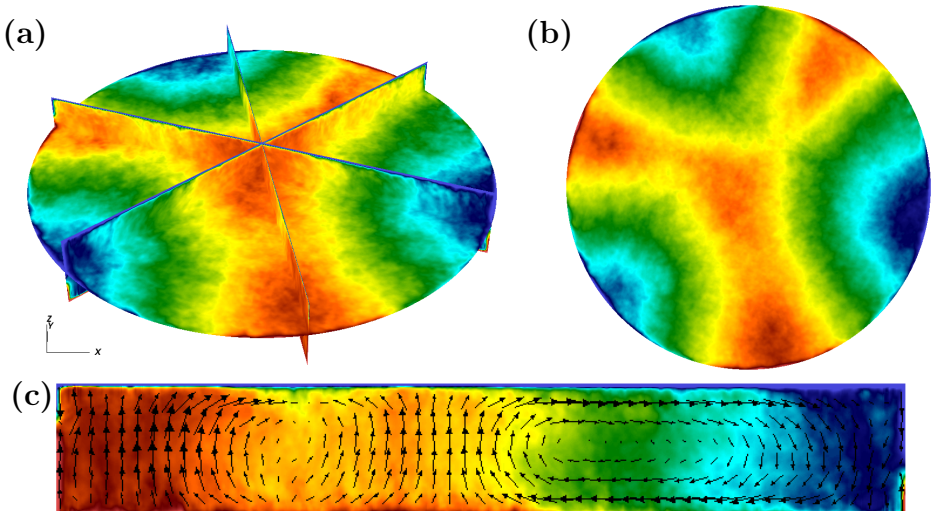


Figure 20: Images of the temperature field scaled between 0.45-0.55ΔT (the color panel is the same as in figure 10) averaged over 0-20 t_ϵ: a horizontal cut plane at $z/h = 0.5$ and vertical cut planes at 40°, 100°, and 160° from the x-axis (a), detailed view of the horizontal cut plane at $z/h = 0.5$ (b), and detailed view of the vertical cut plane 160° from the x-axis with the average, in-plane velocity vectors superimposed (c).

completely centered in the cell, and that the outer thermals are still not uniform in size. However, the dominance of the 120° mode can't be denied when inspecting these figures.

One of the cell's vertical cross-sections can be seen in detail in figure 20(c) and the in-plane, temporally-averaged, velocity vectors are superimposed onto the scalar temperature field. Here we see a single elongated roll-cell on the right side of the cross-section with a width approximately equal to the radius of the domain. On the left side of the cross-section two updrafts exist side-by-side. These two updrafts are able to exist in such close proximity because the flow is able to redirect itself toward the cool thermals on either side, thus establishing the spoke-like vortices illustrated in figure 18. Virtually all of the flow from the thermal on the far left wall of figure 20(c) must contribute to these spoke like structures while the central thermal also contributes

toward the roll-cell in the right half of the figure. These two updrafts display signs of a small recirculation region directly between each other, and we suspect that this interaction plays a significant role in perturbing and relocating the central, large-scale thermal. However, further investigation into this phenomenon would require a longer time series.

3.6 Summary and Conclusions

Large scale circulations are known to exist in low aspect-ratio ($\Gamma < 2$) experiments having rectangular and cylindrical shapes [27, 5]. Experiments and DNS in large aspect ratio domains at Ra above transition [16, 25, 3] have shown that the planform pattern of the convective cells consists of random 3,4,5,6-sided polygons. This DNS study was conducted to explore the structural patterns that characterize RBC in a circular domain of moderate aspect ratio by performing highly resolved simulations in a circular domain of aspect ratio 6.3 at Pr=6.7 and Ra=9.6×10^7. The comparison of several resolution criteria in our simulations with the recommended values for RBC confirms that our study is well resolved [3], [12], [20].

The referred geometry and set of parameter values were chosen to allow direct comparison with a PIV experiment [10] in which many different states of the convection were sampled over several independent realizations, with at least 100 eddy turnover duration per realization, and in which line averaging and time averaging could be done to improve statistical convergence. In the DNS convergence was improved by averaging over time and horizontal planes. In lieu of the absence of a similar comparison in the previously documented numerical studies of Rayleigh-Bénard convection, the current comparison carries a substantial value. We show that good agreement in temperature

and velocity statistics can be obtained between DNS and experimental studies in spite of a large difference in averaging times. In this study we also elucidate on the methodologies allowing for a meaningful comparison between DNS and experimental data obtained with rather different data collection techniques.

Comparison with the experiment confirmed the validity of the DNS with accuracy of 5% for the temperature and vertical velocity statistics when the statistics were averaged over horizontal planes that spanned the entire domain. The velocity field averaged within a central sub-volume was more sensitive, even showing mean vertical velocity of sign opposite to the experiment, indicating an effect of the large scale structures.

The large-scale thermals for this particular configuration organized into a pattern with a high level of symmetry in the azimuthal direction. The pattern of the dominant mode is a 120 degree periodic arrangement of radially oriented up and down motions caused by roll cells that extend across the depth of the domain. This spoke-shaped pattern of period-three persists over the entire duration of the simulation, 615 free-fall times. These patterns are very similar to the ones observed by Bailon-Cuba *et al.* [3] at a similar Γ and Ra despite the difference in Pr.

Instantaneously the spoke pattern is imbedded in small scale turbulence, but still not totally obscured. To extract the spoke pattern from the full turbulent field temporal smoothing over different time-scales was employed. The structure clearly becomes 3 pairs of alternating up and down motions plus a hot rising column in the center of the domain. This pattern persists even after smoothing over 20 t_ϵ, the entire run time of the simulation. The persistence of the azimuthal orientation implies that reorientations of the spokes occur on extremely slow time scales such that in any single experiment or DNS of moderate duration, the underlying circulation biases the

results. Moreover, the persistence of the direction of the central column indicated that the pattern does not "flip" during the DNS, a condition that is needed to sample all of the states of the RBC.

BIBLIOGRAPHY

[1] R. J. Adrian, R. T. D. S. Ferreira, and T. Boberg. "Turbulent thermal convection in wide horizontal fluid layers". In: *Experiments in Fluids* 4 (1986), pp. 121–141.

[2] Guenter Ahlers, Siegfried Grossmann, and Detlef Lohse. "Heat transfer and large scale dynamics in turbulent Rayleigh-Benard convection". In: *Reviews of Modern Physics* 81.2 (2009), pp. 503–537.

[3] J. Bailon-Cuba, M. S. Emran, and J. Schumacher. "Aspect ratio dependence of heat transfer and large-scale flow in turbulent convection". In: *Journal of Fluid Mechanics* 655 (July 2010), pp. 152–173. ISSN: 1469-7645. DOI: 10.1017/S0022112010000820.

[4] Eberhard Bodenschatz, Werner Pesch, and Guenter Ahlers. "Recent developments in Rayleigh-Bénard convection". In: *Annual review of fluid mechanics* 32.1 (2000), pp. 709–778.

[5] E. Brown, A. Nikolaenko, and G. Ahlers. "Reorientation of the Large-Scale Circulations in Turbulent Rayleigh-Benard Convection". In: *Physical Review Letters* 95.084503 (2005).

[6] Laltu Chandra and Günther Grötzbach. "Analysis and modelling of the turbulent diffusion of turbulent heat fluxes in natural convection". In: *International Journal of Heat and Fluid Flow* 29.3 (2008), pp. 743–751.

[7] James W Deardorff. "Convective velocity and temperature scales for the unstable planetary boundary layer and for Rayleigh convection". In: *Journal of the Atmospheric Sciences* 27.8 (1970), pp. 1211–1213.

[8] Michel O Deville, Paul F Fischer, and Ernest H Mund. *High-order methods for incompressible fluid flow*. Vol. 9. Cambridge University Press, 2002.

[9] Mohammad S Emran and Jörg Schumacher. "Large-scale mean patterns in turbulent convection". In: *Journal of Fluid Mechanics* 776 (2015), pp. 96–108.

[10] R. L. Fernandes. "The spatial structure of turbulent Rayleigh-Benard convection". PhD thesis. Urbana, IL: University of Illinois, 2001.

[11] Paul F Fischer, James W Lottes, and Stefan G Kerkemeier. "nek5000 Web page". In: *Web page: http://nek5000. mcs. anl. gov* (2008).

[12] Günther Grötzbach. "Spatial resolution requirements for direct numerical simulation of the Rayleigh-Bénard convection". In: *Journal of Computational Physics* 49.2 (1983), pp. 241–264.

[13] S Kenjereš and K Hanjalić. "LES, T-RANS and hybrid simulations of thermal convection at high Ra numbers". In: *International journal of heat and fluid flow* 27.5 (2006), pp. 800–810.

[14] S Kenjereš and K Hanjalić. "Transient analysis of Rayleigh–Bénard convection with a RANS model". In: *International journal of heat and fluid flow* 20.3 (1999), pp. 329–340.

[15] Robert M Kerr. “Rayleigh number scaling in numerical convection”. In: *Journal of Fluid Mechanics* 310 (1996), pp. 139–179.

[16] Ronald du Puits, Christian Resagk, and André Thess. “Breakdown of wind in turbulent thermal convection”. In: *Physical Review E* 75.1 (2007), p. 016302.

[17] B. A. Puthenveettil and J. Arakeri. “Plume strucutre in high Rayleigh-number convection”. In: *Journal of Fluid Mechanics* 542 (2005), pp. 217–249.

[18] X-L Qiu and P Tong. “Large-scale velocity structures in turbulent thermal convection”. In: *Physical Review E* 64.3 (2001), p. 036304.

[19] J. D. Scheel, E. Kim, and K. R. White. “Thermal and viscous boundary layers in turbulent Rayleigh-Benard convection”. In: *Journal of Fluid Mechanics* 711 (2012), pp. 281–305.

[20] Janet D Scheel, Mohammad S Emran, and Jörg Schumacher. “Resolving the fine-scale structure in turbulent Rayleigh–Bénard convection”. In: *New Journal of Physics* 15.11 (2013), p. 113063.

[21] M. J. Shelly and M. Vinson. “Coherent structures on a boundary layer in Rayleigh-Benard turbulence”. In: *Nonlinearity* 5 (1992), pp. 323–351.

[22] O. Shishkina and C. Wagner. “Analysis of sheet-like thermal plumes in turbulent Rayleigh-Benard convection”. In: *Journal of Fluid Mechanics* 599 (2007), pp. 383–404.

[23] Olga Shishkina and André Thess. "Mean temperature profiles in turbulent Rayleigh–Bénard convection of water". In: *Journal of Fluid Mechanics* 633 (2009), pp. 449–460.

[24] DR Wilson, TJ Craft, and H Iacovides. "Application of RANS turbulence closure models to flows subjected to electromagnetic and buoyancy forces". In: *International Journal of Heat and Fluid Flow* 49 (2014), pp. 80–90.

[25] Ke-Qing Xia, Chao Sun, and Yin-Har Cheung. "Large scale velocity structures in turbulent thermal convection with widely varying aspect ratio". In: *Proc. 14th Int. Symp. on Applications of Laser Techniques to Fluid Mechanics*. 2008.

[26] Claudia Zimmermann and Rodion Groll. "Computational investigation of thermal boundary layers in a turbulent Rayleigh–Bénard problem". In: *International Journal of Heat and Fluid Flow* 54 (2015), pp. 276–291.

[27] G. Zocchi, E. Moses, and A. Libchaber. "Coherent structures in turbulent convection, an experimental study". In: *Physica A* 166 (1990), pp. 387–407.

Chapter 4

MITIGATING THE INFLUENCE OF VERY LONG-LIVED STATISTICS TO IMPROVE STATISTICAL CONVERGENCE IN FINITE-TIME SIMULATION OF RAYLEIGH-BÉNARD CONVECTION

4.1 Introduction

A hallmark of turbulent flow is the random, chaotic spatial structure in the instantaneous flow field. Within the chaos the field also contains recurrent organized motions that are temporally coherent on various time scales ranging from very short to very long. Over an infinite-time turbulent flow manifests an infinite number of states, where a state in this sense is any spatial organization of the flow's primary variables i.e. velocity, pressure, and scalar fields. In addition to observations of instantaneous states and the evolution and interaction of motions contained therein, studies of turbulence also rely heavily on statistical averaging. Most theories of turbulence assume the flow is statistically stationary so that averages over infinite times converge to the ensemble average of an infinite number of random realizations. This makes the infinite-time average calculable in principle. In experiments and numerical simulations the infinite-time average is unreachable, and time averages over finite times often fail to converge well. Supplemental spatial averages over regions of homogeneous statistics or supplemental ensemble averages over additional realizations are often invoked to improve convergence of the finite-time average.

The majority of the turbulent flows contain some kind of large-scale, seemingly chaotic turbulent motion, and in some flows these large-scale motions organize into

very-large-scale motions that evolve on extremely large time scales compared to the viscous time scales of the smallest eddies [45, 8]. One example is turbulent Rayleigh-Bénard convection (RBC) in a wide-aspect-ratio, cylindrical domain [60]. The slowly evolving coherent motions make it very difficult to use conventional time averaging procedures to obtain statistically-converged results over the finite-time of numerical simulations. Spatial averaging over homogeneous directions helps somewhat, as will be shown below, but it does not completely cure the problem. The purpose of this letter is to propose a new technique that accounts for the influence of multiple states in large-scale organization of coherent structures and combines temporal and a specially-constructed ensemble averaging to significantly improve the statistical convergence in finite-time simulations of Rayleigh-Bénard convection.

RBC occurs when fluid between horizontal plates is heated from below and cooled from above. The unstable temperature stratification generates buoyancy forces within the fluid layer which then drive the flow. The Rayleigh number $Ra = \beta g \Delta T h^3 / \alpha \nu$, (where β is the coefficient of thermal expansion, g is the gravitational constant, ΔT is the temperature difference between the two heated plates, h is the plates' vertical separation, α is the thermal diffusivity and ν is the kinematic viscosity), is the primary dimensionless parameter and the Prandtl number $Pr = \nu/\alpha$ is often of less importance. Ra is heavily dependent upon the vertical length scale h. A horizontal length scale (L) is also very important for determining the structure of the flow. The ratio of these two length scales is the aspect-ratio ($\Gamma = L/h$).

The majority of numerical and experimental studies have been performed in unit Γ boxes and cylinders. The "wind of turbulence" concept is often used to describe the flow structure in these small Γ domains. The "wind of turbulence" is characterized by a single roll-cell, or large-scale circulation (LSC), which spans the height and

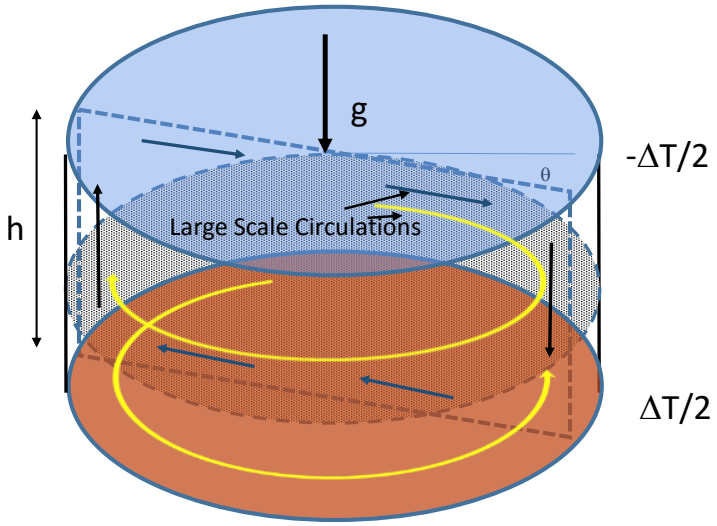


Figure 21: Conceptual diagram of the wind of turbulence in a $\Gamma = 1$ cell. The dotted plane illustrates the one of the infinite possibilities for the azimuthal orientation of the LSC. The yellow vectors indicate the directions for azimuthal drift.

width of the cell, see figure 21. This roll-cell creates boundary layers along the side walls and thermally active top and bottom plates which are well described by the Prandtl-Blasius profiles according to the Grossmann and Lohse theory [32]. The core of these small Γ cells is well-mixed and shows statistical properties in line with homogeneous turbulence [64].

In boxes the LSC may align with the side-walls, but in cylinders circular symmetry of the side-walls and verticality of the gravitational vector combine to imply that there can be no preferred horizontal direction, i.e. structures can align in any horizontal direction, and the infinite-time mean of any quantity must be independent of the azimuthal direction (azimuthal homogeneity). In particular, the LSC is allowed to flow in any direction. Experiments and numerical simulations show that the direction of the LSC (and presumably the azimuthal orientation of any flow pattern) drifts in

time [12, 51], so that over an infinite time all azimuthal orientations become equally probable, implying statistical homogeneity in the azimuthal direction and suggesting the azimuthal averaging as a means to accelerate statistical convergence to an infinite time-average in this flow. In horizontal RBC cells the anti-symmetry of the thermal boundary conditions on the horizontal surfaces also implies anti-symmetry of statistical means for quantities involving temperature or heat flux with respect to reflection about the horizontal mid-plane.

The anti-symmetry about the mid-plane is due to the vertical direction of the gravitational vector and the equal and opposite temperatures (with respect to the mean value) of the thermally active boundaries. There is nothing in the equations or boundary conditions to give preference to updrafts or down-drafts, so thermal plumes rise (fall) from the lower (upper) boundary with equal likelihood. Referring to figure 21, the updraft on the left hand side of the flow has an equal probability of being a down-draft over an infinite time. When the large-scale circulation is a single roll-cell 180° rotation about the central axis changes the updraft on the left to a down-draft. However, as Γ is increased the flow's structure acquires a more complicated form than the relatively two dimensional "wind of turbulence" and azimuthal homogeneity can diverge from anti-symmetry in the vertical direction.

4.2 Additional States in Larger Aspect-Ratio Cylinders

In our recent work we studied the large-scale structures in a 6.3 Γ RBC cell via direct numerical simulation (DNS) [60]. This simulation was setup to mirror an experiment conducted by Fernandes [22]. After smoothing out the small-scales with a running time average we observed that the flow organized itself into a hub and spoke

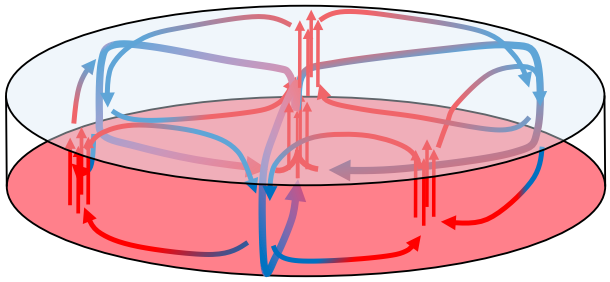


Figure 22: Possible patterns at $\Gamma = 6.3$. This pattern is characterized by large-scale updraft in the center, and six large-scale drafts of alternating direction along the cell's side walls. Three dimensional roll-cells are created by connecting each updraft with the neighboring downdrafts.

like pattern with an updraft in the central region of the cell, and 6 alternating up- and down-drafts near the outer wall. The hub in this pattern is the central thermal and the spokes are the vortex lines that form between drafts of opposing direction along the outer wall. A conceptual illustration of the observed pattern's thermal signature is provided in figure 22. Very similar patterns were seen in the numerical study by Bailon-Cuba *et al* [7]. The large-scale patterns in our recent work [60] and the work of Bailon-Cuba *et al* [7] showed no azimuthal drift or vertical reversal over at least 600 free fall time units ($t_f = \sqrt{h/\beta g \Delta T}$) in numerical simulations. From this we can infer that the large-scale patterns in turbulent RBC are remarkably stable at large Γ.

We will refer to the observed pattern in figure 22 as $state^+$ because the central column of fluid is an updraft. The persistence of the central column destroys the statistical homogeneity at the center of the cell over the set of realizations in $state^+$. This stands in direct conflict with the idea that as Γ is increased the central region of the cell should approach the infinite Γ case which is statistically homogeneous over horizontal planes. Clearly, additional states must exist in the infinite ensemble of realizations for this flow, and these states are not represented in this data set

even though it was sampled over more than $600t_f$. If the temporal sampling were sufficiently extended to truly approach the infinite-time average then an event must occur that will drive the flow into other states. Possible states should, at the very least, include rotations about the central axis, and a reorganization of the large-scales to where the central region of the cell is characterized by a downdraft. We will refer to downdraft organization as $state^-$.

As was mentioned earlier, states that are simply a shift in the pattern's orientation can easily be accounted for by averaging in the homogenous azimuthal direction, as would be sufficient in a low Γ case. However, the downdraft pattern in large Γ case will require another state of the flow to be sampled. Without this additional state the data set can be considered a conditional average of the infinite time field based on the updraft large-scale organization. The realizations of the flow in this data set can not be truly statistically independent because the large-scale structures remain highly correlated throughout the time scales achievable in the simulations.

Traditionally, numerical simulations have relied on temporal averaging for obtaining flow statistics with the expectation that the statistically independent states will be naturally sampled over the duration of the simulation. This assumption has two drawbacks: first, as we see in our RBC example, time-scales on which coherent structures evolve can be very significant, so that the amount of run time needed to follow this evolution through many transitions between up- and down-states can be prohibitively large; second, the mechanisms triggering the transitions between the states are still unknown and might not be easily reproducible in continuously executed numerical simulations. For example, carefully conducted experiments in Rayleigh-Bénard convection involved periodically switching the heat source off and on

to produce significant perturbations to trigger statistically independent realizations [22, 23].

The idea that ensemble averaging, instead or an addition to temporal averaging, is a promising way to improve statistics and models in the simulations has been recognized [13, 17]. In these works, initial conditions were selected randomly with the presumption that this initial randomness would yield significantly different realizations. Although intuitively appropriate, this approach might still fail, since the dependence of large-scale structural organization on initial conditions is little understood. It might happen that all initial conditions chosen at random will produce the same state (for example, $state^+$ as in our simulations). A very large number of random realizations might still be biased to one state or another.

4.3 State Switching Techniques

In this chapter, we propose a simple modification of the conventional sampling and averaging procedures that allows us to select initial conditions for the effective ensemble averaging in a controlled way. With this technique, the additional states that will be sampled are created to possess certain properties (for example, a central downdraft versus updraft) that are missing in the "base" realization. By deliberately constructing and sampling specifically manufactured conditions that sample all states, we ensure that the statistics converge to an unbiased estimate of the infinite-time average using a relatively small number of realizations. For example, in this chapter we achieve significantly improved statistics with only two realizations, sampling over $state^+$ and $state^-$ as discussed below. This technique can be used to expand the statistical significance of numerical data sets and extend the number of *independent*

realizations that can be studied. We will illustrate this technique using our RBC data set, but it can potentially be applied to other flows where there are symmetries that allow solutions in multiple states.

The main idea behind our technique is to transform an instantaneous realization from the numerical data set into an initial condition for a different state that possesses a desired large-scale structure. For this, we explore symmetries in the inhomogeneous directions. In addition, we require that the transformed data set evolves according to the governing equations. Our central goal for the manipulation performed in this chapter is to reverse the flow direction in a central column (updraft versus down-draft) corresponding to $state^+$, identified in figure 2, and its reflection, $state^-$. We recognize that other symmetries (for example, based on the direction of the azimuthal rotation) can also produce other turbulent states.

To reverse the flow direction in the central column, we recast the field so that the structures falling from the cool top plate appear as structures rising from the warm bottom plate and vice versa. Switching states is performed by transforming the vertical velocity component, vertical coordinate and temperature of a developed turbulent data set at every grid point in the simulation. The formulas for performing this switch are as follows:

$$z^-(x, y, z^+) = z_t + z_b - z^+(x, y, z^+) \tag{4.1}$$

$$\theta^-(x, y, z^-) = \theta_t + \theta_b - \theta^+(x, y, z^+) \tag{4.2}$$

$$w^-(x, y, z^-) = -w^+(x, y, z^+) \tag{4.3}$$

where the subscripts t and b refer to the values at the top and bottom boundaries, the superscripts $+$ and $-$ refer to the flow states, z, θ and w are the vertical coordinate, dimensionless temperature and vertical velocity, respectively. The transformation pro-

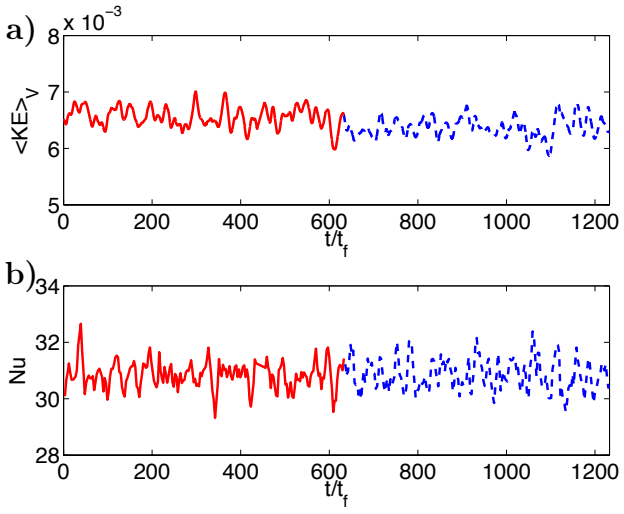


Figure 23: Temporal evolution of the volume average kinetic energy (a), and Nusselt number (b) are shown in the plots above. $state^-$ (- -) was initialized from the last time step of $state^+$ (–).

vided by equations (4.1)-(4.3) reflects all variables in the flow about the midplane and preserves the Navier-Stokes equations with the Boussinesq approximation, continuity equation and thermal energy equation exactly.

The plots in figure 23 show identical signatures in the volume averaged kinetic energy and Nusselt number as the transition from $state^+$ to $state^-$ takes place. These results verify that this methodology preserves the continuity in volume average quantities, such as kinetic energy and total heat flux, during the state transition. Utilizing this technique to transition between converged states with long term statistical significance has the potential to improve statistical convergence in DNS studies of RBC at a significant reduction in computational expense.

To illustrate this point we have included a comparison with the statistical profiles from experiments of Fernandes [22] and our previous work [60] in figure 24. These

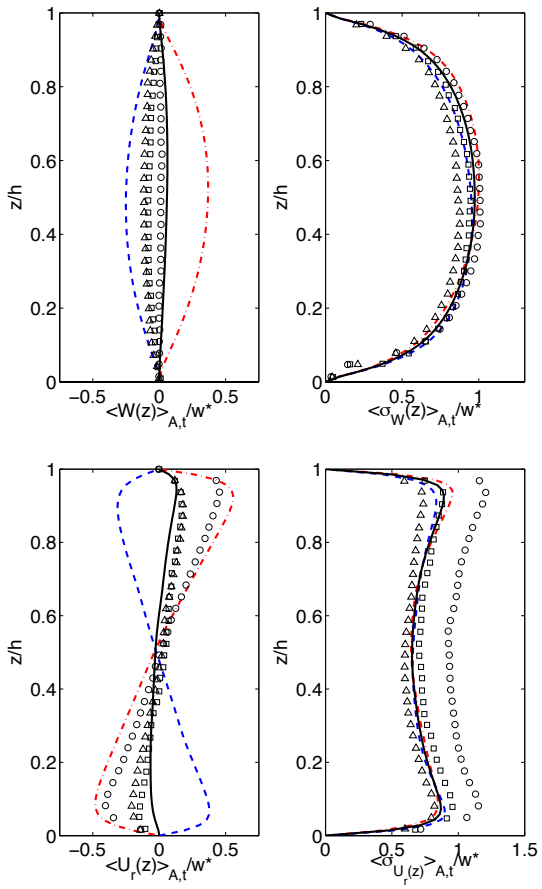


Figure 24: Ensemble and horizontally averaged mean (left) and r.m.s. (right) vertical (top) and radial (bottom) velocity profiles normalized by Deardorff's velocity scale $w*$: DNS: $Ra = 9.6 \times 10^7$, averaging domain Ω_F where red ($-\cdot$) is $state^+$, blue (- -) is $state^-$ and black (–) is the average of $state^+$ and $state^-$; Experiment: $Ra = 6 \times 10^7$ ($\triangle$), $Ra = 2 \times 10^8$ ($\square$), $Ra = 1 \times 10^9$ ($\circ$)

profiles have been normalized by Deardorff's velocity scale $w* = (\beta g Q_o h)^{1/3}$ [18] where Q_o is the kinematic heat flux.

The profiles generated in figure 24 are taken from within the core region of the 6.3 Γ RBC cell with a radius of $0.925h$. It should be noted that the experimental mean vertical velocity profiles decrease in magnitude and the r.m.s vertical velocity profile's magnitude increases with Ra. Since the forcing within the cell increases

with Ra the characteristic instantaneous velocity increases as the vertical r.m.s profile indicates. We hypothesize that the reason for the decay of the mean velocity in the experimental profiles is that the antisymmetric states were more evenly accounted for in Fernandes ensemble averages with higher Ra [22]. When the original DNS results ($state^+$ only) are compared against the experiment we see that the r.m.s profiles fall within $\sim 11\%$ in a pointwise comparison and that the mean profiles are dramatically over predicted. However, when the $state^+$ and $state^-$ are averaged together, the mean velocity profiles are very close to the expected (zero) value of the infinite-time average and the r.m.s profiles show an excellent match with the experimental results. This is truly remarkable when one considers that each instance of Fernades' ensemble average (with the total of 300 instances) was also temporally averaged over a greater time period than our entire simulation. By our estimates it would take us $O(10^8)$ CPU hours to recreate Fernandes experiment on our current grid (and the ability to recreate the desired uncorrelated large-scale patterns with just random initializations still could not be guaranteed). However, the results presented in this chapter took $O(10^5)$ CPU hours to produce. Perhaps the most exciting observation is that the profiles in figure 24 clearly show that we were able to obtain a net downdraft in the central region of the cell over the sampling time of our second state. This shows that we were able to perform a targeted manipulation of instantaneous data to trigger a new state of the large-scale structures.

4.4 Summary and Conclusions

In summary we have discussed the challenges in obtaining the flow statistics that would converge to an infinite-time average in numerical simulations of turbulent flows

and the bias that can be introduced due to insufficient sampling of flow states. Failure to recognize insufficient sampling can lead to variance in the statistics of stationary processes which are due to the correlation of slowly evolving large-scale structures that can persist well beyond the standard integral time scales. We used our recent numerical simulation of a 6.3 Γ RBC cell to provide an example of how this can occur. The results from this simulation were obtained with a high-order method and were numerically well resolved, as well as yielded longer than average temporal sampling [60], but the central region of the cylinder showed inhomogeneous nature due to a large-scale updraft persisting in the cell's core. The only way to resolve this issue is to average these results with another state of the flow field with a downdraft in the center. We then presented a methodology for triggering this state using the inherent symmetries in the inhomogeneous vertical direction that didn't alter the net kinetic or thermal energy in the fully developed turbulent field. The application of this methodology showed that a net downdraft was indeed created in the region of interest and that this downdraft remained dominant over at least the same temporal averaging period that was used to collect the first flow state. This method has the potential for application to other flows that have multiple, long-lived states and an exploitable symmetry.

Chapter 5

AZIMUTHAL FOURIER DECOMPOSITION

5.1 Introduction

The primary objective of this chapter is to quantify and characterize the properties of the large-scale structures that have been described in the previous chapters. It has already been shown that the large-scale structures evolve over very long times [60, 7, 21], and so the computations in this chapter are evaluated over a length of time that is a factor of five longer than the analysis presented in chapter 3 for a total simulation time of $3054t_f$.. This time extension was selected based off the estimate of Emran and Shumacher [21] that the large-scale structures for large Γ should drift on a timescale of $O(10^3)t_f$.

This chapter will utilize Fourier analysis to answer the following research questions.

- How well do the Fourier modes align with the physical structures within the flow field?
- What are the length scales that describe the multi-roll cell large-scale structure in this flow?
- How is the flows structure effected by inhomogeneity in the r and z directions?
- How persistent are these structures? What are their time-scales?
- Are there similarities among the other scales that do not directly describe the large-scale structure, if so what are they?

The chapter is written so that it can be read as an independent work, and is the foundation for a journal article that will be submitted shortly after the defense process

is completed. As such, it provides an outline of the numerics, governing equations and analysis methodology. This is followed by the data analysis which includes defining and commenting on the mean flow, spatially and temporally averaged energy spectra, temporal evolution of the Fourier modes, spatially varying integral time scales for the total flow field and the individual Fourier modes, as well as the effects of spatial inhomogeneity on the energy spectra. The chapter concludes with a discussion and summary of the results as they pertain to the chapter's research questions.

5.2 Numerics, Nomenclature and Definitions

This work relies heavily on Fourier decomposition to analyze the structure of turbulent RBC in a domain where multiple roll-cells are present. This section's primary purpose is to provide an overview of the domain, normalizing scales, governing equations and notation used throughout this work. A small primer on Fourier decomposition is also provided in this section.

5.2.1 Domain and Scaling

The computation domain Ω in this study is a cylinder with height H and diameter D. Ω can be expressed in cylindrical coordinates that are normalized by H and symmeterized about the mid-plane ($H/2 \rightarrow z = 0$) such that normalized Ω is defined as

$$\Omega(r, \theta, z) \rightarrow r \in [0, \Gamma/2], \theta \in [0, 2\pi), z \in [-1/2, 1/2] \tag{5.1}$$

where Γ is the aspect ratio (D/H) of the cylinder. Ω is also aligned with the gravitational vector ($\mathbf{g}$) such that $\frac{\mathbf{g}}{|\mathbf{g}|} = -\hat{e}_z$ where $\hat{e}_z$ is the unit normal z-direction.

Velocity and temporal units are normalized by the "free-fall" velocity ($w_f = \sqrt{\beta g \Delta T H}$) and time ($t_f = H/w_f$) where β is the coefficient of thermal expansion, g is the gravitational constant, and ΔT is the temperature difference between the top and bottom plates of the convection cell. ΔT is also used to normalize the temperature field.

Utilizing the outlined scales the non-dimensional form of the Boussinesq equations for RBC can be expressed as:

$$\nabla \cdot \mathbf{u} = 0 \tag{5.2}$$

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}}\nabla^2 \mathbf{u} + \vartheta \hat{e}_z \tag{5.3}$$

$$\vartheta + (\mathbf{u} \cdot \nabla)\xi = \frac{1}{\sqrt{RaPr}}\nabla^2 \vartheta \tag{5.4}$$

where $\mathbf{u}$, p and ϑ are the dimensionless velocity, pressure and temperature. The Rayleigh (Ra) and Prandtl (Pr) numbers in equations 5.3 and 5.4 are defined as:

$$Ra = \frac{\beta g \Delta T h^3}{\alpha \nu} \tag{5.5}$$

$$Pr = \frac{\nu}{\alpha} \tag{5.6}$$

where g, α and ν are the gravitational constant, thermal diffusivity and kinematic viscosity respectively.

In this study $\Gamma = 6.3$, $Ra = 9.6 \times 10^7$, $Pr = 6.7$ and the boundary conditions are no-slip on all surfaces, constant temperature on the top and bottom plates, and adiabatic side-walls. Additionally, the individual components of the velocity field are expressed in their cylindrical components such that $u_i = \{u_r, u_\theta, u_z\}$.

5.2.2 General Numerics

The data in this study is produced from a direct numerical simulation (DNS) using the open source spectral element code Nek5000. Nek5000 is an extensivly validated research code that has been used to publish hundreds of scientific papers, and details regarding the code can be found at [24]. Additional details regarding resolution, convergence, and comparison with experiments for the specific computations in this work can be found in the prior work ([60]).

The dataset in this work used the results from [60] as initial conditions and contains $3054t_f$ of temporal data sampled every $3t_f$. Each snapshot is projected onto cylindrical coordinates using spectral interpolation routines native to Nek5000, and the velocity components are transformed from cartesian to cylindrical. This was previously done on a smaller scale ([60]), but in this work it has been extended to the entire domain. Cylindrical coordinates is the logical choice for analyzing the dataset and facilitates operations along the domain's periodic, azimuthal direction.

The DNS snapshots are sampled with [160,64,2048] points in r, θ and z respectively to generate the cylindrical grids used for analysis. Non-uniform, Gauss-Legendre (GL) quadrature is used to sample in the r and z directions, but the θ direction uses equispaced sampling points to facilitate Fourier transforms. GL quadrature does not include the end points and is defined on the standard interval $x \in (-1, 1)$. GL quadrature is selected to facilitate high accuracy numerical integration and removes sampling to the along the walls and of cell where the information holds little value. The boundaries in the z direction are defined with Dirichlet boundary conditions, and so sampling on them for post-processing purposes is trivial. In the r direction points along the central axis ($r = 0$) are at a spatial singularity in the cylindrical coordinates

representation and will provide no additional data when Fourier transforms in θ and integration over the r-z plane are applied. The points along $r = \Gamma/2$ have Neumann boundary conditions in the temperature field but virtually no information is lost since gradient at the wall is zero (adiabatic) and the GL quadrature samples very close to the boundaries.

5.2.2.1 Fourier Decomposition

In this work Fourier decomposition in the azimuthal direction is heavily relied on to gain insight into the structure of the flow field. Fourier modes are an ideal choice because the azimuthal direction is analytically periodic, and Fourier modes are analytically defined since the flow is incompressible and there for smooth and continuous. These modes are orthogonal and are an optimal basis for decomposing a continuous, smooth periodic signal. Several studies documented in [9] have also shown that applying Fourier analysis to periodic or statistically homogeneous directions will significantly improve the convergence of POD.

Fourier decomposition provides additional benefits in this study that extend beyond the mathematical significance of the modes. For example, azimuthal motions for RBC in cylinders tend to evolve on extremely long time scales, and the azimuthal velocity signals are relatively weak ([12], [51]). Performing an analytical decomposition such as Fourier analysis allows the azimuthal evolution of the flow to be studied in a well understood format.

Throughout this work Fourier coefficients are indicated by the $\hat{u}$ accent, the Fourier operator is indicated by $\mathcal{F}[u]$ and the Fourier mode numbers are referred to by integer their frequency over the interval $[0 : 2\pi)$ k. All averages will be noted by the brackets

$<>$ and subscripts will be listed by the order in which the averaging operations were applied. For instance, $< u_z(z,t) >_{\theta,r}$ is the time varying vertical profile of the vertical velocity field after averaging in the azimuthal and radial directions. Additional subscripts that indicate averaging operators are: V for volume averaged, A for area averaged, and E for ensemble averaged.

5.3 The Mean Field

The primary interest of this study is to investigate the properties of structures ranging from the largest scales to the integral scales of the flow field. These structures all have a finite life span and therefore reside in the fluctuating field with respect to Reynolds decomposition. However, a fluctuation is a relative quantity that must be defined with respect to some mean value. Therefore it is essential to define the mean field and the averaging operators that create the mean field about which the fluctuations occur.

In this work the mean field will be defined by $<>_{\theta,t}$. The azimuthal average is extracted via Fourier decomposition by the zeroth order mode, and so the mean flow in this work is defined as:

$$< u_i(r,z) >_{\theta,t} = \begin{cases} < \hat{u}_i(r,k,z) >_t & when \quad k = 0 \\ 0 & when \quad |k| > 0 \end{cases} \tag{5.7}$$

where the index k indicates the mode number. Conversely this also means that the fluctuating field contains all wave numbers except for zero, and $\hat{u}_i(r,k,z,t)|_{k=0} - < \hat{u}_i(r,k,z) >_t |_{k=0}$.

Since the flow is statistically stationary in time (see [60]) and periodic in θ these operators represent the best estimate of the true Reynolds average or infinite time-

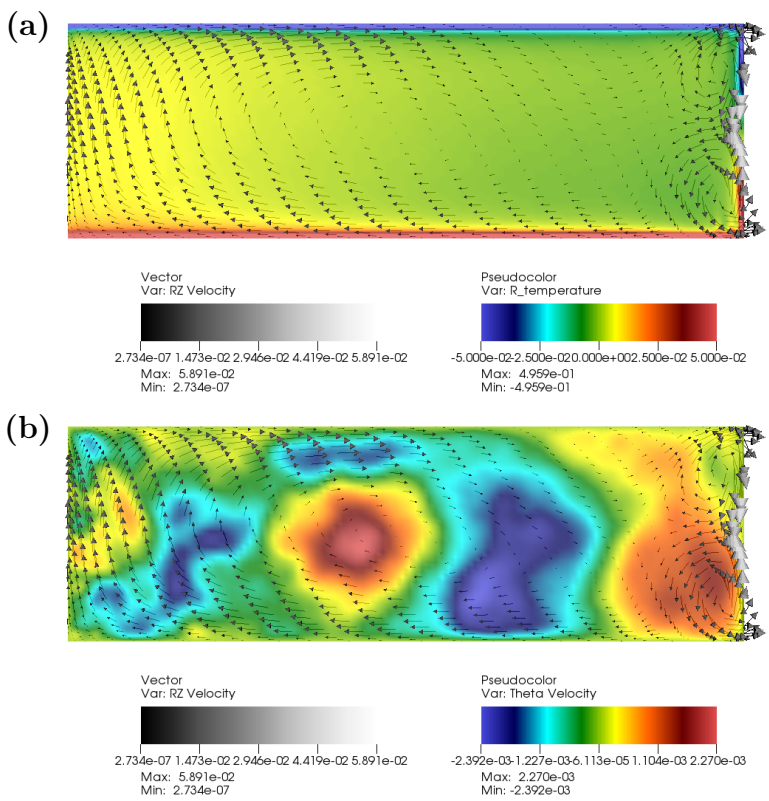


Figure 25: Azimuthal and temporally averaged mean fields. The color scheme in (a) corresponds to $< \vartheta >_{\theta,t}>$and in (b) it corresponds to $< u_\theta >_{\theta,t}>$
while the vector field in both plots is of the two dimensional vector of
$\{< u_r >_{\theta,t}, < u_z >_{\theta,t}\}$

average for this flow field. With an infinite number of realizations or an infinite amount of time it is expected that the mean will converge to an axisymmetric representation due to the symmetry of the domain. While the total temporal sampling period in this study is not sufficient to confidently approach the true Reynolds average field, the use of these averaging operators represents a good approximation. The mean field is displayed in figure 25.

The mean field in figure 25 displays several interesting characteristics. Starting at the sidewalls ($r = \Gamma/2$), two counter rotating roll cells can be observed with stagnation

point at $z = 0$ where the two roll cells meet. Additionally, a thermal boundary layer can be seen along the adiabatic sidewalls. These roll cells and the accompanying boundary layers are inline with the vertical antisymmetry of the convection cell, and thus are most likely present in the true Reynolds averaged flow field. Conversely, the dominant up-draft in the center of the cell does not conform to the inherent symmetries of the RBC cell. The boundary conditions make it equally likely that a down-draft could be present over this region, and so the structures that are described in this study should be interpreted as a subset of the true Reynolds decomposition where an updraft is present in the central region of the cell. The mean azimuthal velocity component shows that a preferential direction for rotation or drift is not consistently present across the entire time series for this data set.

5.4 Spatial Description of the Large-Scale Structure

In this section the largest scales of the flow field are investigated. These scales are of interest because they tend to contain the majority of the energy in the flow field, persist for a long periods of time, and are responsible for a large portion of the inhomogeneity. Figure 26 shows the volume and time averaged energy spectra for the various flow variables. Volume and time averaging were applied to energy coefficients to smooth out transients so the most dominant structures can be seen.

The spectra in figure 26 indicate that the $k = 2$ Fourier mode is the single most dominant mode over the range of the simulation. The peak is very pronounced in the temperature and azimuthal velocity fields, but more subtle in the radial and vertical velocity components.

While the spectra in figure 26 indicates the dominant structure over the life-span

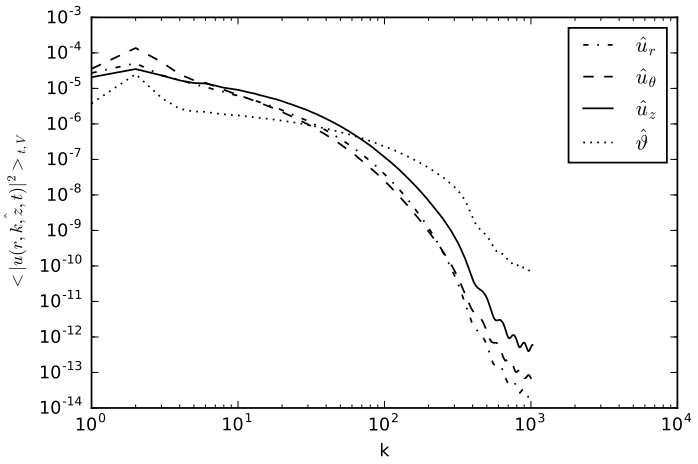


Figure 26: Volume and time averaged energy spectra ($t \in [0 : 3054t_f]$).

of the simulation, it is also possible that the structure evolves throughout the course of the simulation. In the authors' previous work ([60]) and the work of [7] no significant evolution of the large-scale structures were observed. However, in the more recent work of [21] it was predicted that the large-scale structures will evolve on time scales of $O(10^3 t_f)$, and both [7] and [60] only conducted simulations for time scales of $O(10^2 t_f)$. In the present work the simulation time has been extended to the order where temporal evolution of the large-scale structures should occur.

Figure 27 contains the energy spectra for the temporally filtered temperature field. Temporal filtering removes the majority of the small-scale structures leaving the highly correlated large-scale structures and it is a good technique for observing the slowly evolving large-scale dynamics. The temperature field spectrum is selected for comparison because it contains the most distinguished peak in figure 26. The period for the temporal filter was selected to be $600t_f$ which is inline with the time scale of the prior works where no major evolution was observed ([60, 7, 21]). A visualization

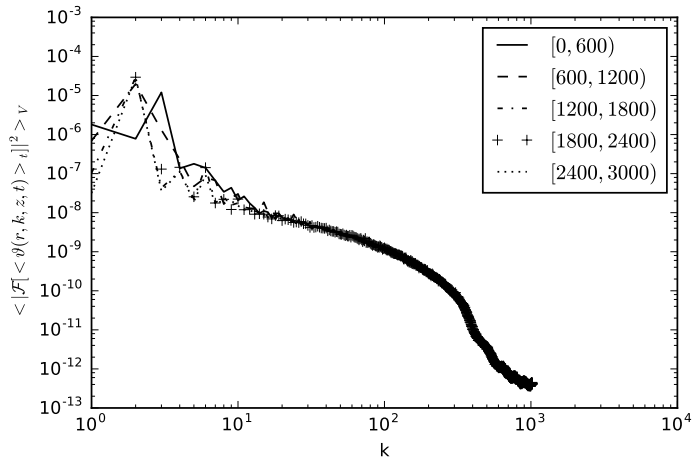


Figure 27: Volume averaged energy spectra for the temporally filtered temperature field. Filtering is performed by applying a running time average with a period of $600t_f$. The legend entries refer to the averaging period of each instance.

of the temporally filtered temperature fields is provided in figure 28, and the instances in figure 28 correspond to the energy spectra in figure 27.

Figure 27 shows that over the first $600t_f$ $k = 3$ is the dominant mode, but that the dominant mode transitions to the $k = 2$ over the next $600t_f$. The first instance of the filtered field also shows a larger distribution of energy in the other low order modes, but by $k = 12$ the energy content is about the same for all instances of the filtered field. The second instance of the filtered field shows higher energy content in modes $k = 1$ and 3, but by the third instances the energy has concentrated itself in $k = 2$. One possible interpretation of this transition is that the $k = 3$ dominant structure is less stable than the structure corresponding to $k = 2$ because the turbulent thermal energy is distributed among a larger number of low order modes.

Looking at the individual modes can help explain their contribution to the over all flow field. The first few low order modes and their cumulative summations

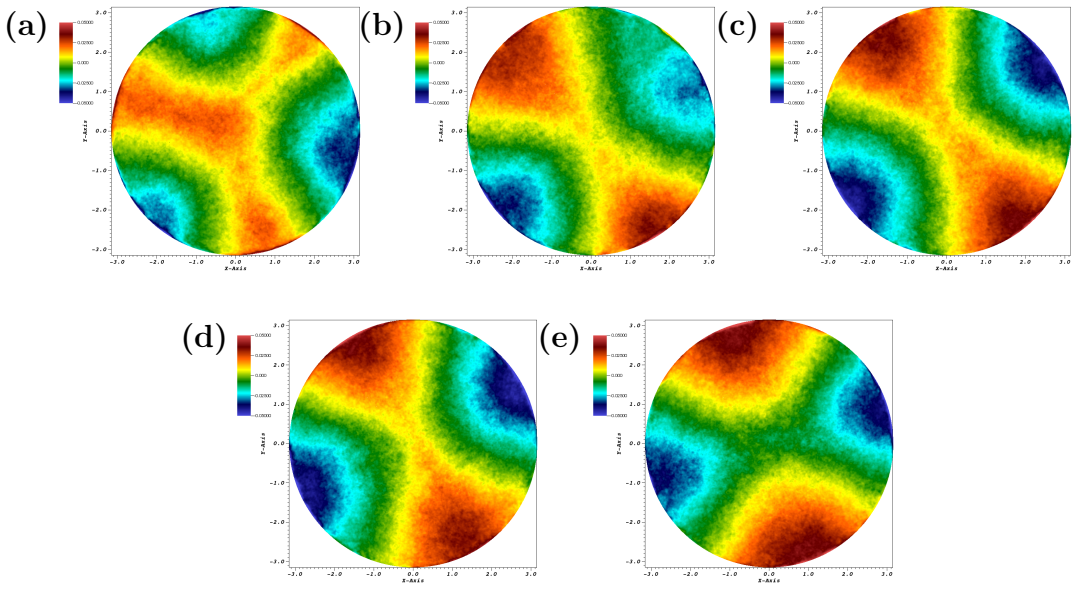


Figure 28: Temperature at the mid-plane of the cell after temporally filtering over a period of $600t_f$ with a running time average. The time ranges covered by each subplot are: a) [0,600), b) [600,1200), c) [1200,1800), d) [1800,2400), e) [2400,3000). Temperature is scaled from $[-0.05 : 0.05]$ in all subplots.

corresponding to temperature fields in figure 28(a) and figure 28(d) are provided in figure 29 and 30.

The modes in figure 29 can be interpreted with the following rolls: $k = 0$ establishes a central, warm column, $k = 1$ and 2 shift the central column and bias the structure along the edge of the convection cell and $k = 3$ finalizes the hub-and-spoke like structure that was outlined in [60]. A qualitative comparison of figure 29(h) and figure 28(a) show that the total structure is well described by the first 4 Fourier modes.

However, examination of the modes displayed in figure 30 show that the structure for this case is almost fully described by $k = 2$. This convergence of energy and structure toward a single mode seems to indicate a stabiliazation for the system as a

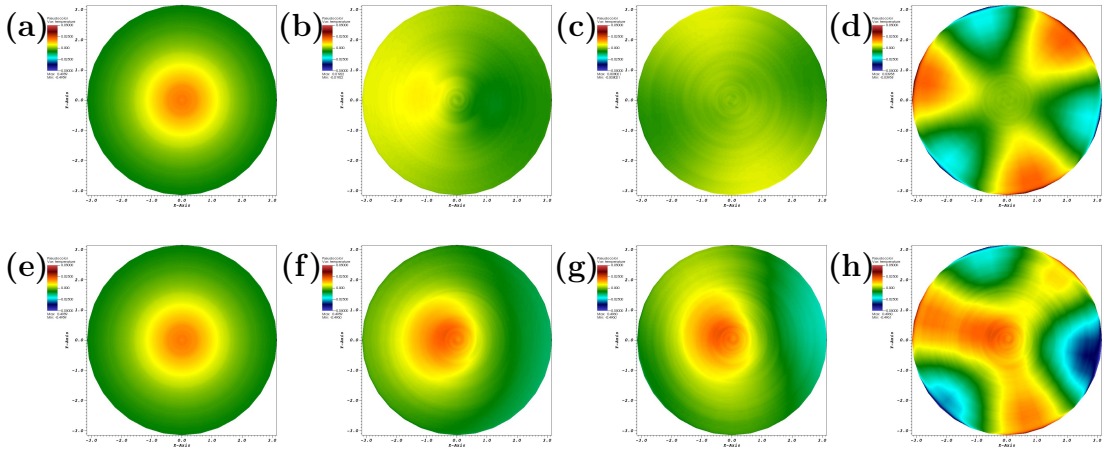


Figure 29: Individual Fourier modes for the temporally filtered temperature field that has been averaged over the interval $t \in [0, 600)$: (a)-(d) corresponding to $k = 0$ to 3 respectively. Summation of Fourier modes $k = 0$ (e), $k = 0 : 1$ (f), $k = 0 : 2$ (g) and $k = 0 : 3$ (h). Temperature is scaled from $[-0.05 : 0.05]$ in all subplots and all plots are at the mid-plane.

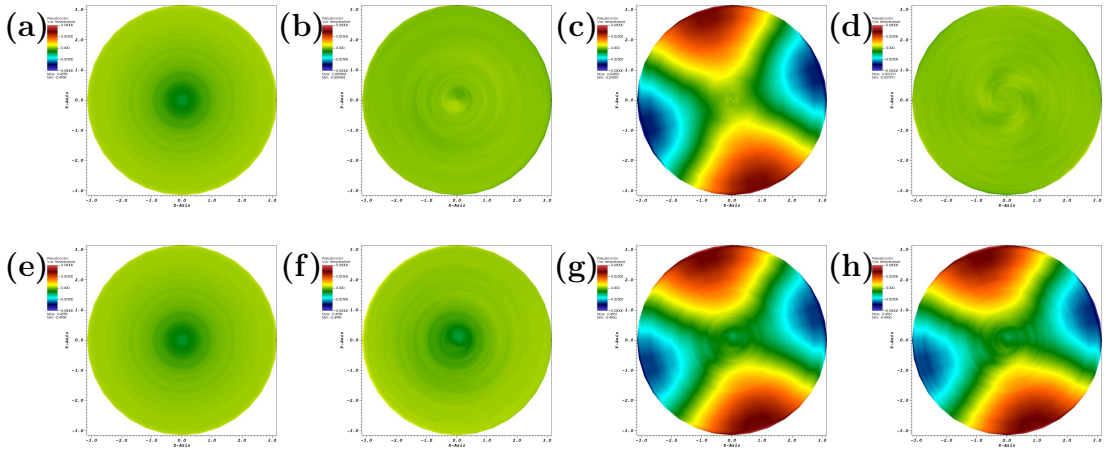


Figure 30: Individual Fourier modes for the temporally filtered temperature field that has been averaged over the interval $t \in [2400, 3000)$: (a)-(d) corresponding to $k = 0$ to 3 respectively. Summation of Fourier modes $k = 0$ (e), $k = 0 : 1$ (f), $k = 0 : 2$ (g) and $k = 0 : 3$ (h). Temperature is scaled from $[-0.05 : 0.05]$ in all subplots and all plots are at the mid-plane.

whole. This presents a reasonable argument for $k = 2$ being the long term structure of the flow field at this Γ, Ra and Pr, but because the $k = 3$ structure remained coherent for approximately 1/5th the total simulation time, nothing definitive can be determined. It is still possible that the system could undergo another transition and modulate back to a $k = 3$ dominated structure. However, it is worth noting the length of time in which this transient evolved for future studies of RBC in Γ where multi-roll cell structures persist.

5.5 Temporal Description of the Large-Scale Structure

5.5.1 Temporal Evolution of the Flow Field

The previous section relied on the smoothing properties of time averaging to investigate the spatial structure of the large-scale structures in the flow field. In this section the temporal evolution of a few select Fourier modes will be investigated in detail to shed further light on the temporal evolution of the large-scale structures. The investigation is performed by plotting the area integrated Fourier coefficients for a given mode on the complex plain (see eq. 5.8).

$$\hat{u}(k,t) = \int_z \int_r \mathcal{F}[u(r,\theta,z,t)] r \partial r \partial z \tag{5.8}$$

Area integration removes the localized spatial variations of the mode and allows the temporal evolution to be investigated from a macro perspective. Even though the area integrated Fourier coefficients only depend on time, they are still complex variables. The phase and amplitude of the volume integrated coefficient can simultaneously change. Plotting on the complex plane allows an intuitive way to view the changes in amplitude and phase for a given wave number.

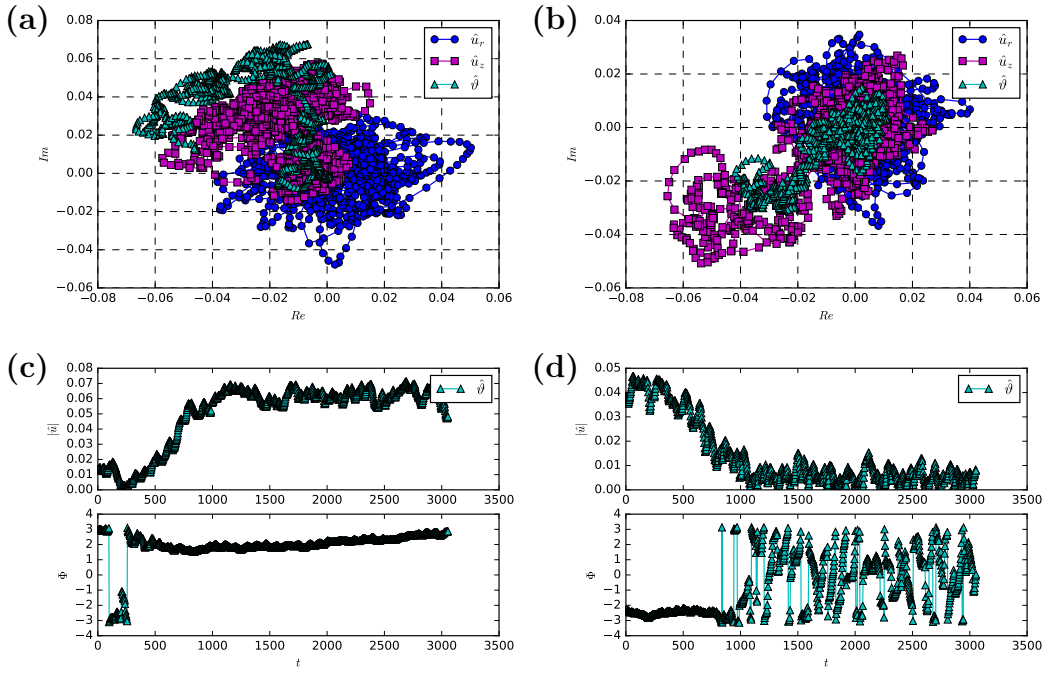


Figure 31: Temporal evolution of the area integrated Fourier coefficients plotted on the complex plane for $k = 2$ (a) and $k = 3$ (b). The temperature field's area integrated Fourier coefficients are also plotted in terms of phase (Φ) and amplitude ($|\cdot|$) for $k = 2$ and $k = 3$ in subplots (c) and (d) respectively.

Figure 31(a) and (b) shows the evolution of modes $k = 2$ and $k = 3$ for $\hat{u}_r$, $\hat{u}_z$ and $\hat{\vartheta}$. $\hat{u}_\theta$ was not included in these plots because it has a behavior that is very similar to $\hat{u}_r$. It is probably difficult for the reader to tell exactly how these plots are behaving in time since the data is still somewhat chaotic. To assist in comprehension a supplemental narrative is provided for both plots from the perspective of $\hat{\vartheta}$.

$\hat{\vartheta}$ in figure 31(a) begins near the origin and as time progresses it tracks up along the complex plane and then begins to drift into quadrant 2 of the real-complex plane. It moves rather chaotically but maintains a somewhat constant radius as it drifts in the counter clock-wise direction. $\hat{\vartheta}$ in figure 31b) begins in quadrant 3 of the real-complex plane. As time progresses it moves its way to the origin and then oscillates randomly

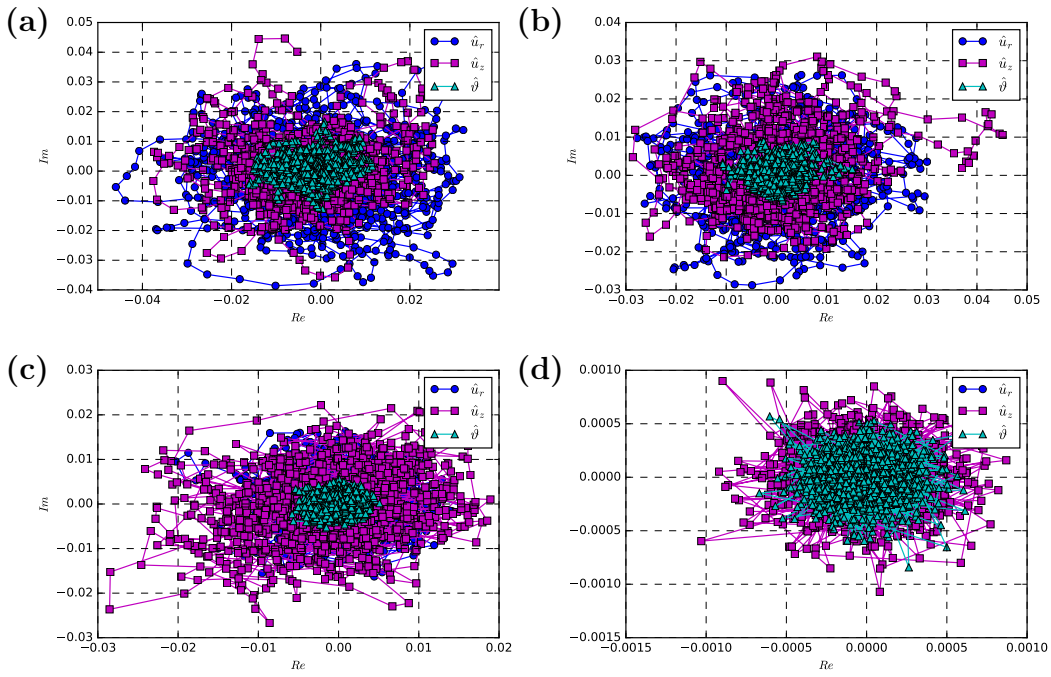


Figure 32: Temporal evolution of the area integrated Fourier coefficients for $k = 4$ (a), $k = 5$ (b), $k = 10$ (c) and $k = 100$ (d) plotted on the complex plane.

about the origin. When the mode is the dominant mode in the large-scale structure $\hat{\vartheta}$ drifts away from the origin, and when it loses its dominance it becomes centered around the origin with an amplitude scattered between zero and a maximum radius. This is further illustrated in figure 31(c) and (d) where $\hat{\vartheta}$ is plotted in terms of its amplitude ($|\hat{u}|$) and phase (Φ) for the $k = 2$ and $k = 3$ modes respectively.

Further inspection of the subplots in figure 31 shows some similarities between the two modes represented by each subplot. In both cases u_r is centered about the origin and is relatively evenly dispersed out to a given radius, and $\hat{u}_z$ follows the evolution of $\hat{\vartheta}$ albeit somewhat more dispersed. The complex plain evolution of several additional modes is plotted in figure 32.

All the modes in figure 32 show that the area integrated Fourier coefficients for all

the other modes display the same sort of scatter as $\hat{u}_r$ in figure 31. Similar scattering is also seen for $\hat{u}_z$ and $\hat{\vartheta}$ in figure 31(b) when the $k = 3$ is not the dominant mode. The scattering seen in non-dominant modes indicates that these modes behave as random processes from a global perspective. The dominant modes ($k = 2$ and $k = 3$) show additional structure that is not seen in the other cases implying a more deterministic nature exists for these modes.

Perhaps one of the most interesting things in the dominant modes is that a net rotation at a relatively fixed radius in the $k = 2$ mode is observed in figure 31a). Careful inspection of figure 28 also shows that a very slow rotation is starting to occur in the large-scale structure. However, it is hard to discern by just looking at the structure because the individual lobes of the large-scale structure modulate and shift in size. Figure 31a) gives a much clearer indication rotation is indeed occurring in the large-scale structure of the flow. However, the direction and magnitude of rotation in figure 31a) is different from that seen in figure 28.

5.5.2 Integral Time Scale

Now that temporal evolution of the large-scale structure has been verified, the next logical question to ask is: what are the time scales, or coherence times? An important quantity that can be used to measure the coherence times within the flow field is the integral time scale ($\mathcal{T}$). $\mathcal{T}$ is a metric for determining the temporal correlation of the flow field and the common metric for determining the appropriate spacing for two statistically independent instances for a stationary turbulent flow such as RBC is $2\mathcal{T}$. $\mathcal{T}$ is defined in terms of the auto-correlation (R_{ii}) which is defined as

$$R_{ii}(\Omega, \tau) =< V_i(\Omega, t + \tau)V_i(\Omega, t) > \tag{5.9}$$

where V is a vector containing variables of interest and τ is the temporal offset between the two instances of the flow field, or snapshots. V is often the turbulent velocity field ($V = \{u'_r, u'_\theta, u'_z\}$) where the prime indicates the fluctuating field. An autocorrelation based of this particular vector will determine a turbulent kinetic energy based correlation. In the case of RBC, another quantity of interest is the total turbulent energy, and in this case V is defined as $V = \{u'_r, u'_\theta, u'_z, \vartheta'\}$. Additionally, the averaging operator in equation 5.9 should be the same averaging operator that is used to define the mean field.

The classical definition of $\mathcal{T}$, and the one used in this work, can be expressed in terms of R_{ii} as

$$\mathcal{T}(\Omega) = \int_0^\infty \frac{R_{ii}(\Omega, \tau)}{R_{ii}(\Omega, 0)} d\tau \tag{5.10}$$

In this work the interest isn't just in the global time-scales, but also the time scales of the Fourier modes since the structures can be expressed in terms of Fourier decomposition. As such a definition of R_{ii} can be provided in terms of the Fourier coefficients as follows

$$R_{ii}(r, \theta, z, \tau) =< \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{V}_i(r, k, z, t+\tau) \hat{V}_i(r, k', z, t) e^{j(k+k')\theta} \partial k \partial k' > \tag{5.11}$$

where $j = \sqrt{-1}$. Equation 5.11 contains a convolution integral over the Fourier spectra of the two different snapshots. However, the expensive convolution computation can be avoided since the averaging operator includes the azimuthal averaging operator. Only the terms where the wave numbers sum to zero are included in the convolution integral since the exponent of the Fourier basis must equal zero to contribute to the azimuthal mean. Using these properties equation 5.11 can be expressed as

$$R_{ii}(r, z, \tau) =< \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{V}_i(r, k, z, t+\tau) \hat{V}_i(r, k', z, t) \delta_{k,-k'} \partial k \partial k' >_t \tag{5.12}$$

Since all the flow variables are real signals the negative Fourier mode $-k$ can be expressed as the complex conjugate of the the positive Fourier mode k. Therefore the dirac-delta in equation 5.12 shows that all wave numbers will contribute to the correlation, but only when they are multiplied by their complex conjugates. This also ensures that the correlation will be comprised entirely of real numbers which is required since this flow fields are defined in real space. The discrete representation of equation 5.12 is

$$R_{ii}(r,z,\tau) =< \sum_{k=0}^{N_\theta/2-1} \Re[a_k \hat{V}_i(r,k,z,t+\tau)\hat{V}_i^*(r,k,z,t)] >_t \tag{5.13}$$

where N_θ is the sampling rate for the Fourier transform in the θ direction, * indicates the complex conjugate, and $a_k = 1$ if $k = 0$ and $a_k = 2$ other wise. $\Re[]$ is an operator indicating that the real portion of the coefficient product. This operator along with the coefficient a_k are employed to use symmetry in the Fourier spectrum to reduce the calculation from two sums ranging from $[-N_\theta/2-1 : N_\theta/2-1]$ to one sum over the range $[0 : N_\theta/2-1]$. Technically the wave numbers are defined from $k \in [-N_\theta/2 : N_\theta/2-1]$ for the discrete Fourier transform, but the odd-ball wave number has been left out of the computations in this work because of it's incredibly small value (see figure 26). $R_{ii}(r,z,\tau)$ can also be interpreted as a weighted sum of the individual wave number correlations by defining a wave number correlation $R_{ii}(r,k,z,\tau)$. $R_{ii}(r,k,z,\tau)$ is defined in equation 5.14 by interchanging the linear summation and temporal averaging operators in equation 5.13

$$R_{ii}(r,k,z,\tau) =< \hat{V}_i(r,k,z,t+\tau)\hat{V}_i^*(r,k,z,t) >_t \tag{5.14}$$

Figure 33(a),(b) and (c) shows $\mathcal{T}$ for the entire field when v is defined as the turbulent kinetic energy, the turbulent thermal energy and the total turbulent energy

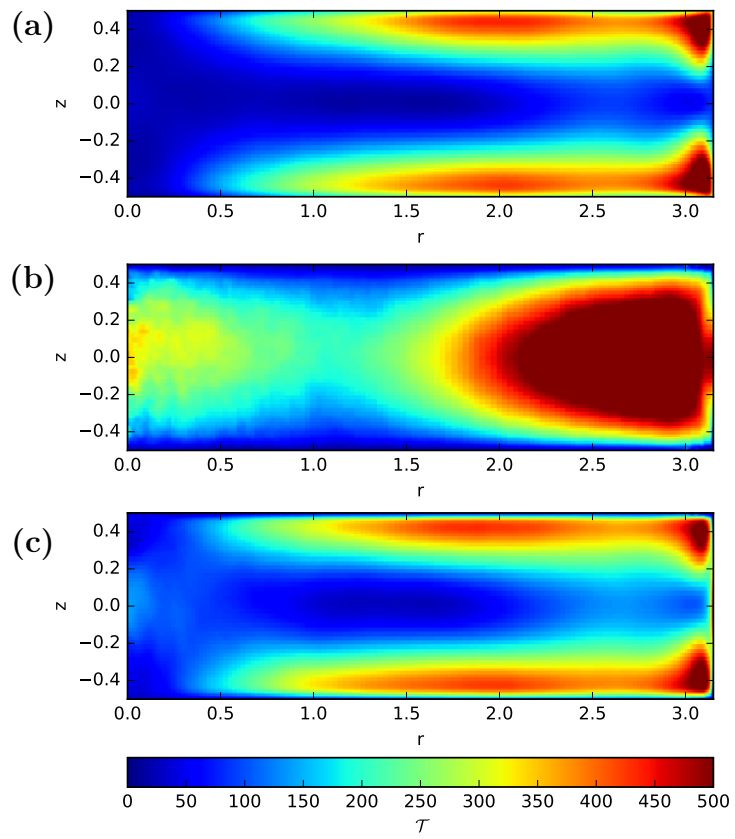


Figure 33: Spatially varying integral time scale based on the kinetic energy (a), temperature fluctuations (b) and total turbulent energy (c).

respectively. The r-z plots of $\mathcal{T}$ also give insight into the structure of the flow field by indicating which regions of the flow field have longer correlation times, and also by how much the correlation times vary.

Figure 33a) and figure 33(b) show very different behavior between the correlation of the kinetic and turbulent thermal energy fields. The two fields have very little overlap between regions with very long correlation times. The kinetic energy field has very long boundary layer correlation times, and the turbulent thermal energy field has very long bulk correlation times. The differing characteristics of the thermal

Mode (k)	Turb. Kinetic Energy	Turb. Thermal Energy	Total Turb. Energy
All	211	348	225
0	25	521	258
1	17.5	90.1	19.9
2	741	1050	786
3	181	226	181
4	47.1	34.4	47.0
5	17.2	11.9	16.4
6	47.1	46.9	45.6
7	12.7	9.34	12.2
8	7.61	6.47	7.83
9	11.7	7.30	11.3
10	11.9	6.54	11.4
11	7.53	5.04	7.31
12	5.20	4.41	5.22
13	3.72	3.21	3.69

Table 3: Area averaged integral time scale for the total field and a selection of Fourier modes in terms of t_f

and kinetic energy fields can be explained by their different properties. Virtually no temperature fluctuations occur within the thermal boundary layers and so it makes sense that there would be very little correlation in these regions. Conversely, the boundary layers along the top and bottom plates are are very highly correlated.

The regions where thermal and kinetic energy field fluctuations persist is inline with the location of the large-scale structures observed in figure 28. The turbulent thermal energy field in figure 33b) shows a large $\mathcal{T}$ in the region near the sidewalls where the up- and down-drafts occur and a fainter peak in the core region where the central up-draft resided when mode $k = 3$ dominated the large-scale structure. Since there has only been a small shift in the phase of the large-scale structures (see figure 31) these regions remain highly correlated.

The effect of the dominant Fourier modes is further illustrated in table 3 and figure 34

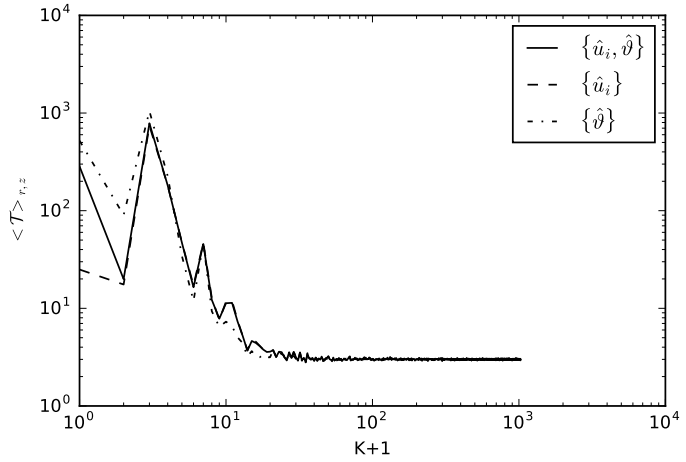


Figure 34: Modal integral time scales for each of the mode number integrated over the domain. Modes are plotted vs $k+1$ to make the $k=0$ mode visible on the log-scale plot.

where $< \mathcal{T} >_A$ is shown per mode number. The differences seen in table 3 and figure 34 show that thermal and kinetic energy based correlations differ in global magnitude as well as spatial structure.

Table 3 and figure 34 show that the turbulent thermal energy vector has a much higher correlation time for modes $k \in [0:3]$ than the kinetic energy field, but that the kinetic energy field has a larger correlation time for modes $k \in [4:12]$. Figure 34 also shows that $\mathcal{T}$ decays to a value of approximately $3t_f$ for all three energy vectors after the first 12 Fourier modes. This is the minimum limit that can be obtained with this data set since the snapshots were sampled $3t_f$ apart and much shorter $\mathcal{T}$'s are probable for the higher wave numbers.

In general, the total turbulent energy based $\mathcal{T}$ biases toward the kinetic energy based $\mathcal{T}$ since the kinetic energy vector comprises 3 of the 4 components in the total turbulent energy. However, the total turbulent energy vector still accounts for

contributions from the kinetic and turbulent thermal energy vectors and it will be used as the metric for determining $\mathcal{T}$ for the rest of this study.

5.6 Effects of the Inhomogeneous Spatial Directions

In the previous sections the effects of spatial inhomogeneity have been seen and lightly discussed. The large-scale structures that are so dominant in this flow are a result of spatial inhomogeneity and so any discussion of the large-scale structure is a discussion on inhomogeneity. However, in this section the effects will be analyzed more carefully by looking at the r-z variations in the autocorrelation, $\mathcal{T}$ and the Fourier spectra.

5.6.1 Spatial Inhomogeneity's Effect on Time Scales

In the previous sections $\mathcal{T}$'s tie to the large-scale structures in the flow field is discussed for various vectors (kinetic energy, turbulent thermal energy and total turbulent energy). The total turbulent energy based $\mathcal{T}$ in figure 33c) varies in both the r and z directions. To gain further insights into the extent of the variability the normalized autocorrelation is plotted vs snapshot spacing in figure 35 at a selection of points in the r-z plane.

Recall that the definition of $\mathcal{T}$ in this work is an integral of the normalized of the autocorrelation function (see equation 5.10) and so $\mathcal{T}(r, z)$ is equal to the area under the curve for each of the plots in figure 35(a). Figure 35(a) shows that the two probes in the highly correlated viscous boundary layer appear to be monotonically decaying, but have remained correlated over the entire data set.

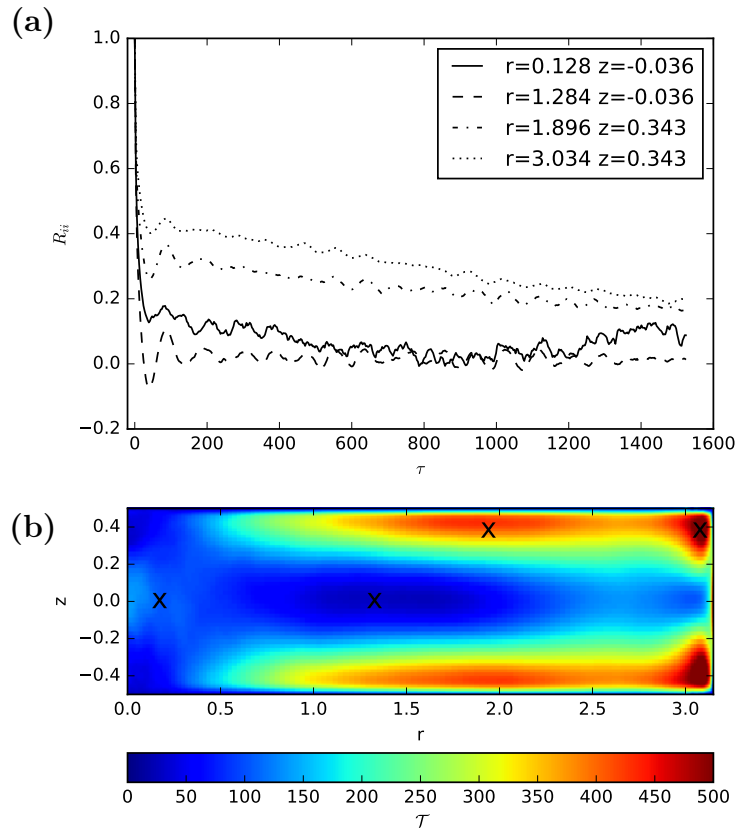


Figure 35: Temporal correlation at select points throughout the domain. Subplot (a) shows the correlation, and subplot (b) marks where the plotted correlations are with respect $\mathcal{T}(r, z)$.

The other two probes are taken at the mid-plane. The probe at $r = 1.284$ is at a local minimum in $\mathcal{T}$ and shows sufficient decay in R_{ii} to indicate the values become uncorrelated during this computation. The other probe at $r = 0.128$ is near a local maxima in $\mathcal{T}$. It shows signs of a long-lived transient as the correlation decays to zero with a separation time of approximately $800t_f$, but then begins to grow again. These results show a wide variation in behavior and convergence of $\mathcal{T}$ across the r-z plane due to the different the different physics that occur in the inhomogeneous directions.

Individual Fourier modes contain a different range of length scales and thus

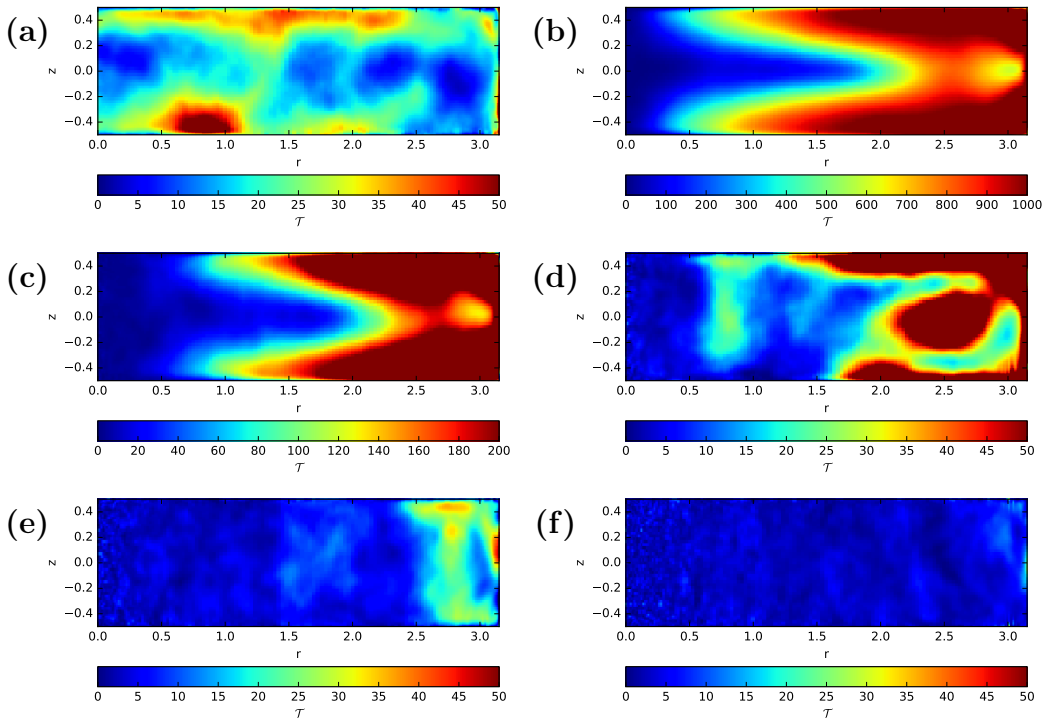


Figure 36: Spatially varying integral time scale based on total turbulent energy for modes $k = 1$ (a), 2 (b), 3 (c) ,4 (d), 10 (e) and 20 (d)

contribute to different portions of energy across the r-z plane. Additional insight into the spatial variance of $\mathcal{T}(r, z)$ can be found by investigating the contribution from the various Fourier modes. Recall that $R_{ii}(r, z, \tau)$ can be defined as a summation of $R_{ii}(r, k, z, \tau)$ over all k's. A $\mathcal{T}(r, k, z)$ field can be calculated for each Fourier mode giving an indication as to how the individual modes contribute in the total correlation. Plots of the total turbulent energy based $\mathcal{T}(r, k, z)$ for a selection of Fourier modes is provided in figure 36.

One observation of the subplots in figure 36 is that the globally dominant, highly correlated modes (subplot's b and c) show a high level of symmetry about the mid-plane, but the other modes do not. Subplots a,d,e and f also show much smaller peak

values for $\mathcal{T}$. The lack of spatial symmetry and smaller range of $\mathcal{T}$ indicate that these modes describe rare events in the flow field who have life spans much less that the length of the simulation, but much longer than the sampling rate of $3t_f$.

5.6.2 Spatial Inhomogeneity's Effect on Length Scales

In this section the spatial inhomogeneity's effects on length scales will be investigated by evaluating the time-averaged energy spectra at different r-z locations. Up to this point in the paper all data has been presented with respect to the azimuthal Fourier modes. The Fourier modes are identified by the mode number k which is the azimuthal frequency, and the energy from the Fourier modes at a given r-z location is essentially the integral with respect to θ along an azimuthal ring with constant radius. Therefore examining Fourier coefficients at different radii corresponds to different physical length scales and energy densities per unit length. A more consistent way to compare the flow structure at various locations in the flow field is to normalize the energy spectra and frequency with respect to a geometric length scale $\lambda_H = 2\pi r/k$. This is done by premultiplying the energy spectra with the radial location and plotting against λ_H's corresponding frequency $1/\lambda_H$. A sampling of the spectra at 7 different locations is provided in figure 37. These locations are at various points within the boundary layers (bottom plate and side walls) and bulk regions of the flow field to observe how the energy distribution changes in the regions where different physical phenomena dominate the flow field. $z = -0.45$ and $r = 3.1$ are within the viscous boundary layers for the bottom and and side walls respectively while $z = -0.4$ is just out side the viscous boundary layer in the vertical direction.

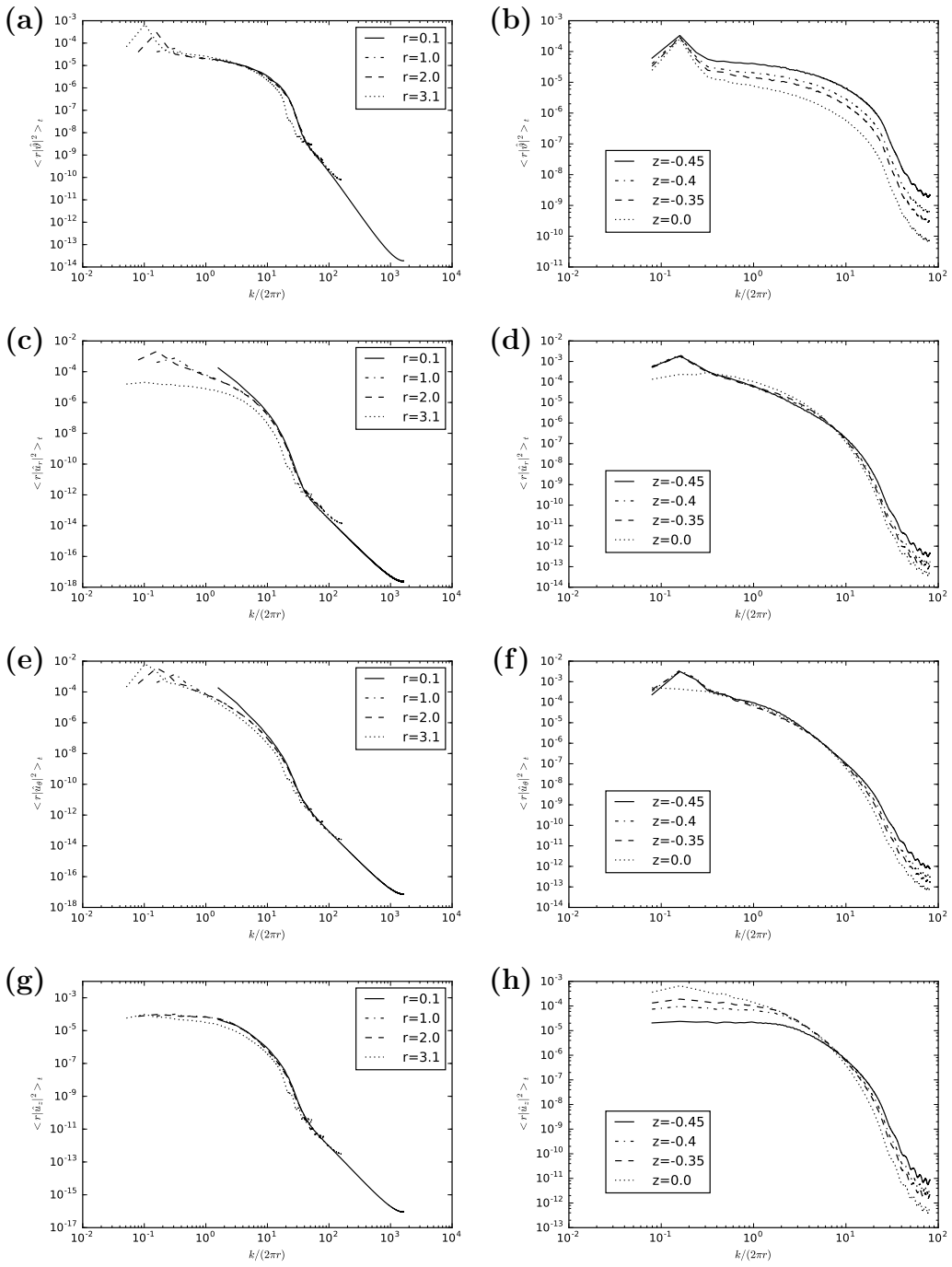


Figure 37: Time averaged energy spectra for each of the components in the total turbulent energy vector at various locations in the flow field. Subplots (a) and (b) are for the temperature field, (c) and (d) are for the radial velocity component, (e) and (f) are for the azimuthal velocity component and (g) and (h) are the vertical velocity component. Subplots on the left (a,c,d and e) are at a fixed height of $z = -0.4$, and plots on the right (b,d,f and g) are at a fixed radius $r = 2.0$.

5.6.2.1 Variations in Radial Location

Sampling at various radial locations with a fixed height shows excellent collapse across virtually all length scales for the spectra associated with ϑ, u_θ and u_z (see figure 37(a),(c),(e) and (h)). ϑ and u_θ show a poorer collapse for length scales that are greater than Γ ($\lambda_H > \Gamma$), and this behavior is also seen in the u_r plot. In fact, the collapse in u_r is also quite good when it is away from the geometric singularity $r = 0$ and the side wall. The lack of collapse in these two regions is not unexpected since u_r analytically must decay in these regions. It should also be noted that the same scaling behavior with respect to shifts in the r direction is also observed at $z = -0.45, -0.35$ and 0.0 as well.

Failure to collapse in the larger length scales can be attributed to the dominance of low order Fourier modes that have been shown to describe the flow field's large-scale structure. Fourier mode $k = 2$ contains a large amount of energy throughout the entire domain it will disrupt the collapse of the spectra since it affects a different length scale at each radii. In fact, if the spectra were to collapse across all length scales for all variables then it would be horizontally homogenous as in the canonical form of RBC with infinite Γ. In a sense the side walls of the convection cell act as a high pass filter because they limit the size of the largest length scales that can be observed in the flow. The fact that the $k = 2$ mode dominants the energy spectra at multiple length scales indicates that the underlying structure has a modal nature, and that it is the principle cause for radial inhomogeneity. This is most likely due to the confining, geometric effects of the cylinder. As Γ is increased it is expected that the various lower frequencies peaks in figure 37 will smooth out because more energy can be transported to larger length scales. This will allow the patterns to form

at their natural length scales, and become free from the geometric effects of the side walls. This presents itself as a metric to answer one of the basic questions regarding turbulent RBC: "how wide must Γ be to approximate the infinite Γ case?" The answer for RBC in a cylindrical domain being: "when the azimuthal spectra across all length scales collapses with respect to shifts in r."

5.6.2.2 Variations in Vertical Location

When the spectra is sampled over various heights at a fixed radius the behavior is virtually opposite to the fixed heigh case (see figure 37(b),(d),(f) and (g)). In the previous case ϑ and u_z's spectra showed the best collapse, but when the shift is vertical their collapse is considerably worse than u_r and u_θ. Additionally, u_r and u_θ show the best collapse at the lowest frequencies, and a poorer collapse at higher frequencies. In fact, divergence at high frequencies is seen for all three velocity components and the energy content decreases as the vertical position approaches the mid-plane. This is because the dissipative scales are removing more energy in the near wall region.

The spectrum for ϑ shows a strong collapse at the frequency associated with the $k = 2$ Fourier mode but a decay with increasing height for all other frequencies. The decay in the other frequencies is most likely due to the fact that the temperature fluctuations get increasingly weaker as they approach the mid-plane due to diffusion and turbulent mixing. The similar shape at each height indicates that the structure is not changing dramatically, but the energy content is. Conceptually this behavior is in line with a diffusion dominated process.

The spectra for u_z in figure 37g) shows some special characteristics that deserve a discussion of their own. Perhaps the most notable is that the spectra at the mid-plane

transitions from the smallest energy at high frequencies to the largest energy at lower frequencies. This inflection point occurs at a frequency of approximately 8.5 corresponding to a physical length scale of $0.118H$. Beyond the point of inflection there is a region where the spectra collapse for the vertical positions that are outside the viscous boundary layer. This region of collapse starts to break apart at a frequency of 2 corresponding to a length scale of $0.5H$ and the energy in frequencies less than 2 are the energy increases with vertical position.

5.7 Discussion and Conclusions

One of the primary questions of this work is how well Fourier modes represent the underlying physics of turbulent RBC. In a cylindrical domain Fourier decomposition can be applied in the azimuthal direction because it is analytically periodic. It has been shown that the large-scale structures in this study are very well described by a small selection of low-order Fourier modes across the entire domain. The azimuthal Fourier modes represent a different physical length scale at different radii, and this shows that the large-scale structures are strongly related to the modes because they are not restricted to one length-scale. The fact that the large-scale structure has an almost perfect alignment with the $k = 2$ Fourier mode as the simulation progresses is particularly telling.

As the Fourier wave number increases the relationship to the physical structures begins to decay. Evidence of this can be seen by examining the energy spectra in terms of physical length scale instead of mode number. It has been shown that the spectra collapses with respect to shifts in r across virtually every length scale below a threshold value that is approximately equal to Γ. This indicates a similarity among

the smaller length scales that is not dependent upon the mode number, and is less influenced by the cylindrical geometry. Based off these findings it can be concluded that the low-order, azimuthal Fourier modes are well aligned with the physics in the system, but the high-order modes are not. It can also be concluded that the large-scale structures have a large range of length scales associated with them and that are imposed by the nature of the cylindrical geometry.

Additional supporting evidence for the alignment of the large-scale structures and the low order Fourier modes is found in the area integrated Fourier coefficients. Examining these coefficients clearly indicates when a mode is responsible for the majority of the large-scale structure, and even provides insight into it's temporal behavior. For example, the transition between $k = 3$ and $k = 2$ dominate structures is clearly identifiable by the transition in the area integrated coefficients from having a constant magnitude and measurable phase, to a random phase and magnitude (or vice-versa). Also, the rotation that begins when the $k = 2$ mode becomes dominant is clearly depicted by the phase of the area integrated coefficients.

Additional insights into the effects of spatial inhomogeneity are identified by representing the energy spectra with respect to the physical length scales. The collapse of all variables with respect to shifts in r is across a much larger range of length scales than is necessary to enforce the modeling assumptions of for filtering methodologies such as large-eddy simulation. This indicates a connection between the horizontally homogeneous infinite Γ case and the inhomogeneous, horizontally confined case in this study. The small range of large length-scales that do not collapse indicates that the inhomogeneity is weak, and that as Γ increases the energy content will eventually saturate for large length-scales. It is predicted that the Γ for which this occurs will be sufficient to represent the infinite Γ case.

Vertical shifts for horizontal velocity components show a collapse for the energy spectra at small length-scales and a different scaling at large scales that is inline with the concept of a shear dominated boundary layer. The temperature and vertical velocity component spectrums show a behavior more inline with the conceptualization of plume generation through buoyancy. The temperature spectra decays as a function of height for all length-scales except the one that matches the dominant Fourier mode. The conceptual interpretation of this spectra is that as temperature transport is a diffusion dominated process across the layer depth except in the large-scale structures where the kinetic energy is sufficient to carry a large amount thermal energy across the layer depth. The vertical velocity component shows a decay with height for small length-scales, an increase with height for large length-scales and a region of collapse between these two regions. Decaying with height near the boundary layer aligns with the concept of the diffusion dominated near-wall regions, and the length scales where collapse begins to occur is approximately equal to the height of the viscous boundary layer. The region of collapse covers length-scales that range in size from the height of the boundary layer to half the layer depth and can be interpreted as the sizes corresponding to eddies that are not diffusion dominant, but also are too small to be associated with the large-scale structures. Obviously the larger length-scales must be associated with the large-scale structures and the increase in energy with vertical position can be conceptualized as the acceleration process that occurs while the collection of thermally charged plumes crosses the convection cell.

The correlation times associated with the large-scale structure have proven to extend well beyond the temporal range of this simulation. This has been shown by calculating the temporal correlations with respect to turbulent kinetic energy, turbulent thermal energy and turbulent total energy for the entire field as well as

the individual Fourier modes. The correlation times based on these three metrics shows differing magnitudes and structure with in the flow field indicating a strong dependence on spatial location. It has been shown that the integral time scale can vary by at least three orders of magnitude depending on the spatial location of the flow field which is the maximum separation that can be measure from the sampling rate and total time period of this study. The strongest correlations for the velocity components is in the boundary layers that are created by the large-scale structures. Conversely, the thermal correlation is strongest in the bulk region, but clearly both are associated with the large-scale structures. Based off the structure of the spatially varying integral time-scale and the observed rotation it seems that the unresolved correlation time aligns with the rate of rotation for the large-scale structure. This indicates that rotation of the large-scale structure must occur to sufficiently approximate the true Reynolds average when Γ is small enough that the geometric effects of the sidewalls affect the global organization of the structure. It is possible to imagine that at sufficiently large Γ global rotation is not required because the structures can shift in an out of phase on a local scale. Another mechanism that could speed up this process is the cessations noted by Brown *et al.* [12] and Mishra *et al.* [51]. The transition between $k = 3$ and $k = 2$ dominated structures is similar in nature to the cessations because a dramatic change in the global structure occurred over a very short time. However no other evidence of this sort of mechanism has been observed in this dataset or other large Γ studies to date.

In summary, it has been determined that the low order Fourier modes are well aligned with the physics of the large-scale structures in turbulent RBC, that the very largest length scales are responsible for inhomogeneity across the horizontal layers and that spatially varying correlation times support a need for global rotation of the

large-scale structures to generate uncorrelated samples of the flow field with respect to the true Reynolds average.

Chapter 6

PROPER ORTHOGONAL DECOMPOSITION OF FOURIER MODES

Proper orthogonal decomposition (POD) or the Karhunen-Loéve method is a technique that has been adopted by the turbulence community to decompose complicated turbulent signals into a series of linearly independent modes. The modes generated by POD are often referred to as empirical modes. This is because POD modes are not controlled by the governing equations or the boundary conditions of the problem, but rather from the data itself. POD is essentially an eigenvalue problem where the associated eigenvectors and eigenvalues are generated from a correlation matrix. There are two methods of POD that are often applied to fluid mechanics problems: classical (often referred to as the direct method) and the method of snapshots. Both methods maximize the energy over a series of instances of the flow with respect to a user defined norm. The two methods are mathematically equivalent for a finite number of snapshots, but the actual implementations differ significantly. Classical POD decomposes the time averaged two point spatial correlation while the method of snapshots decomposes a symmetric matrix formed from the inner-product of the various snapshots. In other words, classical POD forms modes from a spatial correlation, while the method of snapshots uses a temporal correlation. An extensive source on POD, dynamical systems and other related subjects can be found in the text of Holmes *et al.* [37].

This is not the first work to utilize POD for analyzing turbulent thermal convection, see Sirovich and Park [68] and Bailon-Cuba *et al.* [7] among others. What sets this work apart from the previous cases is that POD is employed on the individual Fourier

modes to further investigate the structure and energy content within the flow field. The goal of performing POD independently on each set of Fourier coefficients is to provide additional insight into structure of flow field in the inhomogeneous r and z directions. If POD is solely used to evaluate the entire three-dimensional flow field for RBC in a cylinder then the unsurprising result is a series of low order modes that strongly resemble Fourier modes. This is because POD only seeks to maximize the energy over the supplied domain with respect to a given norm, and as shown in the previous chapter, Fourier modes describe the highly-energetic large-scale structures very well.

However, proceeding the POD with a Fourier decomposition in this geometry focuses the POD process entirely onto the structures that reside in the inhomogeneous r-z plane, and because both forms of decomposition are linear, the modes can still be summed with appropriate weighting coefficients to recreate the original field. One item of interest is to see if POD will reveal further global structure in the dominant Fourier modes $k = 2$ and 3, or if it extract unseen structure from the other Fourier modes which show a higher degree of randomness in the temporal evolution of the area integrate Fourier coefficients. Another item of interest in this chapter is to identify structures and length scales in the r-z plane, and both topics are addressed in the later sections of the chapter.

Since the POD modes are a subset of the Fourier modes the symbol m will be used to indicate the POD mode number, and k will continue to be used for the Fourier mode number. The method of snapshots is employed, and a brief introduction to its nuances is provided in the following paragraphs. The POD calculations are performed using the open source python library modred (see Belson *et al* [10]) whose documentation also serves as a short primer to POD and other, similar techniques.

6.1 Method of Snapshots

The method of snapshots was first introduced by [69]. The basic premise of the method of snapshots is that the empirical POD modes ($\phi_i(\Omega)$) can be interpreted as a weighted sum of M statistically independent snapshots of a given vector field ($\mathbf{V}(\Omega, t^i)$). The vector field in $\mathbf{V}$ is not a physical vector field such as velocity, but rather the linear algebra sense. In other words $\mathbf{V}$ is a list of the degrees of freedom that are being studied.

$$\phi_j(\Omega) = \sum_{i=1}^{M} A_{ij}\mathbf{V}(\Omega, t^i) \tag{6.1}$$

Determining the A coefficients in equation 6.1 is done by finding the eigenvalues and vectors of the correlation matrix Q whose entries are the inner-products of the snapshots with one another.

$$Q_{ij} = \sum_{i=1}^{M}\sum_{j=1}^{M}(\mathbf{V}(\Omega, t^i), \mathbf{V}(\Omega, t^j)) \tag{6.2}$$

From here the eigenvalue problem in equation 6.3 can be solved:

$$Q_{ij}\psi_{ij} = \lambda_i\psi_{ij} \tag{6.3}$$

where ψ_{ij} and λ_i are the respective eigenvectors and eigenvalues of Q_{ij}. Note that if the values of $\mathbf{V}$ are real then Q_{ij} is symmetric, and in the most generalized sense Q_{ij} is Hermitian. This symmetry property is due to the nature of Q_{ij}'s construction and as a result all of its eigenvalues must real.

The coefficients A in 6.1 are generated from the eigenvectors from equation 6.3. Orthonormality can be ensured in the modes by using the relationship between the eigenvectors, eigenvalues and snapshot matrix shown in equation 6.4.

$$\phi_i = \mathbf{V}(\Omega, t^j)\psi_{ij}(M\lambda_j)^{-1/2} \tag{6.4}$$

The modes generated by equations 6.1-6.4 are driven by three critical parameters: 1) the definition of the vector $\mathbf{V}$, 2) the implementation of the inner product, and 3) the choice of snapshots. Since the method of snapshots maximizes the energy in terms of the inner product $(\mathbf{V}, \mathbf{V})$, the choice of $\mathbf{V}$ is critical to determining the shape of the modes and requires little explanation. In this work the method of snapshots is employed independently on each Fourier mode using the total turbulent energy for $\mathbf{V}$.

The inner product over a finite interval is formally defined as,

$$(\mathbf{V}, \mathbf{V}') = \int_\Omega \mathbf{V}^* \mathbf{V}' \partial\Omega \tag{6.5}$$

and so the manner in which the spatial integral is evaluated has a large impact on the construction of the Q_{ij}. In this work the inner product is defined by equation 6.6

$$(\mathbf{V}, \mathbf{V}') = \sum_{j=1}^{N_z} \sum_{i=1}^{N_r} w_i w_j \mathbf{V}(r, k, z, t)^* \mathbf{V}'(r, k, z, t') r J \tag{6.6}$$

where the w's are the Gauss-Legende quadrature weights, and J is the Jacobian associated with the transformation from the unit interval $[-1 : 1]$ to the physical domain size. However, the choice of snapshots is a less clear issue and is discussed in the next section.

6.2 Choosing Snapshots

When Sirovich first introduced the method of snapshots in 1987 he clearly stated that the snapshots should be sampled at a time approximatly equal to or greater than the correlation time of the flow and that a sufficiently large number of snapshots should be included [69]. The time-averaged, two-point correlation ($\mathbf{K}(\mathbf{x}, \mathbf{x}')$) generated from these snapshots can be used to approximate the classical POD kernel. However,

in practice $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is degenerate and so its eigenvectors can be represented as a linear combination of the original vectors [69]. This degeneracy is where the method of snapshots finds its origin.

Sirovich's two requirements for POD snapshots, uncorrelated and sufficient quantity, are in direct competition for numerical simulations because the cost of generating a sufficient number of uncorrelated snapshots is very high.

The common metric used to determine if snapshots are far enough apart is a temporal spacing of $2\mathcal{T}$. It has been shown in the previous chapter that $\mathcal{T}$ varies by large amounts through out the domain and that choice of vector will also affect $\mathcal{T}$ (see section 5.5.2). Since the snapshot correlation matrix is constructed from an inner product over the spatial domain, the varying levels of correlation will have a global impact on the computation. This makes the question of what spacing to employ some what less clear.

Two different snapshot spacings are compared to evaluate the sensitivity of the modal structure to this parameter. The first snapshot spacing is defined using $\mathcal{T}$ as defined in equation 6.7. This definition of $\mathcal{T}$ corresponds to the values in table 3 and figure 34.

$$\mathcal{T}(k) =< \int_{\tau} \frac{R_{ii}(r, k, z, \tau)}{R_{ii}(r, k, z, 0)} \partial\tau >_{r,z} \tag{6.7}$$

This definition acts as a global estimate for $\mathcal{T}$ which is inline with the global correlation metric used by the method of snapshots. The second is an ad-hoc spacing of $6t_f$ which accounts to every other stored data point in the dataset. Each of these sampling rates has complementary advantages and disadvantages. Spacing the snapshots $2\mathcal{T}$ apart provides the most assurance that the snapshots are uncorrelated, but there is a concern about if a sufficient number of snapshots are present. For example, snapshots from $k = 2$ will need to be spaced 1572 t_f apart based on the estimate of $\mathcal{T}$ in table 3.

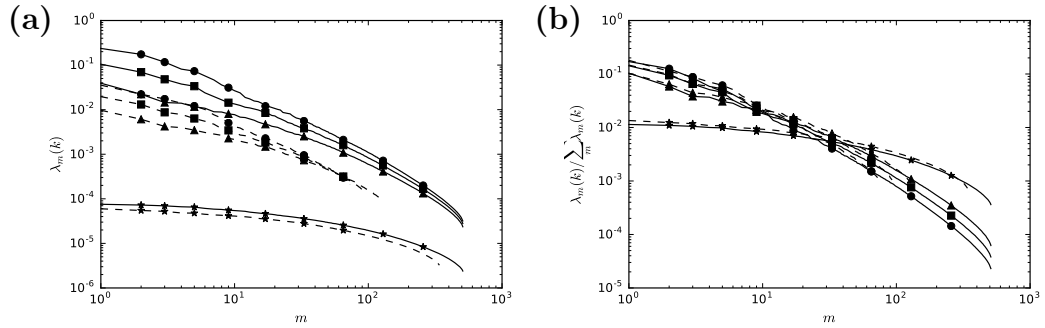


Figure 38: Eigenvalues (a) and normalized eigenvalues (b) corresponding to the POD modes for Fourier wave numbers $k = 1$ ($\circ$), 5 ($\square$), 10 ($\triangle$) and 100 ($\star$). The dashed lines represent the modes from sampling snapshots at $2\mathcal{T}$ and the solid lines are from sampling at $6t_f$.

This represents the most extreme case, and it is clear that this is not a sufficient number of snapshots to compute converged POD modes. Spacing the snapshots $6t_f$ apart ensures that a sufficient number of snapshots are provided, but the correlation between snapshots is in question. Comparison between these two sampling rates is performed for Fourier wave numbers that span three orders of magnitude so that the difference between $2\mathcal{T}$ and $6t_f$ also spans a large range.

Figure 38 shows a sample of the eigenvalues for POD modes calculated from the two different sampling rates. The un-normalized modes in figure 38(a) show that the largest eigenvalues from the $6t_f$ sampling rate have a larger magnitude, but when the eigenvalues are normalized by the total energy in the two cases collapse on top of one another perfectly. It is not surprising that the $6t_f$ sampled case has higher magnitude eigenvalues since it effectively spans a higher dimensional space than the $2\mathcal{T}$ sampled data set. In fact, the correlation matrix for the $2\mathcal{T}$ sampled dataset ($Q_{\mathcal{T}}$) is a subspace of the $6t_f$ correlation matrix (Q_{t_f}). Since both correlation matrices

are Hermitian it can be shown via the Courant-Fischer minimax principle that the eigenvalues of $Q_{\mathcal{T}}$ are related to the eigenvalues of Q_{t_f} by

$$\lambda_k(Q_{t_f}) \leq \lambda_k(Q_{\mathcal{T}}) \leq \lambda_{k+n-r},\ 1 \leq k \leq r \tag{6.8}$$

where r is the rank of $Q_{\mathcal{T}}$, n is the rank of Q_{t_f}, and $\lambda_k(\cdot)$ represents the k smallest eigenvalue of the correlation matrix. Equation 6.8 proves that the eigenvalues of a Hermitian space will always bound the eigenvalues of a principle subspace so that the nesting behavior of figure 38(a) will be repeated for a reduction in sampling frequency over a given interval.

However, the collapse of the normalized eigenvalues seen in figure 38(b) is not assured by equation 6.8,. This shows that the relative energy distribution between modes is the same between both cases and this collapse is a sign that the modes generated by both datasets might be similar. It should also be noted that this collapse is observed for all values of k, but only a small sample are shown in figure 38(b).

A side by side comparison of the first 5 POD modes from three different Fourier wave numbers is presented in figures 39-41. The total turbulent energy is compared since this is the quantity that POD is maximizing. Visual inspection shows that there is strong agreement between the spatial structure and magnitude between the modes even though the sampling rate is different.

A more quantitative comparison can be performed by projecting the POD modes from the two different sampling rate sets onto one another. The projection is performed by taking the inner product of the modes from each dataset and because both datasets are orthonormal basis the projection is already normalized. This serves as an estimate

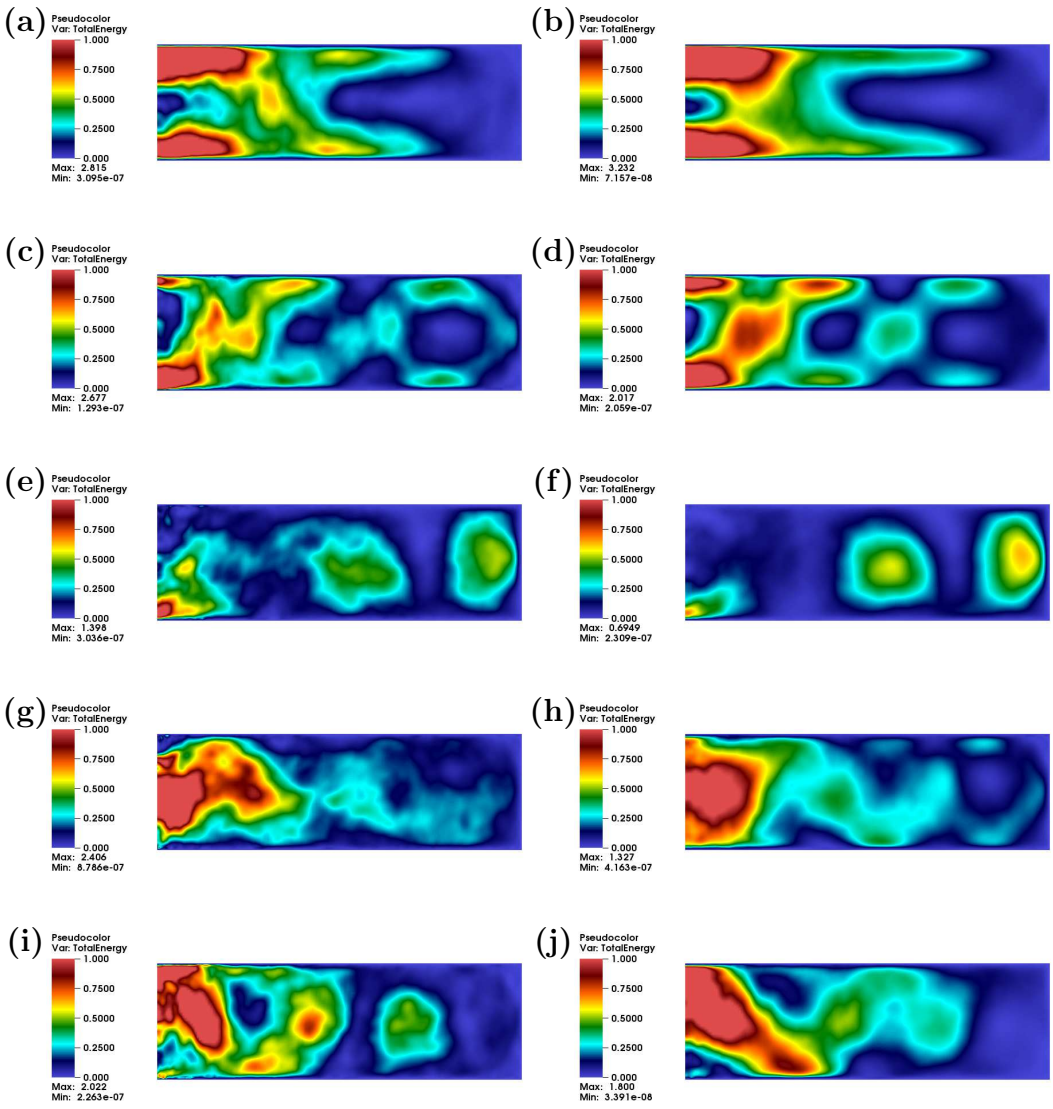


Figure 39: Comparison of POD modes generated from different sampling rates ($2\mathcal{T}$ left, $6t_f$ right): $m = 1$ (a,b), $m = 2$ (c,d), $m = 3$ (e,f), $m = 4$ (g,h) and $m = 5$ (i,j). These modes are generated from the Fourier coefficients for wave number $k = 1$, and total turbulent energy in the r-z plane is the plotted quantity.

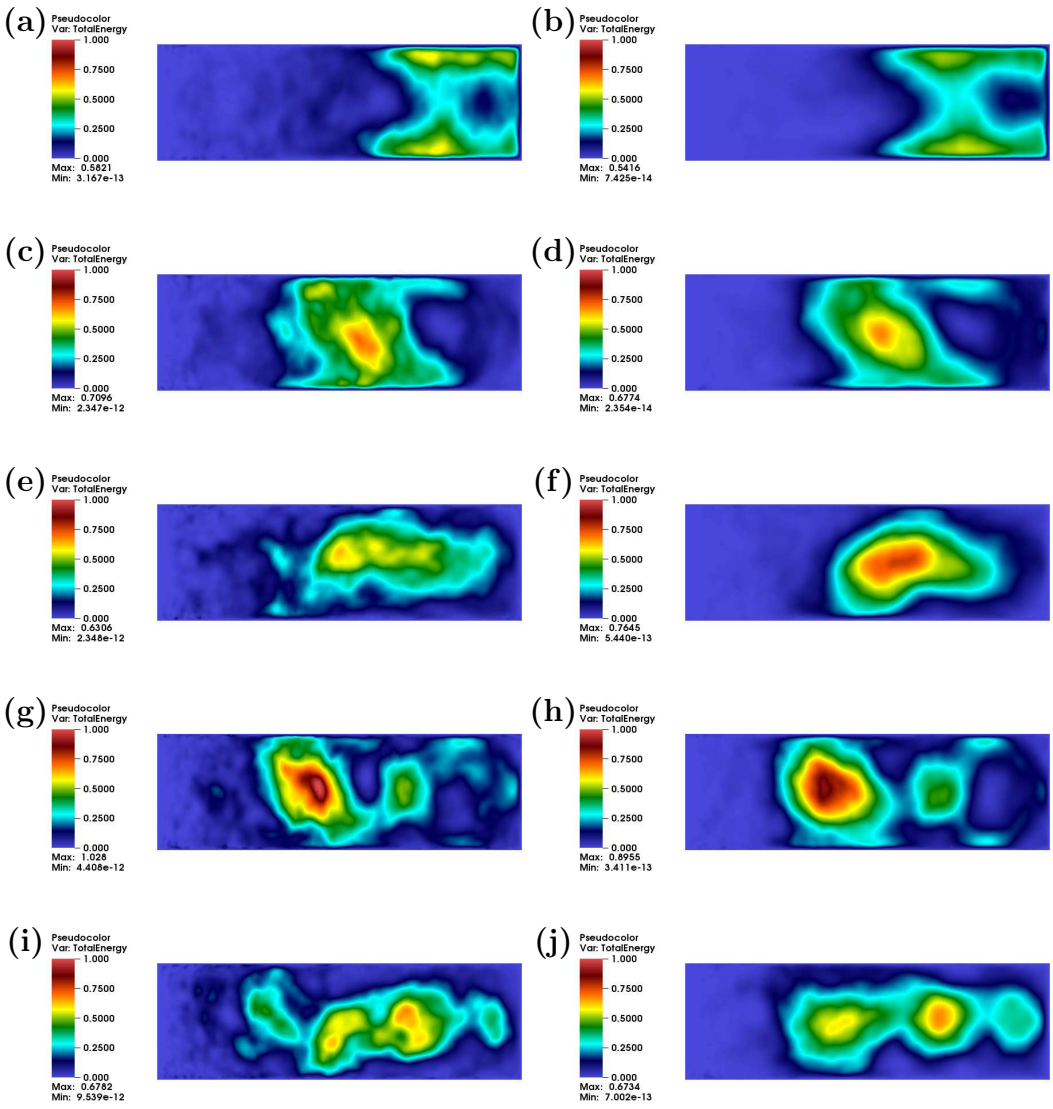


Figure 40: Comparison of POD modes generated from different sampling rates ($2\mathcal{T}$ left, $6t_f$ right): $m = 1$ (a,b), $m = 2$ (c,d), $m = 3$ (e,f), $m = 4$ (g,h) and $m = 5$ (i,j). These modes are generated from the Fourier coefficients for wave number $k = 5$, and total turbulent energy in the r-z plane is the plotted quantity.

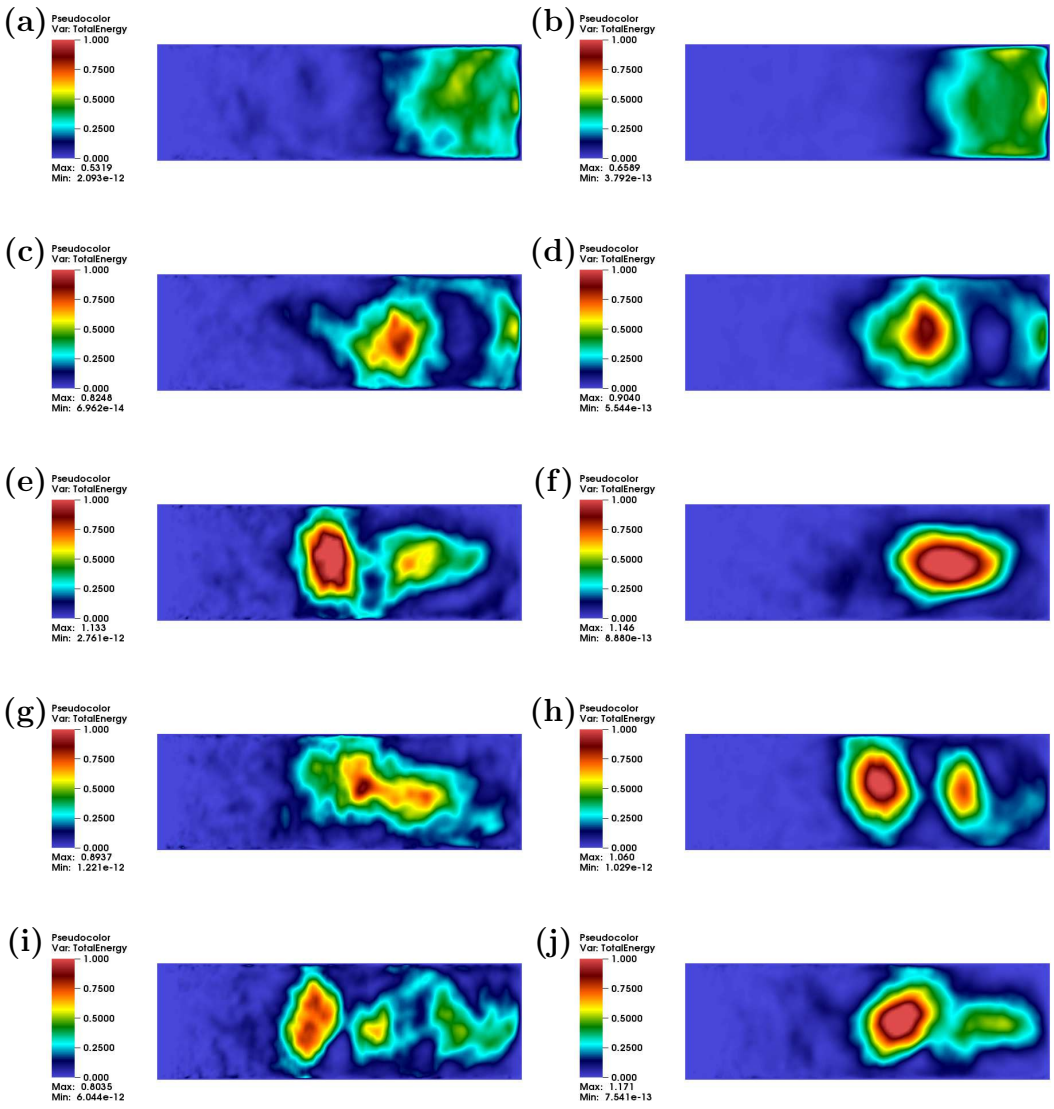


Figure 41: Comparison of POD modes generated from different sampling rates ($2\mathcal{T}$ left, $6t_f$ right): $m = 1$ (a,b), $m = 2$ (c,d), $m = 3$ (e,f), $m = 4$ (g,h) and $m = 5$ (i,j). These modes are generated from the Fourier coefficients for wave number $k = 10$, and total turbulent energy in the r-z plane is the plotted quantity.

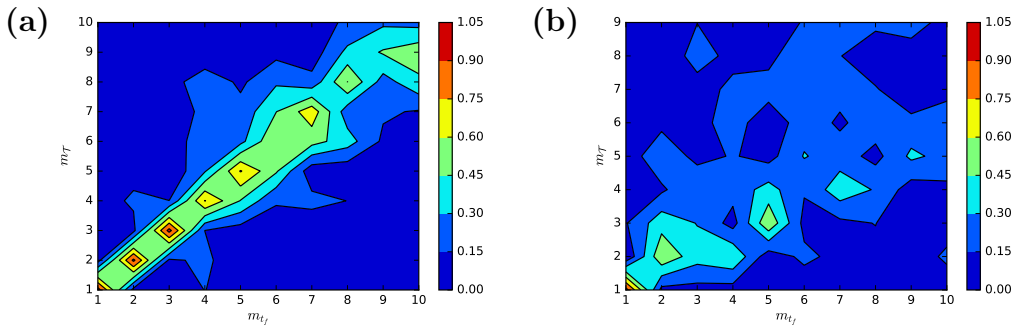


Figure 42: Absolute value of the L2 norm between first 10 POD modes sampled at $6t_f$ and $2\mathcal{T}$. The snapshots are taken from the Fourier wave numbers $k = 1$ (a) and $k = 3$ (b).

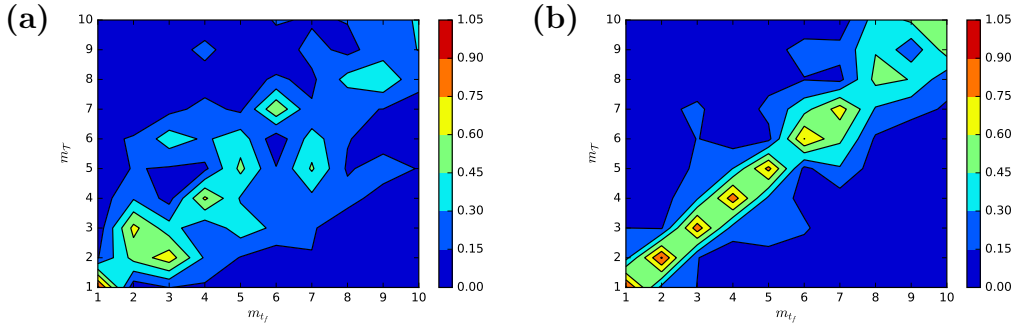


Figure 43: Absolute value of the L2 norm between first 10 POD modes sampled at $6t_f$ and $2\mathcal{T}$. The snapshots are taken from the Fourier wave numbers $k = 4$ (a) and $k = 5$ (b).

of the L2 norm between any two modes. The resulting value is complex and so there is a representative amplitude and phase shift for the inner product of any two modes. The amplitude, $|(\phi_{\mathcal{T}}, \phi_{t_f})|$, is plotted in figures 42- 44 since this quantity is indicative of the spatial alignment of energy in the r-z plane which is the primary quantity of interest in this study.

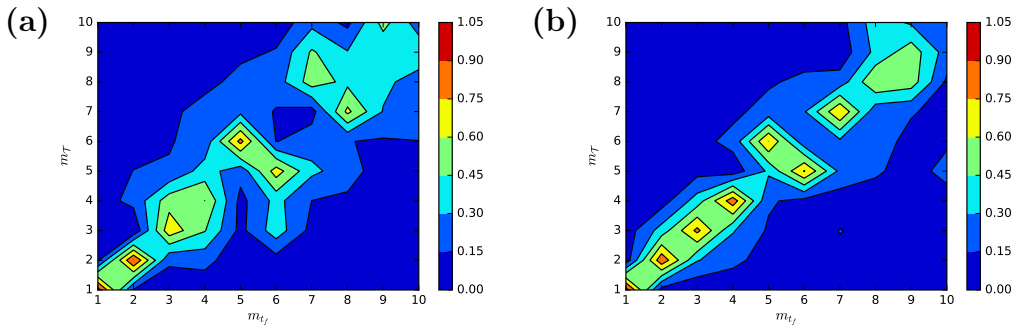


Figure 44: Absolute value of the L2 norm between first 10 POD modes sampled at $6t_f$ and $2\mathcal{T}$. The snapshots are taken from the Fourier wave numbers $k = 10$ (a) and $k = 20$ (b).

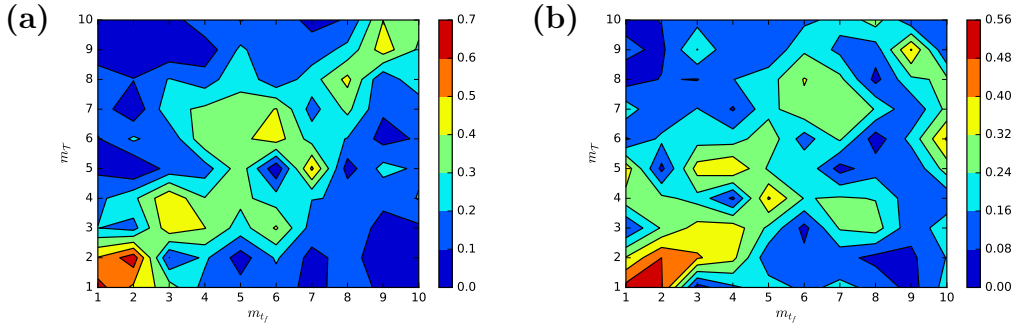


Figure 45: Absolute value of the L2 norm between first 10 POD modes sampled at $6t_f$ and $2\mathcal{T}$. The snapshots are taken from the Fourier wave numbers $k = 60$ (a) and $k = 100$ (b).

Figures 42- 45 show varying alignment between the POD modes that are calculated over the two datasets. The low-order Fourier modes that have relatively small values for $\mathcal{T}$ show strong alignment between the both sampling sets. High-order Fourier modes and Fourier modes where $\mathcal{T}$ is large show much poorer alignment. One might expect the higher-order Fourier modes to have good alignment since the sampling rate is very close between both sets. However, a closer inspection of the eigenvalues provides an explanation for this departure.

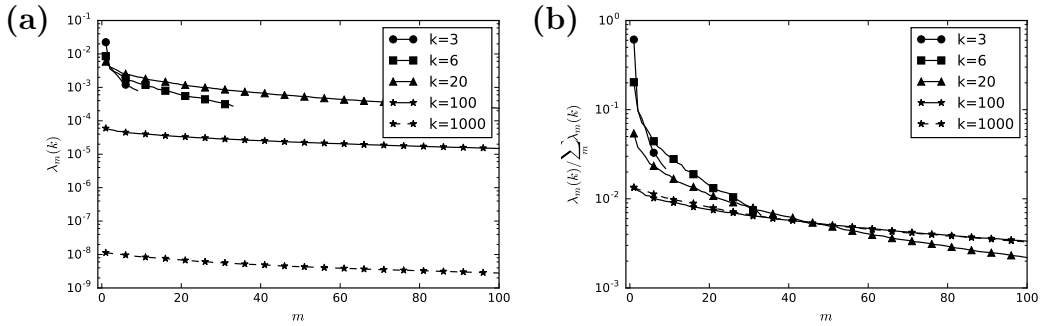


Figure 46: Eigenvalue spectrum for POD modes (snapshot sampling rate of $2\mathcal{T}$) for a selection of Fourier modes (a), and the normalized eigenvalue spectrum for the same selection of Fourier modes (b)

Figure 46 shows the POD eigenvalues for different values of k. The eigenvalues for $k = 100$ and $k = 1000$ are very flat indicating that a very large number of POD modes are necessary to represent a significant portion of the energy. This also indicates that the POD modes are not very distinct and so a slight difference in sampling can dramatically affect the structure of the modes.

Inspection of the modes generated from both sampling rates indicates that the $2\mathcal{T}$ modes appear to be similar too the $6t_f$ modes, but less converged. Since the method of snapshots constructs its correlation matrix from the inner product of the individual snapshots if the spacing is too close it will lead to a poorly conditioned, or singular matrix. However, the only Fourier wave numbers that have a truely large $\mathcal{T}$ are $k = 2$ and 3. Since the POD for these two wave numbers is inherently suspect, the author has decided to use the $6t_f$ spacing for the rest of this chapter and to document a selection of modes in Appendix B to take advantage of the increased convergence.

6.3 Temporal Evolution of the POD Projections

One of the objectives that has been outlined for the POD analysis in this chapter is to determine if additional structure exists in the temporal evolution of the Fourier modes. Structure is found in a global sense for Fourier modes 2 and 3 when the spatially integrated Fourier coefficients are plotted as a function of time (see figure 31). In a similar sense the POD modes can be used to see how the global structure evolves in time by projecting the time series data onto the POD modes. This projection is performed by taking the inner product of a Fourier mode's snapshot with its associated POD modes and the corresponding value is the amount of energy represented by the POD mode at that instance in time.

$$E_i(k,t) = (\mathbf{V}(r,k,z,t), \phi_i(r,k,z)) \tag{6.9}$$

Equation 6.9 represents a global quantity that is akin to the spatial integrated Fourier coefficients in figures 31 and 32. The first Fourier modes to be investigated via POD are the two that were determined to be globally dominant i.e. $k = 2$ and 3. Figure 47 shows the projection of POD mode $m = 1$ onto the time series for $k = 2$ and 3 Fourier modes. Figure 47 shows that the $m = 0$ POD mode bears a strong similarity to the spatially integrated Fourier coefficients. It should also be noted that the transition time between $k = 3$ to $k = 2$ dominance in the large-scale structure takes almost exactly 10 eddy-turnover time units to complete.

The key difference between the equation 6.9 and the spatial integrated Fourier coefficients is that the r-z structure from the POD projection will not change over time because it is a fixed mode. Even though POD mode $m = 1$ for $k = 2$ is not a converged POD mode, it can still be used as a template to quantify the rate of

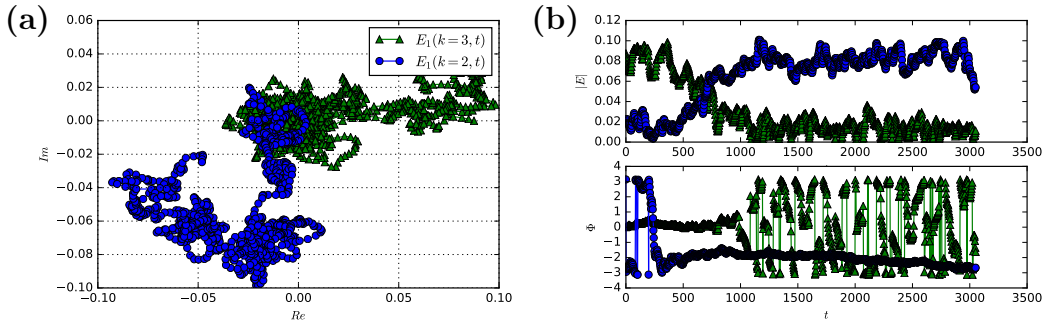


Figure 47: Projection of the time series onto the first POD mode ($m = 1$) for $k = 2$ and $k = 3$ Fourier modes expressed in the complex plane (a) and as phase and amplitude (b).

rotation when $k = 2$ dominates the large-scale structure. The spatially integrated Fourier coefficients are not as well suited for quantifying the rotation of the large-scale pattern because the r-z structure changes with time. Using a least-squares regression to measure the rate of change in the phase of $k = 2$, $m = 1$ it is estimated that the large-scale structure rotates at a rate of approximately $3.7 \times 10^{-}2 deg/t_f$ or $1.1 deg/t_\epsilon$.

Unfortunately, $\mathcal{T}$ for the $k = 2$ mode is very large and only two snapshots are considered uncorrelated over the available time series by the metric in equation 6.7 and so no additional insight into the structure of the Fourier mode can be trusted. However, $\mathcal{T}$ for the $k = 3$ Fourier mode is a bit better with 9 uncorrelated snapshots. Figure 48 displays projections of the first 4 POD modes for $k = 3$ onto the time series data.

The results in figure 48 show that the structure in the complex plain provided by the temporal evolution of the Fourier mode $k = 3$ is strictly contained in the first POD mode, and that all other POD modes for $k = 3$ exhibit the characteristic distribution of the non-dominant Fourier modes. Additionally, no obvious structure

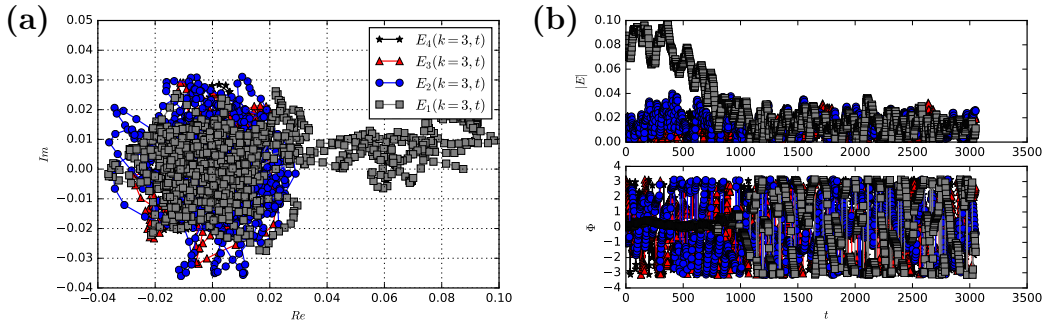


Figure 48: Projection of the time series onto POD modes for $k = 3$ Fourier modes expressed in the complex plane (a) and as phase and amplitude (b).

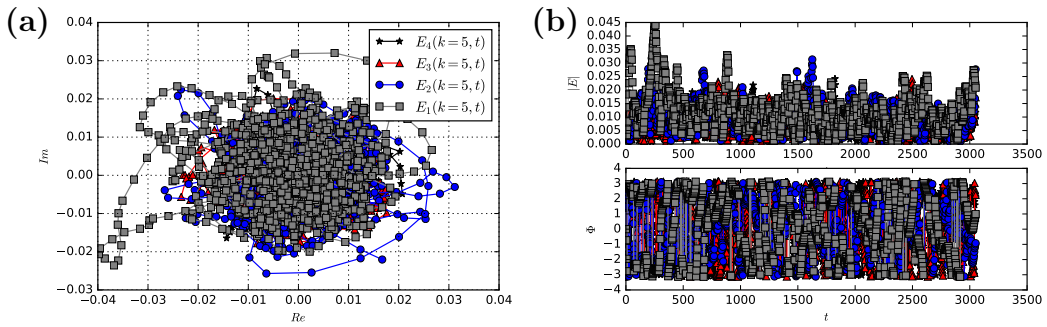


Figure 49: Projection of the time series onto POD modes for $k = 5$ Fourier modes expressed in the complex plane (a) and as phase and amplitude (b).

in the temporal evolution of the non-dominant Fourier modes is revealed through the POD, and this illustrated in figure 49 for Fourier mode $k = 5$. The net conclusion from this analysis is that the POD does not expose any macro dynamics that were not already seen in the individual Fourier modes, and that the dynamics that were seen are fully contained in the $m = 1$ POD modes.

6.4 Spatial Structure of the POD Modes

The next area of investigation is the spatial structure of the individual POD modes. Since all the POD modes are a decomposition of Fourier coefficients, they can be projected back into real space by performing an inverse Fourier transform. Figure 50 provides a visualization of the first POD mode for Fourier modes $k = 1 : 9$. The visualizations in figure 50 are useful for understanding the global structure of the modes in terms of the primary variables (velocity and temperature). A dictionary has been provided in Appendix A for a larger sampling of POD modes.

A single striking feature throughout all $m = 1$ modes in figure 50 is the prevalence of roll cells. All modes except the $k = 1$ feature roll-cells along the side walls. Interestingly, the mode corresponding to $k = 1$ contains a roll-cell in the center of the domain that resembles the familiar "wind of turbulence" which is prominent feature in smaller Γ domains. This singular roll-cell in $k = 1$, $m = 1$ extends to a radius of approximately $1.5H$ and the nodes of the roll cells are located at a radius approximately equal to H. The fact that the most energetic POD mode for $k = 1$ is almost identical to the dominant large-scale structure seen in the low Γ studies shows that the structure of the unit Γ case is still present at larger Γ. This indicates that there may be a natural length scale for each of the modes, and that the energetic structures are not required to attach to the sidewalls as seen in all the other $m = 0$ modes. However, the spectra in figures 26 and 37 clearly shows that the majority of the energy has migrated toward larger wave numbers and longer length scales.

As the Fourier mode increase the POD modes in figure 50 show finer scales in the central region. This is not entirely surprising since the length scales associated with the Fourier wave number are directly proportional to r and inversely proportional

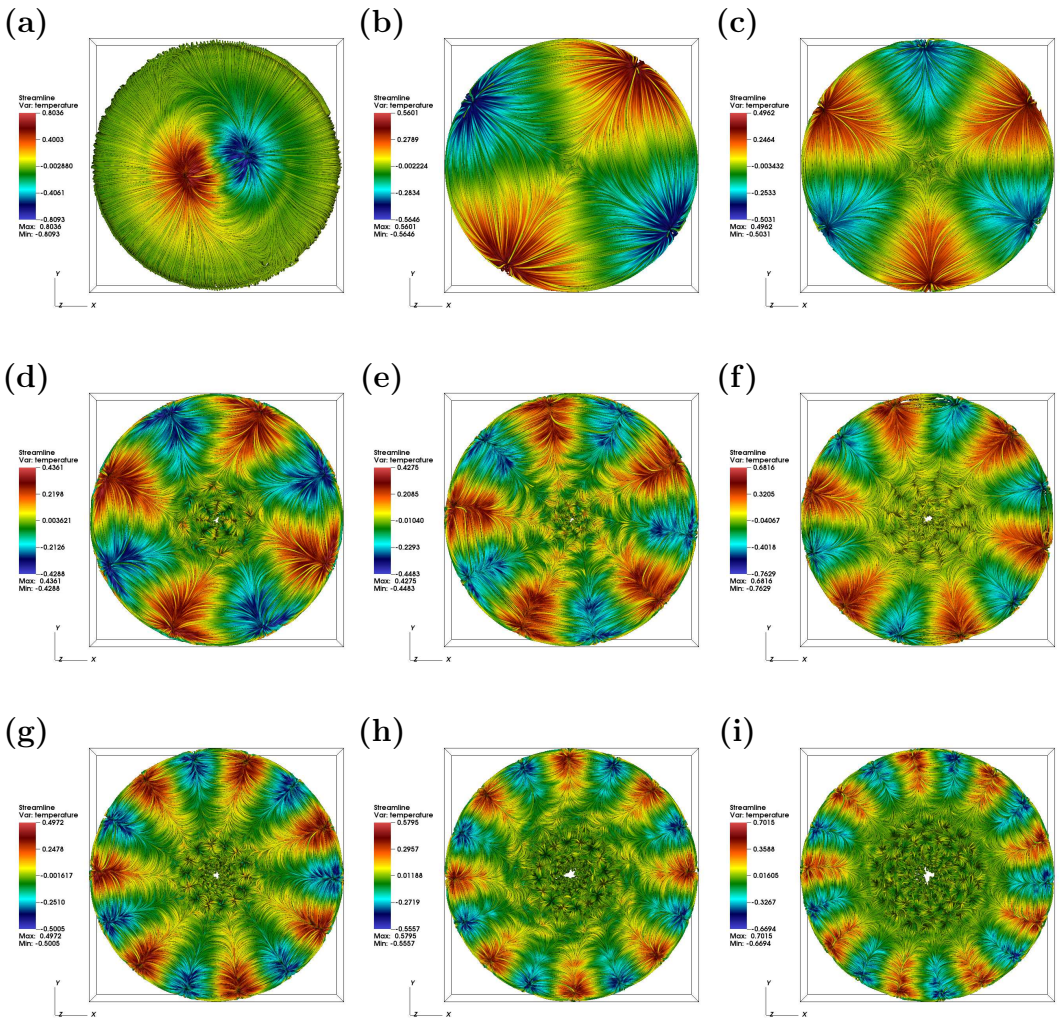


Figure 50: Velocity streamlines colored by temperature for the first POD mode transformed to real space for the $k \in [1:9]$ Fourier modes corresponding to subplots [(a):(i)] respectively.

to k. However it is worth noting that the distinct seperation of scales first starts to take place with $k = 4$ which is the first Fourier wave number after the geometrically dominant modes. These finer velocity scales also have smaller temperature ranges showing a similitude to the thermal plumes, and as k grows large enough the velocity structures cease to exist.

The dictionary in Appendix A shows 1 period of each mode so that the detailed structure can be seen more clearly. While the images of streamlines in figure 50 are excellent at communicating the global structure of the velocity and temperature fields, it is still difficult to see the r-z variations across the domain in each mode due to the complicated nature of the plots. Plots of the total turbulent energy such as figures 39-41 provided a compact way to visualize the r-z variations across the entire domain. The r-z distribution of total turbulent energy in the first three POD modes are provided in figures 51-53 for a sample of Fourier wave numbers.

Some general observations can be drawn from the visualizations in figures 39-41 and 51-53. First, the $m = 1$ POD modes in this data set feature a large amount of energy in the boundary layers. For the $m = 1$ modes associated with higher k the energy is clustered near the side wall, and for lower k it is clustered near the top and bottom plates. As m increases the energy begins to be concentrated in the bulk region, and the number of concentrations tends to increase with m. In general the division of energy in the r direction seems to be favored over divisions in z.

Physically, it makes sense for the highest energy POD modes to focus on the boundary layers because that is where kinetic energy has its highest concentration. The boundary layers are where the horizontal velocity components are strong, and it

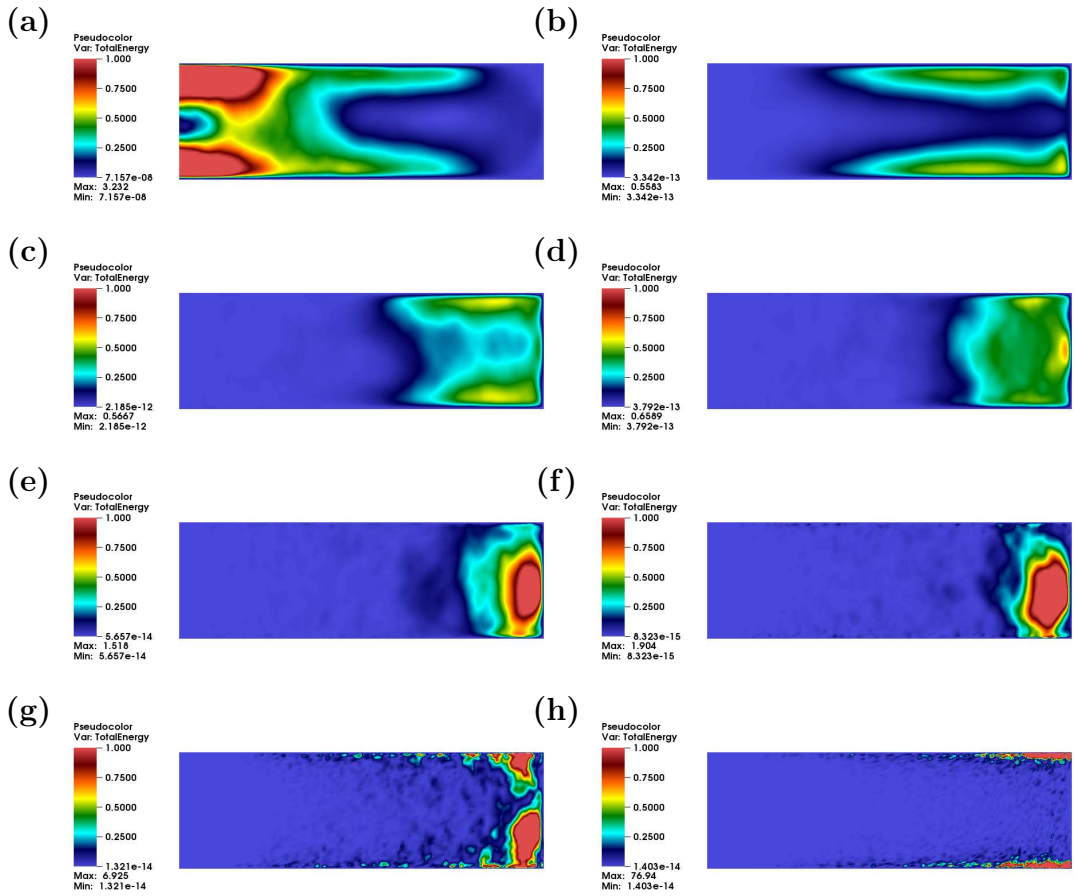


Figure 51: Total turbulent energy distribution in the r-z plane for the first POD mode for $k = 1$ (a), 3 (b), 6 (c), 10 (d), 20 (e), 40 (e), 100 (f) and 1000 (g).

is also where plumes are gain and deposit their kinetic energy. Subsequently, POD modes with less energy are concentrated in the bulk, where less energy resides. The increase of division in the bulk region's energy concentrations as the mode number increases is inline with a physical concept. This is the concept of an energy cascade where the energy from larger scales is transported to smaller scales. The fact that the division tend to favor the r direction is also inline with the physics because the r direction sees little viscous effects from the wall, and is less inhomogeneous than the z direction.

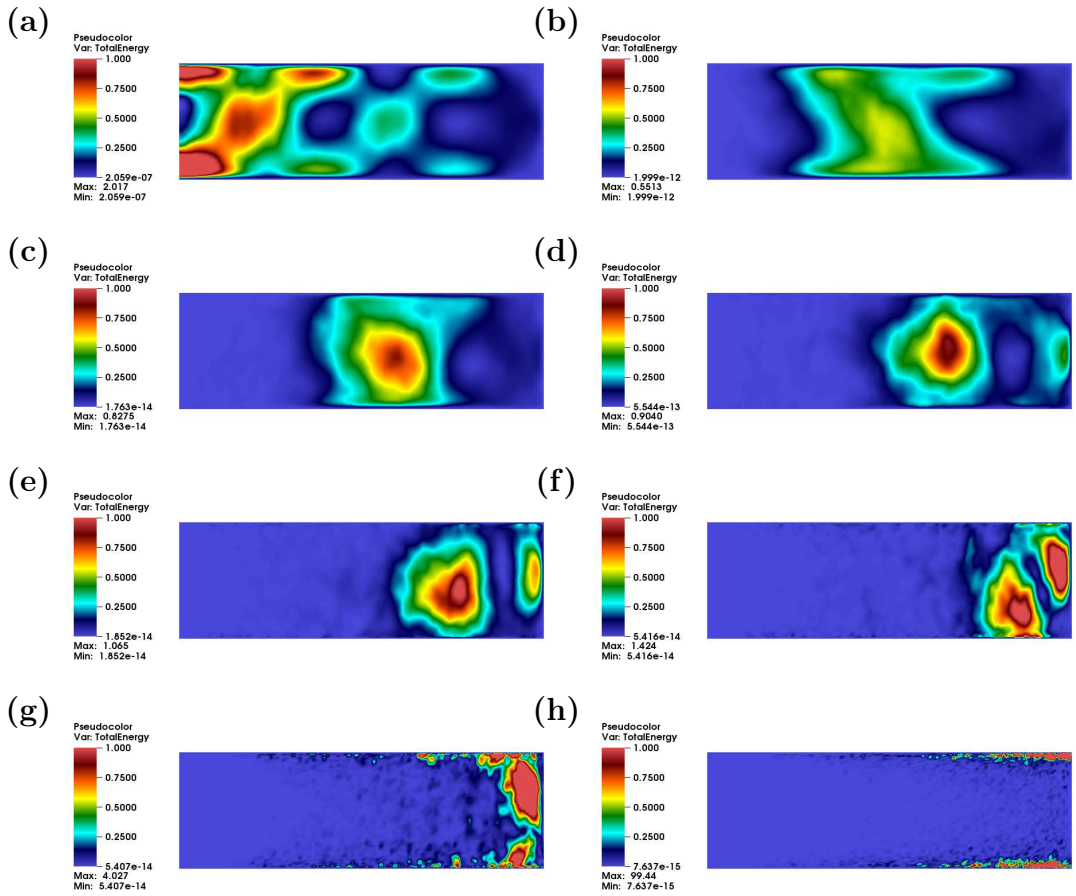


Figure 52: Total turbulent energy distribution in the r-z plane for the second POD mode for $k = 1$ (a), 3 (b), 6 (c), 10 (d), 20 (e), 40 (e), 100 (f) and 1000 (g).

Finally, it is worth pointing out that the concentrations of energy in the $k = 100$ and 1000 POD modes appear to only be associated with small scale structures near the wall i.e. plumes. The genesis of plumes in the boundary layer has been shown to be a log-normal distribution process [57, 53] that organizes into larger scales as the plumes rise and are swept up in the large circulatory currents. Therefore it is logical that the POD eigenspectra associated with the higher Fourier numbers is flat, because there is little organization at the length scales and spatial locations that these modes are associated with.

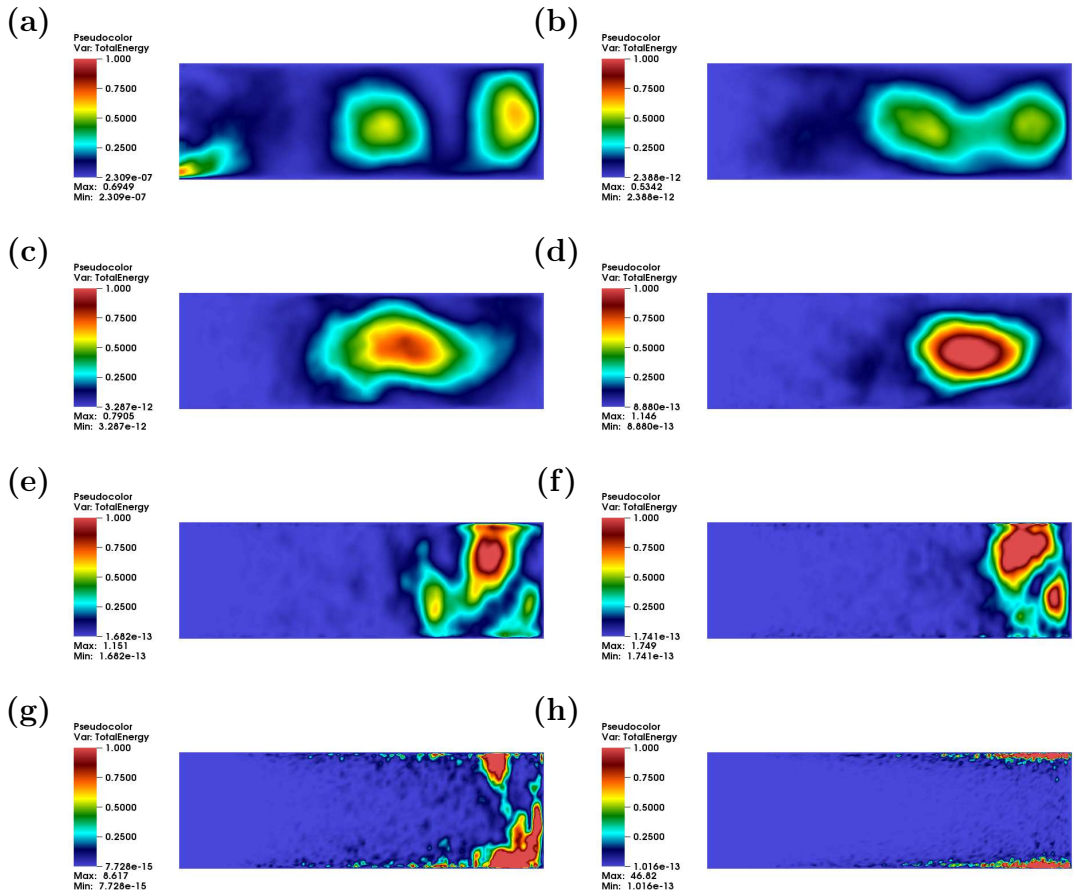


Figure 53: Total turbulent energy distribution in the r-z plane for the third POD mode for $k = 1$ (a), 3 (b), 6 (c), 10 (d), 20 (e), 40 (e), 100 (f) and 1000 (g).

6.5 Summary and Conclusions

This chapter is focused on using proper orthogonal decomposition (POD) to extract additional physics from the azimuthal Fourier modes of turbulent Rayleigh-Bénard convection in a cylinder. The specific items of interest are understanding the r-z distribution of the most energetic structures in each of the Fourier modes, and determining if additional insights can be found in the temporal evolution of the global structures through the lens of POD.

It has been found that the global shifts in phase and amplitude of the Fourier modes is fully captured by the first POD mode and does not subdivide into the additional higher-order POD modes. The behavior of the first POD mode is very similar to the area integrated Fourier coefficients that are described in the previous chapter. This shows that the structure responsible for the most energy over time for the dominant Fourier modes is well represented by the first POD mode. Using the projection of the $k = 2$, $m = 1$ mode onto the time series an estimate for the speed of the azimuthal drift is measured by fitting the phase change with a linear regression. The rotation rate has been determined to be approximately 1.1 degrees per eddy turnover. This is an order of magnitude faster than the long-time drift reported by Brown *et al.*, but an order of magnitude slower than the medium-time drift event reported in the same work [12].

It has also been shown that energy in the boundary layers is encapsulated in the first few POD modes. This shows that the majority of the energy in the flow field is contained in the boundary layers. The three-dimensional, spatial structure of the first few POD modes are dominated by large-scale roll-cells crossing the entire layer depth. This is evidence that the most energetic structures in the flow field are the large-scale roll-cells. Interestingly, the roll-cell associated with $k = 1$, $m = 1$ is the only $m = 1$ structure not connected to the side walls. It resembles the "wind of turbulence" phenomena seen in unit Γ convection cells since it has the same general dimensions. This indicates that there may be a natural length scale associated with this structure, and that it could be present at even higher values of Γ. However, it is a very weak structure in terms of energy percentage, and is not discernible without the modal decomposition.

While the low order POD modes show boundary layer dominated structures, the

higher order POD modes correspond to structures in the bulk of the flow field. The higher order POD modes also show a division of structures represented by energy concentrations in the bulk. This division process is consistent with the well known concept that energy cascades from large scales to small scales.

These results show that the physics of the system can be well represented by POD of Fourier modes, and that additional insights into the physical composition of the flow field can be drawn by this decomposition process. However, this connection to physics decays as the Fourier wave number increases. It has been shown that the POD eigenspectra for large Fourier wave numbers becomes increasingly flat, and that the individual POD modes structure becomes very sensitive to the snapshot selection process. It is believed that this is due to the fact that the length scales represented by these modes are in a range where there is little distinction between the physical structures with respect to the total integration domain of the inner product i.e. the genesis of plumes within the viscous boundary layer. The lack of distinguishable and/or significant physical structures makes limits the ability of POD to capture meaningful physics, and so POD is not a good tool to analyze structures on these length scales. A possible correction for studying these phenomena would be to sample over a smaller portion of the domain, such as the near wall region, so that these structures represent a larger portion of the energy with respect to the L2 norm.

Chapter 7

SUMMARY AND CONCLUSIONS

The main objective of this work is to examine the three-dimensional structure of turbulent Rayleigh-Bénard convection (RBC) in cylindrical domains at an aspect-ratio where the large-scale structures differs from the unit aspect-ratio case. The data for this work has been generated using direct numerical simulation with the code Nek5000 to simulate RBC at a Rayleigh number of $Ra = 9.6 \times 10^7$, Prandtl number of $Pr = 6.7$ and aspect-ratio of $\Gamma = 6.3$. The simulation was initialized at a lower Rayleigh number with small perturbations added to a conduction profile, and then this was gradually ramped up to the target Rayleigh number over 5 discrete intervals. The flow field was judged to be at steady state when the volume averaged kinetic energy stabilized. Good statistical agreement with experiments has been demonstrated, and the numerics have been shown to meet modern standards for resolution in turbulent RBC. Three high-level research questions have been used to focus the analysis of this study, and the answer to these questions that have been obtained in this body of work are provided in the sections below.

7.1 The Best Estimate for an Infinite Time Averaged Field for Rayleigh-Bénard Convection in Cylindrical Cells

It has been shown that the best estimate for the infinite time averaged field is one where all available symmetries in the flow field have been accounted for. In the case of RBC in cylinders this includes an azimuthal average and an additional symmetry

arguments that accounts for the vertical antisymmetry. It has also been acknowledged that an additional symmetry may exist that accounts for the net rotations observed in cylindrical convection cells.

This is at odds with the concept of a mean wind, or the "wind of turbulence" that is oft referenced in the Rayleigh-Bénard literature. This singular roll-cell does not obey the geometric symmetry of the system and extends across the entire diameter of the convection cell, and it changes its azimuthal orientation indicating a dynamic nature. The multi-roll cell structures observed throughout this work also show similar properties. While there is no denying the existence of and dominant nature of roll-cells in turbulent RBC, the fundamental properties of these roll-cells are such that they must reside in the fluctuating field.

The time scales over which these fluctuations evolve are very long and so it is often favorable to study the flow field in reference to the large scale circulations. On time scales where the roll-cells azimuthal orientation is constant the time-averaged field is actually a conditional average where the specified condition is the azimuthal orientation. This can also be extended to a phase-average over longer time-scales by aligning the orientation of the large scale structure.

7.2 Properties of the Large-Scale Coherent Structures in Rayleigh-Bénard Convection When the Domain is a Moderate Aspect-Ratio Cylinder

The large-scale structures in this study have proven to be roll-cells that are periodically aligned along the side walls. These roll-cells extend radially from the wall to the center of the cell and show that the domain is not yet wide enough to approximate the infinite aspect-ratio case. This conclusion is also supported by the fact

that the azimuthal energy spectra shows a radial dependence for energy distribution with respect to length-scales. The radial dependence is small, and only affects the very largest length scales of the flow that are directly related to the orginizatinonal frequency of the large-scale structures.

Over the evolution of $O(10^2)$ eddy-turnovers, or $O(10^3)$ free-fall times, a major shift in the large-scale structure takes place and the spatial organization goes from a three pronged hub-and-spoke pattern, where the spokes are a lines of vorticity and the hub is a central updraft, to a two roll-cell structure. This transition occurred over a time scale of 10 eddy-turnovers which is the same time-scale of reorientation events at lower aspect-ratios. This similarity suggests that 10 eddy-turnover time units is an important scale for dynamic events in turbulent RBC.

7.3 Physical Insights can be Obtained about the Large-Scale Structures through a Modal Representation of the Flow Field inside the Cylindrical Domain

It has been shown that the structure of large-scale structure is well described by a small selection of low-order azimuthal Fourier modes. A single, dominant Fourier mode represents the each of the two large-scale patterns that were observed in the time series. These dominant modes are the obvious peaks in the energy spectra and they have been used to quantify the transition time between the patterns by integrating the Fourier coefficients across the r-z plane. This shows that there is a physical connection between the large-scale structures and the Fourier modes in this study. The azimuthal Fourier modes contain a radial dependency in the length scales that they describe and from this it can be concluded large-scale structures are modal in nature. In other

words the organizational large-scale structures are not associated with a single length scale, but rather a pattern, or mode.

The Fourier modes associated with the non-dominant modes do not show the same type of global temporal behavior, but rather show a more random distribution of energy and amplitude. This is not surprising since the large-scale structures are described so well by the dominant Fourier modes, and it shows that the smaller scale structures are more dependent on length scales. The Fourier modes are further evaluated with proper orthogonal decomposition (POD) to investigate the structures based on their r-z dependence. POD reveals that the first mode for each Fourier number has a concentration of energy in the boundary layers which indicates that the most energetic structures with respect to total turbulent energy are found in the boundary layers of the flow field. These structures manifest themselves as roll-cells that span the entire layer depth. Additional distinct bands of smaller scale structures can be seen in the first POD modes, and as the Fourier number increases these bands shift in radial position toward the wall. This is a manifestation of distinct length scales for the small-scale structures that is not easily identifiable by just looking at the Fourier modes.

Higher order POD modes show bulk dominant structures which are not as easily examined via visual inspection of streamlines. However, the bifurcation of concentrations of total turbulent energy in the modes indicates that the structures in the bulk experience an energy cascade from large to small structures.

7.4 Concluding Remarks and Future Work

In conclusion, this work has explored turbulent RBC in a manner that is only available today through numerical simulation. By decomposing the flow field with modal techniques rich insight into the underlying structure of the fluid dynamics has been gained. The code and that has been used to perform this analysis is documented in Appendix B and at https://github.com/psakievich/DissertationCode to ensure that others can perform similar analysis in the future. Particular studies of interest would be repeating these analysis over a large range of Γ, Rayleigh numbers and Prandtl numbers.

There are still a multitude of opportunities for analysis and additional physical insights with this dataset alone. For example, only a small section of the POD modes were analyzed as part of this document. In total there are several thousand POD modes available, and a dictionary of modes has been provided in Appendix B for future study. This dictionary only represents is still a subset of the total modes available, but its a more comprehensive resource than what could reasonably be included in the main chapters of this document. In general, there is still much to learn from the POD. Since POD on the Fourier wave numbers that are not directly related to the large-scale structures show better resolution, and a strong dependence on physical length scales, the POD results could possibly be made more conceptually tractable by filtering out the large-scale structures in Fourier space and then performing a 3D method of snapshots on the remaining field. A similar study could also involve filtering out Fourier modes based on their local radius so that resulting field only contains a narrow band of physical lengths scales. This would facilitate a detailed analysis of the

small scale structures within the flow field and provide a complete description of the turbulent structures' life-cycle within the convection cell.

Additional future studies that are more applied to engineering applications could also include the introduction of pillars in the flow field, or modifications to the local temperature field at the boundaries to attempt to enhance heat transport.

7.5 Extended Impact

The results of this work have the potential to impact other research fields beyond RBC. In the first chapter of this document an outline is provided that shows how the hierarchical organization of coherent structures in RBC is similar to the structures observed in shear flows. The identification of very-large-scale motions (VSLM's) or superstructures [45, 8, 38, 9] in turbulent boundary layers, channel flow and pipe flow, has received increased interest in the shear flow community. These parallels suggest an important crossover between the two fields of study, and that additional insights into the nature of turbulence can be found by directing attention to these similarities. For example, the similarity between the structures in this study and the VLSM's measured in pipe flow [36] has been acknowledged by authors from both studies in personal conversation. This is of interest because the mechanism by which production of turbulent energy is created differs substantially between the two flows and yet the organization of the turbulent structures are very similar.

This work also has the potential to impact specific engineering applications. Two examples where results from this work could be applied are cooling of electronics and HVAC designs that optimize circulation currents in wide enclosures. Modern design trends such as thin electronic devices and open concept floor plans must

increasingly rely on buoyancy driven convection to achieve efficient, cost effective thermal management. The demand for energy efficiency and compactness increases the demand for well engineered thermal convection systems. Understanding the details of the physical mechanisms within turbulent thermal convection through studies like this becomes increasingly important as the engineering process becomes more refined. For example, the modal framework used in this work (Fourier and POD) provide insight that can be useful for heat transfer optimization and control. Understanding the modal structure of the flow field can help designers modify the geometry and/or boundary conditions to enhance the modes that transport the most heat. Additionally, low-order models can be derived from the modal representation of the flow field to reduce the cost of running design iterations [50].

REFERENCES

[1] R. J. Adrian, K.T. Christensen, and Z.C. Liu. "Analysis and interpretation of instantaneous turbulent velocity fields". In: *Experiments in Fluids* 29.3 (2000), pp. 275–290.

[2] R. J. Adrian, R. T. D. S. Ferreira, and T. Boberg. "Turbulent thermal convection in wide horizontal fluid layers". In: *Experiments in Fluids* 4 (1986), pp. 121–141.

[3] R. J. Adrian, C. D. Meinhart, and C. D. Tomkins. "Vortex organization in the outer region of the turbulent boundary layer". In: *Journal of Fluid Mechanics* 422 (Nov. 2000), pp. 1–54. ISSN: 1469-7645. DOI: 10.1017/S0022112000001580.

[4] Ronald J Adrian. "Hairpin vortex organization in wall turbulencea)". In: *Physics of Fluids (1994-present)* 19.4 (2007), p. 041301.

[5] Guenter Ahlers, Siegfried Grossmann, and Detlef Lohse. "Heat transfer and large scale dynamics in turbulent Rayleigh-Benard convection". In: *Reviews of Modern Physics* 81.2 (2009), pp. 503–537.

[6] C Altantzis et al. "Detailed numerical simulations of intrinsically unstable two-dimensional planar lean premixed hydrogen/air flames". In: *Proceedings of the Combustion Institute* 33.1 (2011), pp. 1261–1268.

[7] J. Bailon-Cuba, M. S. Emran, and J. Schumacher. "Aspect ratio dependence of heat transfer and large-scale flow in turbulent convection". In: *Journal of Fluid Mechanics* 655 (July 2010), pp. 152–173. ISSN: 1469-7645. DOI: 10.1017/S0022112010000820.

[8] BJ Balakumar and RJ Adrian. "Large-and very-large-scale motions in channel and boundary-layer flows". In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 365.1852 (2007), pp. 665–681.

[9] J Baltzer, R Adrian, and X Wu. "Turbulent boundary layer structure identification via POD". In: *Proceedings of the summer program*. 2010, p. 55.

[10] Brandt A Belson, Jonathan H Tu, and Clarence W Rowley. "Algorithm 945: modred—a parallelized model reduction library". In: *ACM Transactions on Mathematical Software (TOMS)* 40.4 (2014), p. 30.

[11] Eberhard Bodenschatz, Werner Pesch, and Guenter Ahlers. "Recent developments in Rayleigh-Bénard convection". In: *Annual review of fluid mechanics* 32.1 (2000), pp. 709–778.

[12] E. Brown, A. Nikolaenko, and G. Ahlers. "Reorientation of the Large-Scale Circulations in Turbulent Rayleigh-Benard Convection". In: *Physical Review Letters* 95.084503 (2005).

[13] D. Carati, A. Wray, and W. Cabot. *Ensemble averaged dynamic modeling*. Center for Turbulent Research, Proceedings of the Summer Program. 1996, pp. 237–248.

[14] Laltu Chandra and Günther Grötzbach. "Analysis and modelling of the turbulent diffusion of turbulent heat fluxes in natural convection". In: *International Journal of Heat and Fluid Flow* 29.3 (2008), pp. 743–751.

[15] Hank Childs et al. "Extreme scaling of production visualization software on diverse architectures". In: *IEEE Computer Graphics and Applications* 3 (2010), pp. 22–31.

[16] Emily SC Ching. *Statistics and scaling in turbulent Rayleigh-Benard convection*. Springer, 2014.

[17] G. N. Coleman, J. Kim, and P. R. Spalart. "A numerical study of strained three-dimensional wallbounded turbulence". In: *Journal of Fluid Mechanics* 417 (2000), p. 025112.

[18] James W Deardorff. "Convective velocity and temperature scales for the unstable planetary boundary layer and for Rayleigh convection". In: *Journal of the Atmospheric Sciences* 27.8 (1970), pp. 1211–1213.

[19] Michel O Deville, Paul F Fischer, and Ernest H Mund. *High-order methods for incompressible fluid flow*. Vol. 9. Cambridge University Press, 2002.

[20] Andrew Duggleby et al. "Dynamical Eigenfunction Decomposition of Turbulent Pipe Flow". In: *Journal of Turbulence* 8.43 (2007), pp. 1–24.

[21] Mohammad S Emran and Jörg Schumacher. "Large-scale mean patterns in turbulent convection". In: *Journal of Fluid Mechanics* 776 (2015), pp. 96–108.

[22] R. L. Fernandes. "The spatial structure of turbulent Rayleigh-Benard convection". PhD thesis. Urbana, IL: University of Illinois, 2001.

[23] R. L. Fernandes and R. J. Adrian. "Scaling of velocity and temperature fluctuations in turbulent thermal convection". In: *Exp. Thermal Fluid Science* 26 (2002), pp. 355–360.

[24] Paul F Fischer, James W Lottes, and Stefan G Kerkemeier. "nek5000 Web page". In: *Web page: http://nek5000. mcs. anl. gov* (2008).

[25] Paul Fischer et al. "Petascale algorithms for reactor hydrodynamics". In: *Journal of Physics: Conference Series*. Vol. 125. 1. IOP Publishing. 2008, p. 012076.

[26] Matteo Frigo. "A fast Fourier transform compiler". In: *Acm sigplan notices*. Vol. 34. 5. ACM. 1999, pp. 169–180.

[27] Matteo Frigo and Steven G Johnson. "FFTW: An adaptive software architecture for the FFT". In: *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 3. IEEE. 1998, pp. 1381–1384.

[28] Denis Funfschilling et al. "Heat transport by turbulent Rayleigh–Bénard convection in cylindrical samples with aspect ratio one and larger". In: *Journal of Fluid Mechanics* 536 (Aug. 2005), pp. 145–154. ISSN: 1469-7645. DOI: 10.1017/S0022112005005057.

[29] S. Grossmann and D. Lohse. "Fluctuations in turbulent Rayleigh-Benard convection: The role of plumes". In: *Physics of Fluids* 16 (2004), pp. 4462–4472.

[30] S. Grossmann and D. Lohse. "Prandtl and Rayleigh number dependece of the Reynolds number in turbulent thermal convection". In: *Physics Review E* 66.016305 (2002).

[31] S. Grossmann and D. Lohse. "Thermal convection for large Prandtl numbers". In: *Physical Review Letters* 86 (2001), pp. 3316–3319.

[32] Siegfried Grossmann and Detlef Lohse. "Scaling in thermal convection: a unifying theory". In: *Journal of Fluid Mechanics* 407 (2000), pp. 27–56.

[33] Günther Grötzbach. "Spatial resolution requirements for direct numerical simulation of the Rayleigh-Bénard convection". In: *Journal of Computational Physics* 49.2 (1983), pp. 241–264.

[34] M Guala, SE Hommema, and RJ Adrian. "Large-scale and very-large-scale motions in turbulent pipe flow". In: *Journal of Fluid Mechanics* 554 (2006), pp. 521–542.

[35] Xiaozhou He et al. “Transition to the Ultimate State of Turbulent Rayleigh-Benard Convection”. In: *Physical Review Letters* 108.024502 (Jan. 2012).

[36] Leo HO Hellström, Aman Sinha, and Alexander J Smits. “Visualizing the very-large-scale motions in turbulent pipe flow”. In: *Physics of Fluids* 23.1 (2011), p. 011703.

[37] Philip Holmes, John L Lumley, and Gal Berkooz. *Turbulence, coherent structures, dynamical systems and symmetry*. 2nd ed. Cambridge university press, 2012.

[38] N Hutchins and Ivan Marusic. “Evidence of very long meandering features in the logarithmic region of turbulent boundary layers”. In: *Journal of Fluid Mechanics* 579 (2007), pp. 1–28.

[39] E. Jeyapal, G. N. Coleman, and C. L. Rumsey. *Assessment of Higher-order RANS Closures in a Decelerated Planar Wall-bounded Turbulent Flow*. AIAA Paper 2014-2088. 44th AIAA Fluid Dynamics Conference, Atlanta, GA. 2014.

[40] S Kenjereš and K Hanjalić. “LES, T-RANS and hybrid simulations of thermal convection at high Ra numbers”. In: *International journal of heat and fluid flow* 27.5 (2006), pp. 800–810.

[41] S Kenjereš and K Hanjalić. “Transient analysis of Rayleigh–Bénard convection with a RANS model”. In: *International journal of heat and fluid flow* 20.3 (1999), pp. 329–340.

[42] S. Kerkemeier and S. Parker. *PFF, 2010. Scalability of the NEK5000 spectral element code. Jülich Blue Gene/P Extreme Scaling Workshop 2010*. Tech. rep. Technical Report, 2010.

[43] Stefan Georg Kerkemeier. “Direct numerical simulation of combustion on petascale platforms: application to turbulent non-premixed hydrogen autoignition”. PhD thesis. ETH, 2010.

[44] Robert M Kerr. “Rayleigh number scaling in numerical convection”. In: *Journal of Fluid Mechanics* 310 (1996), pp. 139–179.

[45] KC Kim and RJ Adrian. “Very large-scale motion in the outer layer”. In: *Physics of Fluids* 11.2 (1999), pp. 417–422.

[46] R. H. Kraichnan. “Turblent thermal convection at arbitrary Prandtl number”. In: *Physics of Fluids* 5 (1962), pp. 1374–1389.

[47] Z. Liu, R.J. Adrian, and T. J. Hanratty. "Large-scale modes of turbulent channel flow: transport and structure". In: *Journal of Fluid Mechanics* 448 (2001), pp. 53–80.

[48] Brandon E Merrill et al. "A spectrally accurate method for overlapping grid solution of incompressible Navier–Stokes equations". In: *Journal of Computational Physics* 307 (2016), pp. 60–93.

[49] Brandon Merrill and Yulia Peet. "High-Order Moving Overlapping Grid Methodology for Aerospace Applications". In: *53rd AIAA Aerospace Sciences Meeting*. 2015, p. 1743.

[50] Elia Merzari, W David Pointer, and Paul Fischer. "A POD-based solver for the advection-diffusion equation". In: *ASME-JSME-KSME 2011 Joint Fluids Engineering Conference*. American Society of Mechanical Engineers. 2011, pp. 1139–1147.

[51] Pankaj Kumar Mishra et al. "Dynamics of reorientations and reversals of large-scale flow in Rayleigh–Bénard convection". In: *Journal of Fluid Mechanics* 668 (2011), pp. 480–499.

[52] Ronald L Panton. *Incompressible flow*. Third. John Wiley &; Sons, 2006.

[53] Antonio Parodi et al. "Clustering of plumes in turbulent convection". In: *Physical review letters* 92.19 (2004), p. 194503.

[54] YT Peet and PF Fischer. "Legendre spectral element method with nearly incompressible materials". In: *European Journal of Mechanics-A/Solids* 44 (2014), pp. 91–103.

[55] K. Petschel et al. "Kinetic energy transport in Rayleigh-Bénard convection". In: *Journal of Fluid Mechanics* 773 (2015), pp. 395–417.

[56] Ronald du Puits, Christian Resagk, and André Thess. "Breakdown of wind in turbulent thermal convection". In: *Physical Review E* 75.1 (2007), p. 016302.

[57] B. A. Puthenveettil and J. Arakeri. "Plume strucutre in high Rayleigh-number convection". In: *Journal of Fluid Mechanics* 542 (2005), pp. 217–249.

[58] X-L Qiu and P Tong. "Large-scale velocity structures in turbulent thermal convection". In: *Physical Review E* 64.3 (2001), p. 036304.

[59] P. J. Sakievich, Y. T. Peet, and R. J. Adrian. "Large scale coherent structures in wide-aspect-ratio, turbulent, Rayleigh-Benard convection". In: *Int Symp on Turbulence and Shear Flow Phenomena* 9 (2015).

[60] PJ Sakievich, YT Peet, and RJ Adrian. "Large-scale thermal motions of turbulent Rayleigh–Bénard convection in a wide aspect-ratio cylindrical domain". In: *International Journal of Heat and Fluid Flow* 61 (2016), pp. 183–196.

[61] J. D. Scheel, E. Kim, and K. R. White. "Thermal and viscous boundary layers in turbulent Rayleigh-Benard convection". In: *Journal of Fluid Mechanics* 711 (2012), pp. 281–305.

[62] Janet D Scheel, Mohammad S Emran, and Jörg Schumacher. "Resolving the fine-scale structure in turbulent Rayleigh–Bénard convection". In: *New Journal of Physics* 15.11 (2013), p. 113063.

[63] William J Schroeder and Kenneth M Martin. "The Visualization Toolkit-30". In: (1996).

[64] Jörg Schumacher et al. "Small-scale universality in fluid turbulence". In: *Proceedings of the National Academy of Sciences* 111.30 (2014), pp. 10961–10965.

[65] M. J. Shelly and M. Vinson. "Coherent structures on a boundary layer in Rayleigh-Benard turbulence". In: *Nonlinearity* 5 (1992), pp. 323–351.

[66] O. Shishkina and C. Wagner. "Analysis of sheet-like thermal plumes in turbulent Rayleigh-Benard convection". In: *Journal of Fluid Mechanics* 599 (2007), pp. 383–404.

[67] Olga Shishkina and André Thess. "Mean temperature profiles in turbulent Rayleigh–Bénard convection of water". In: *Journal of Fluid Mechanics* 633 (2009), pp. 449–460.

[68] L Sirovich and H Park. "Turbulent thermal convection in a finite domain: Part I. Theory". In: *Physics of Fluids A: Fluid Dynamics* 2.9 (1990), pp. 1649–1658.

[69] Lawrence Sirovich. "Turbulence and the dynamics of coherent structures. I. Coherent structures". In: *Quarterly of applied mathematics* 45.3 (1987), pp. 561–571.

[70] Troy R. Smith, Jeff Moehlis, and Philip Holmes. "Low-Dimensional Modelling of Turbulence Using Proper Orthogonal Decomposition: A Tutorial". In: *Nonlinear Dynamics* 41 (2005), pp. 275–307.

[71] Richard JAM Stevens, Roberto Verzicco, and Detlef Lohse. "Radial boundary layer structure and Nusselt number in Rayleigh–Bénard convection". In: *Journal of Fluid Mechanics* 643 (2010), pp. 495–507.

[72] DR Wilson, TJ Craft, and H Iacovides. "Application of RANS turbulence closure models to flows subjected to electromagnetic and buoyancy forces". In: *International Journal of Heat and Fluid Flow* 49 (2014), pp. 80–90.

[73] Xiaohua Wu and Parviz Moin. "Direct numerical simulation of turbulence in a nominally zero-pressure-gradient flat-plate boundary layer". In: *Journal of Fluid Mechanics* 630 (July 2009), pp. 5–41. ISSN: 1469-7645. DOI: 10.1017/S0022112009006624.

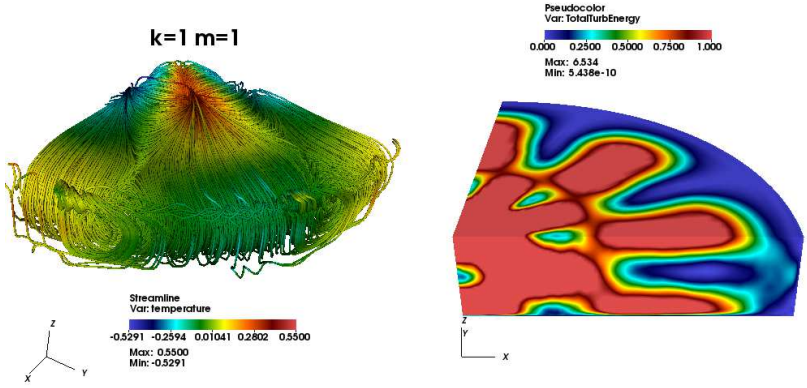
[74] Ke-Qing Xia, Chao Sun, and Yin-Har Cheung. "Large scale velocity structures in turbulent thermal convection with widely varying aspect ratio". In: *Proc. 14th Int. Symp. on Applications of Laser Techniques to Fluid Mechanics*. 2008.

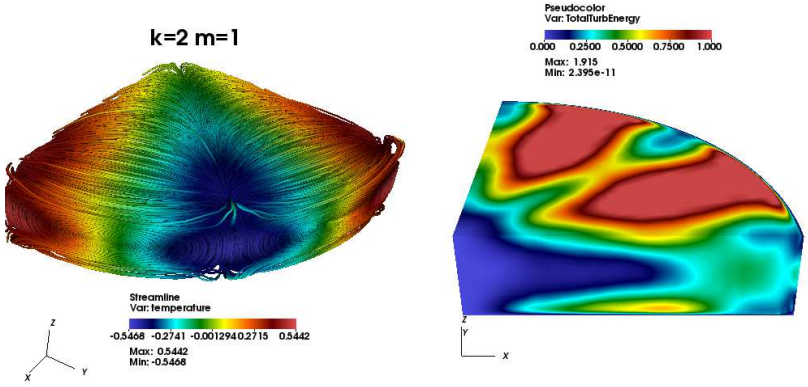
[75] Claudia Zimmermann and Rodion Groll. "Computational investigation of thermal boundary layers in a turbulent Rayleigh–Bénard problem". In: *International Journal of Heat and Fluid Flow* 54 (2015), pp. 276–291.

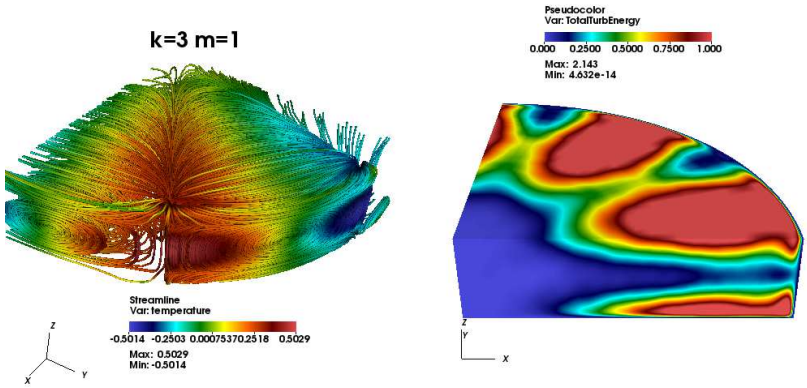
[76] G. Zocchi, E. Moses, and A. Libchaber. "Coherent structures in turbulent convection, an experimental study". In: *Physica A* 166 (1990), pp. 387–407.

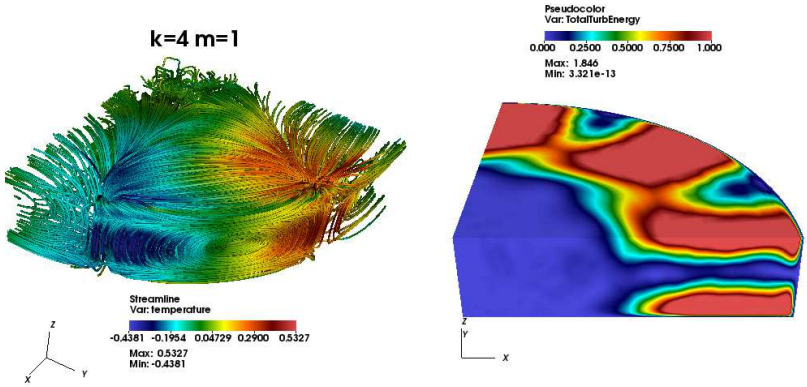
APPENDIX A

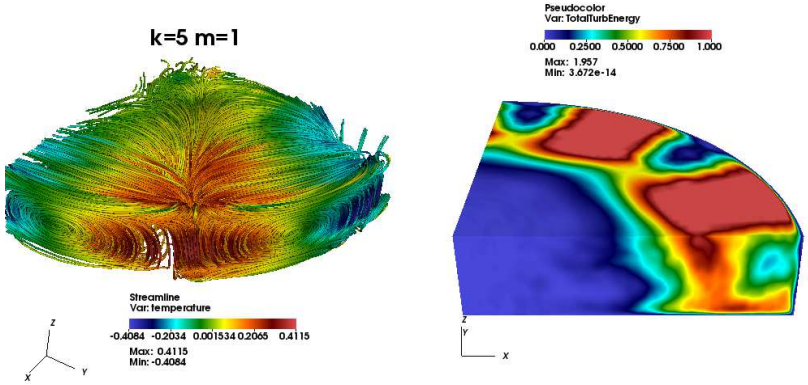
POD MODE DOCUMETATION

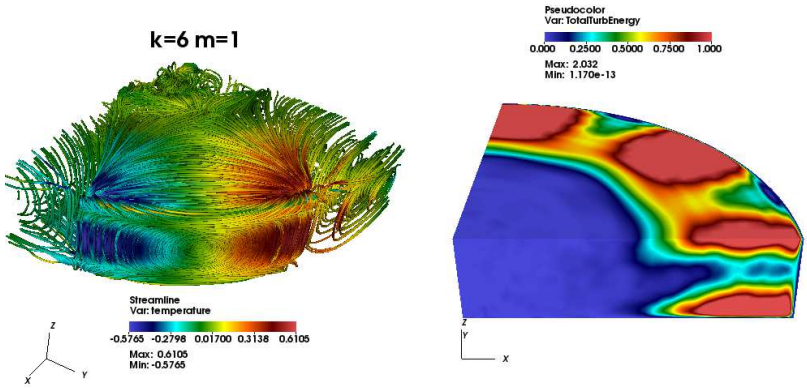

k=1 m=1
Streamline
Var: temperature
-0.5291 -0.2594 0.01041 0.2802 0.5500
Max: 0.5500
Min: -0.5291
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 6.534
Min: 5.438e-10

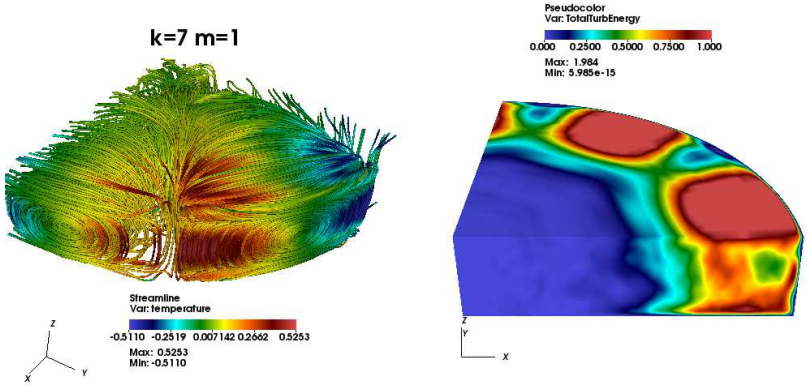

k=2 m=1
Streamline
Var: temperature
-0.5468 -0.2741 -0.0012940.2715 0.5442
Max: 0.5442
Min: -0.5468
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.915
Min: 2.395e-11

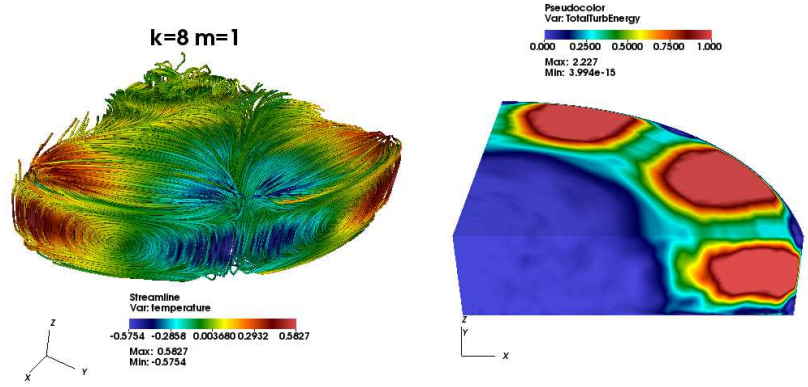

k=3 m=1
Streamline
Var: temperature
-0.5014 -0.2503 0.00075370.2518 0.5029
Max: 0.5029
Min: -0.5014
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.143
Min: 4.632e-14

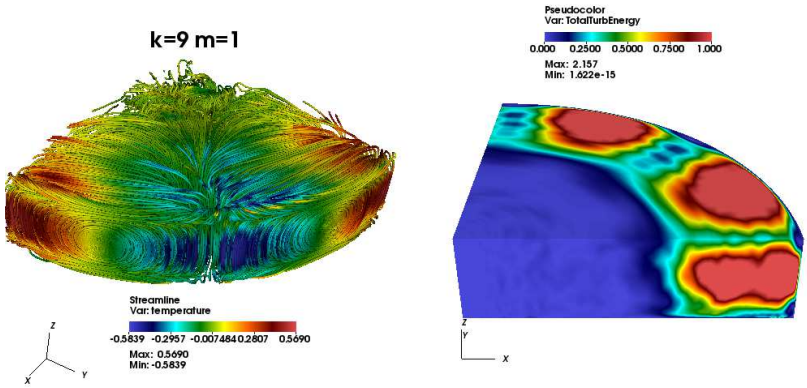

k=4 m=1
Streamline
Var: temperature
-0.4381 -0.1954 0.04729 0.2900 0.5327
Max: 0.5327
Min: -0.4381
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.846
Min: 3.321e-13

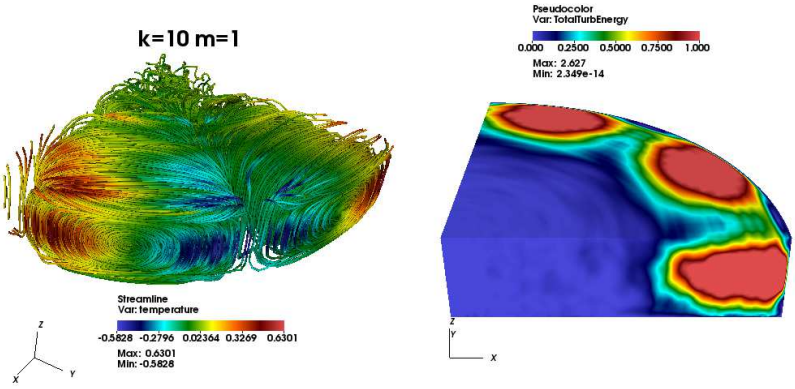

k=5 m=1
Streamline
Var: temperature
-0.4084 -0.2034 0.001534 0.2065 0.4115
Max: 0.4115
Min: -0.4084
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.957
Min: 3.672e-14

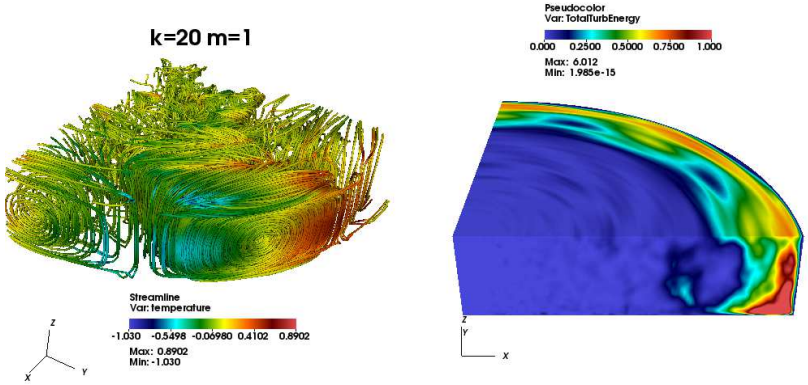

k=6 m=1
Streamline
Var: temperature
-0.5765 -0.2798 0.01700 0.3138 0.6105
Max: 0.6105
Min: -0.5765
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.032
Min: 1.170e-13

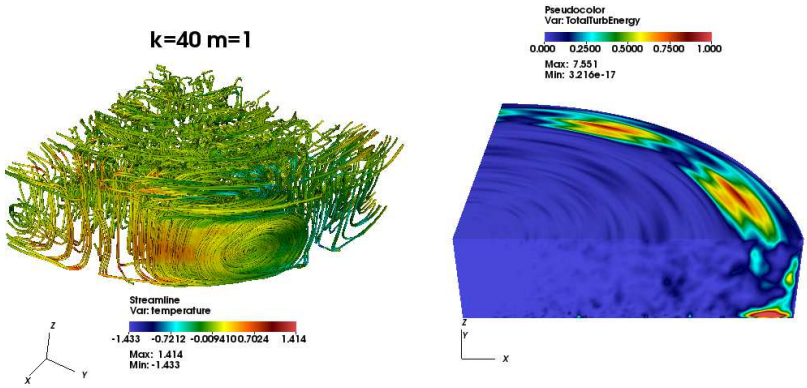

k=7 m=1
Streamline
Var: temperature
-0.5110 -0.2519 0.007142 0.2662 0.5253
Max: 0.5253
Min: -0.5110
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.984
Min: 5.985e-15

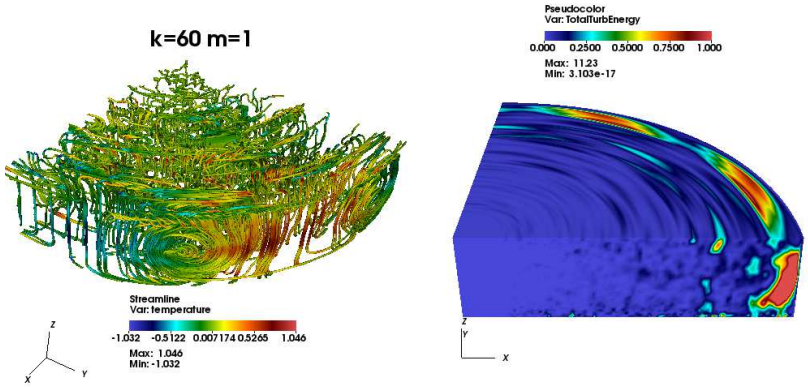

k=8 m=1
Streamline
Var: temperature
-0.5754 -0.2858 0.003680 0.2932 0.5827
Max: 0.5827
Min: -0.5754
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.227
Min: 3.994e-15

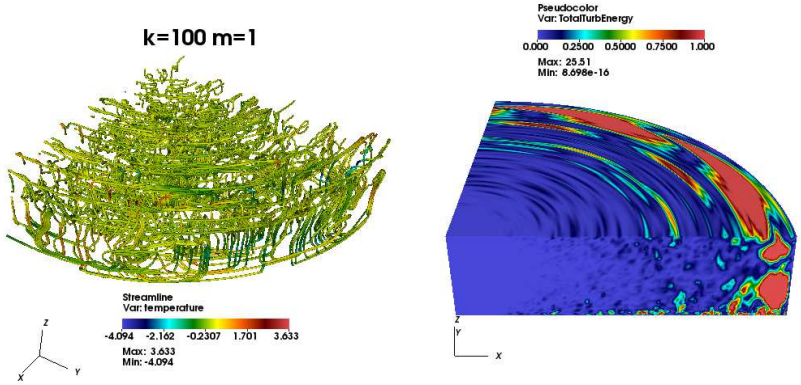

k=9 m=1
Streamline
Var: temperature
-0.5839 -0.2957 -0.007484 0.2807 0.5690
Max: 0.5690
Min: -0.5839
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.157
Min: 1.622e-15

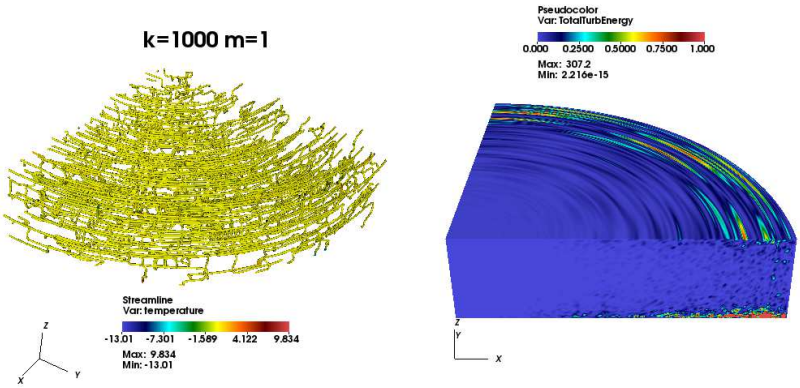

k=10 m=1
Streamline
Var: temperature
-0.5828 -0.2796 0.02364 0.3269 0.6301
Max: 0.6301
Min: -0.5828
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.627
Min: 2.349e-14

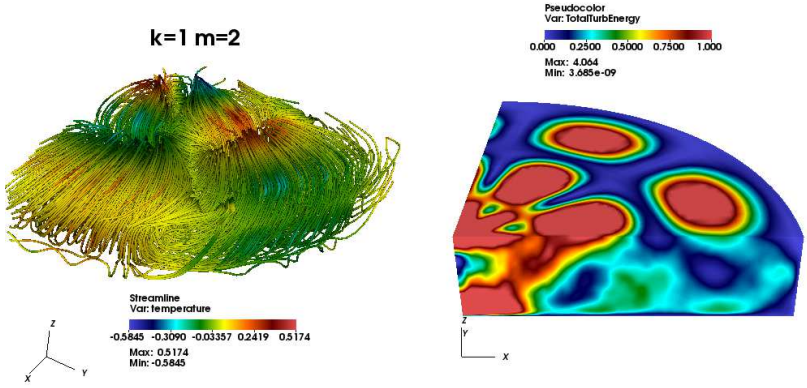

k=20 m=1
Streamline
Var: temperature
-1.030 -0.5498 -0.06980 0.4102 0.8902
Max: 0.8902
Min: -1.030
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 6.012
Min: 1.985e-15

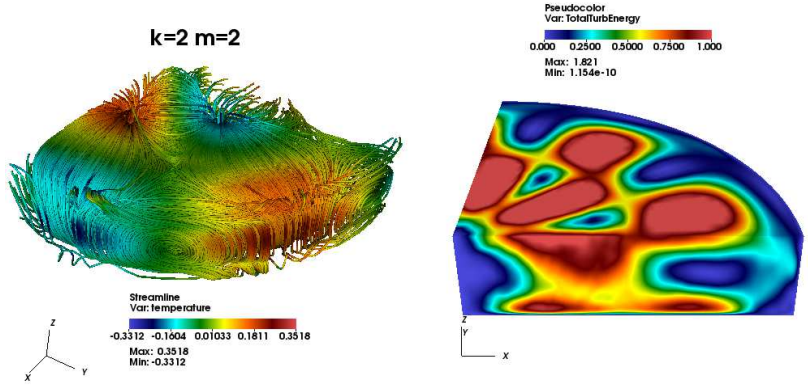

k=40 m=1
Streamline
Var: temperature
-1.433 -0.7212 -0.009410 0.7024 1.414
Max: 1.414
Min: -1.433
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 7.551
Min: 3.216e-17

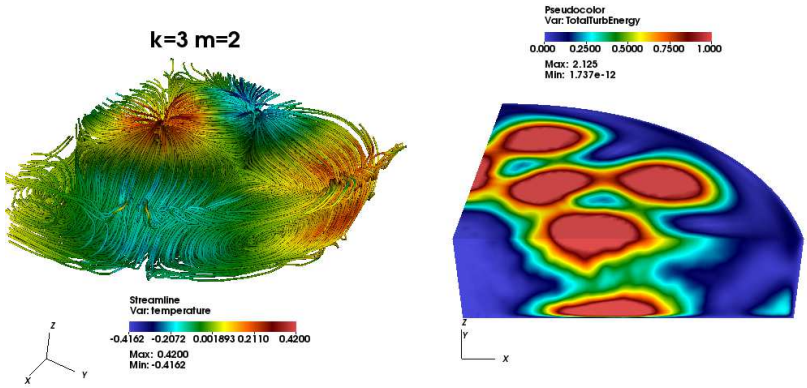

k=60 m=1
Streamline
Var: temperature
-1.032 -0.5122 0.007174 0.5265 1.046
Max: 1.046
Min: -1.032
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 11.23
Min: 3.103e-17

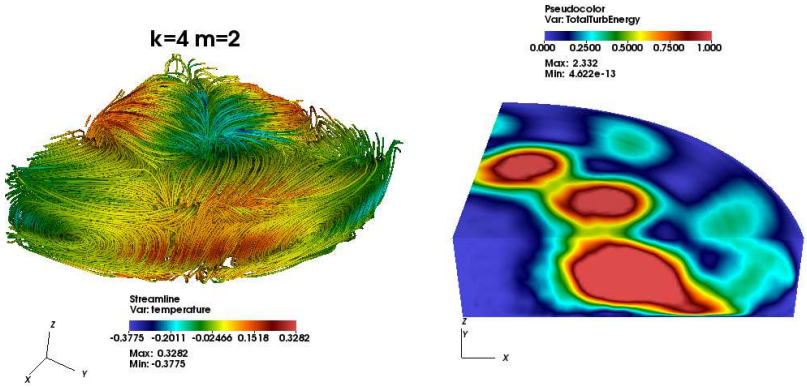

k=100 m=1
Streamline
Var: temperature
-4.094 -2.162 -0.2307 1.701 3.633
Max: 3.633
Min: -4.094
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 25.51
Min: 8.698e-16

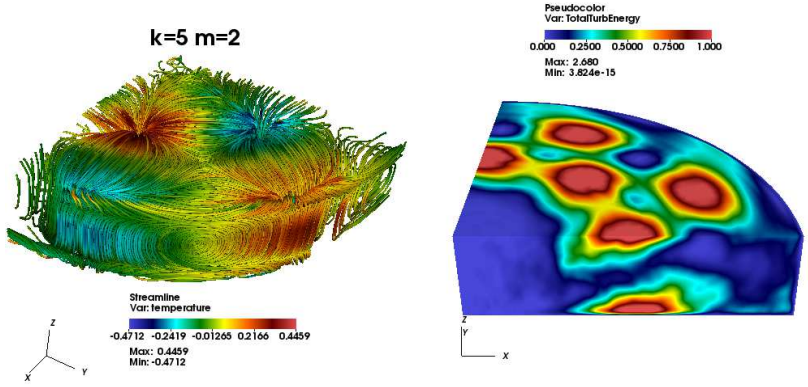

k=1000 m=1
Streamline
Var: temperature
-13.01 -7.301 -1.589 4.122 9.834
Max: 9.834
Min: -13.01
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 307.2
Min: 2.216e-15

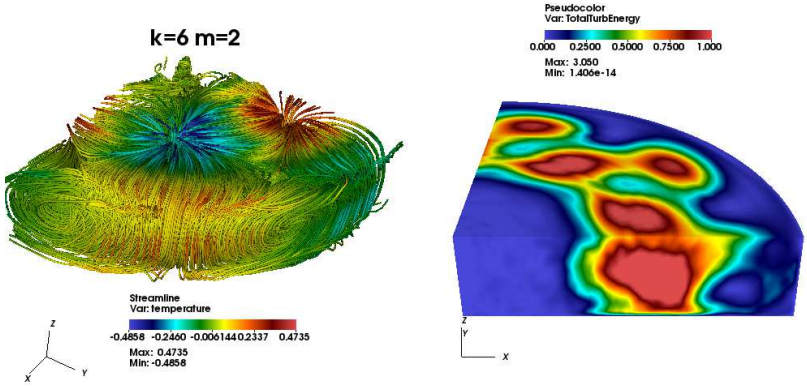

k=1 m=2
Streamline
Var: temperature
-0.5845 -0.3090 -0.03357 0.2419 0.5174
Max: 0.5174
Min: -0.5845
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.064
Min: 3.685e-09

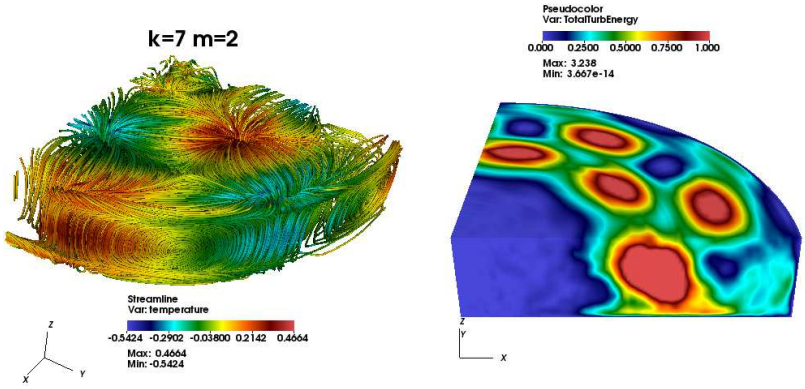

k=2 m=2
Streamline
Var: temperature
-0.3312 -0.1604 0.01033 0.1811 0.3518
Max: 0.3518
Min: -0.3312
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.821
Min: 1.154e-10

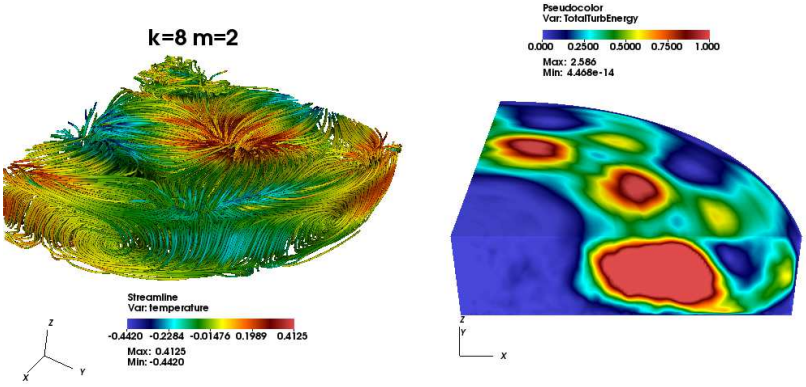

k=3 m=2
Streamline
Var: temperature
-0.4162 -0.2072 0.001893 0.2110 0.4200
Max: 0.4200
Min: -0.4162
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.125
Min: 1.737e-12

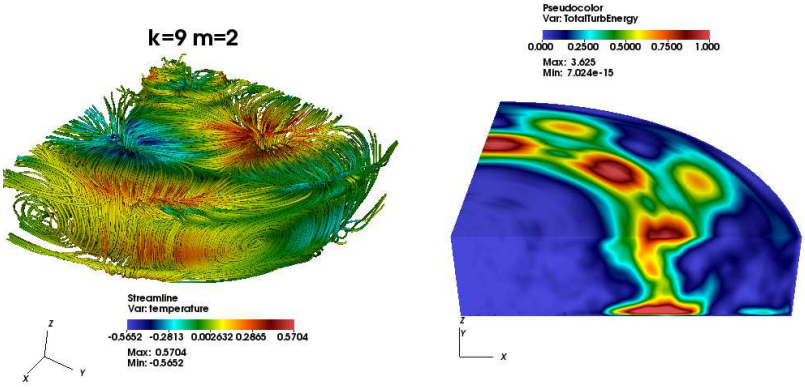

k=4 m=2
Streamline
Var: temperature
-0.3775 -0.2011 -0.02466 0.1518 0.3282
Max: 0.3282
Min: -0.3775
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.332
Min: 4.622e-13

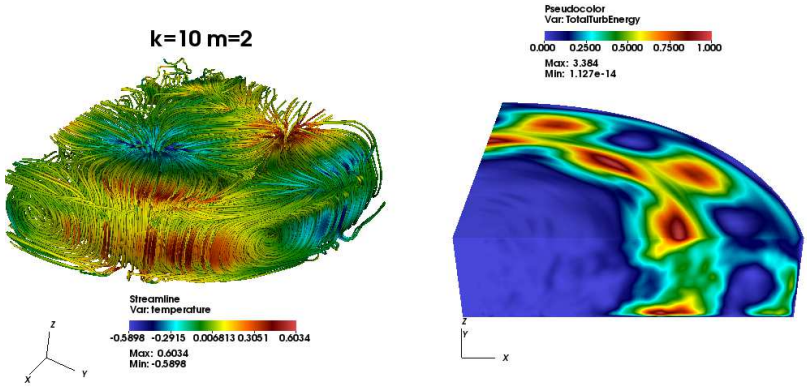

k=5 m=2
Streamline
Var: temperature
-0.4712 -0.2419 -0.01265 0.2166 0.4459
Max: 0.4459
Min: -0.4712
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.680
Min: 3.824e-15

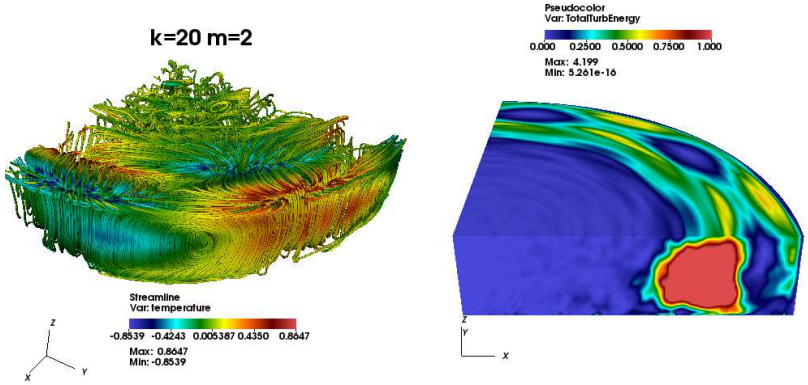

k=6 m=2
Streamline
Var: temperature
-0.4858 -0.2460 -0.006144 0.2337 0.4735
Max: 0.4735
Min: -0.4858
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.050
Min: 1.406e-14

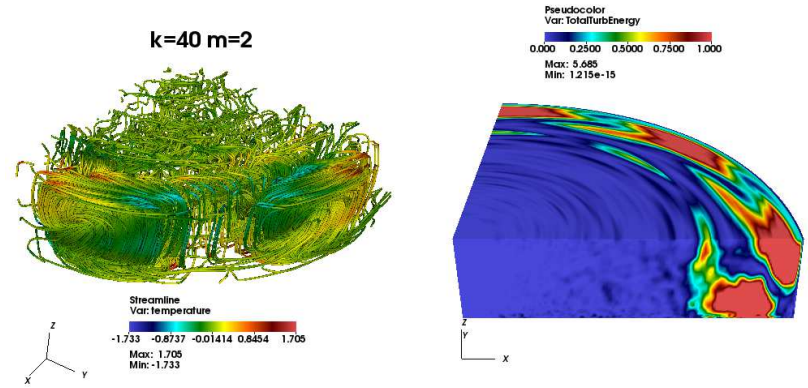

k=7 m=2
Streamline
Var: temperature
-0.5424 -0.2902 -0.03800 0.2142 0.4664
Max: 0.4664
Min: -0.5424
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.238
Min: 3.667e-14

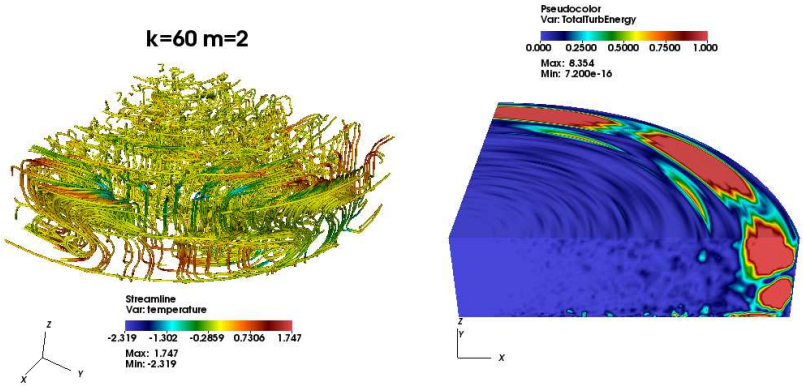

k=8 m=2
Streamline
Var: temperature
-0.4420 -0.2284 -0.01476 0.1989 0.4125
Max: 0.4125
Min: -0.4420
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.586
Min: 4.468e-14

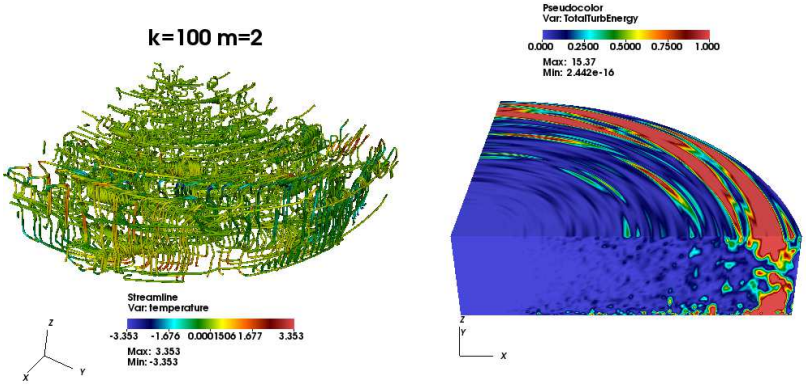

k=9 m=2
Streamline
Var: temperature
-0.5652 -0.2813 0.002632 0.2865 0.5704
Max: 0.5704
Min: -0.5652
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.625
Min: 7.024e-15

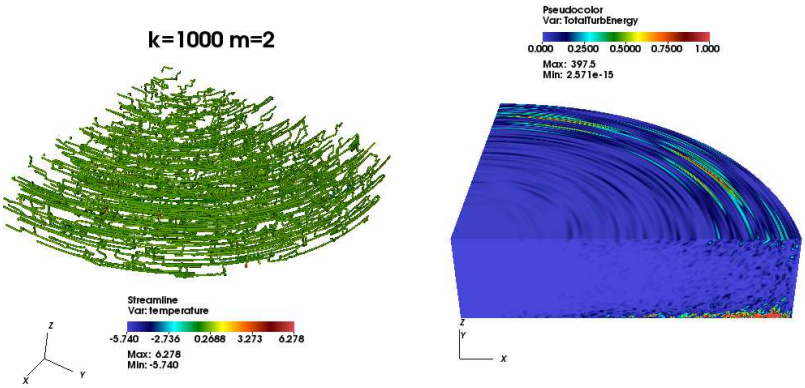

k=10 m=2
Streamline
Var: temperature
-0.5898 -0.2915 0.006813 0.3051 0.6034
Max: 0.6034
Min: -0.5898
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.384
Min: 1.127e-14

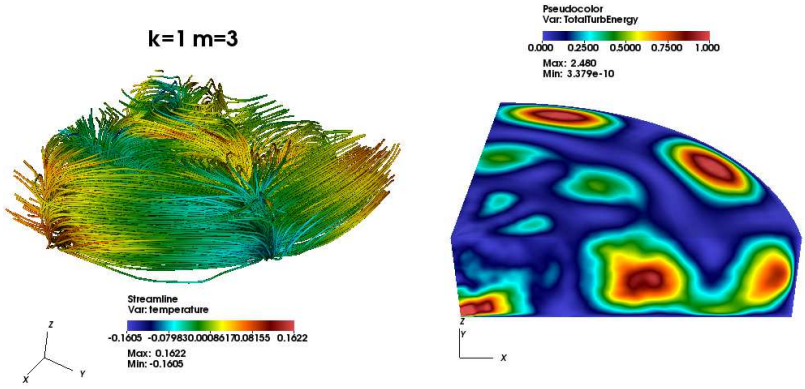

k=20 m=2
Streamline
Var: temperature
-0.8539 -0.4243 0.005387 0.4350 0.8647
Max: 0.8647
Min: -0.8539
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.199
Min: 5.261e-16

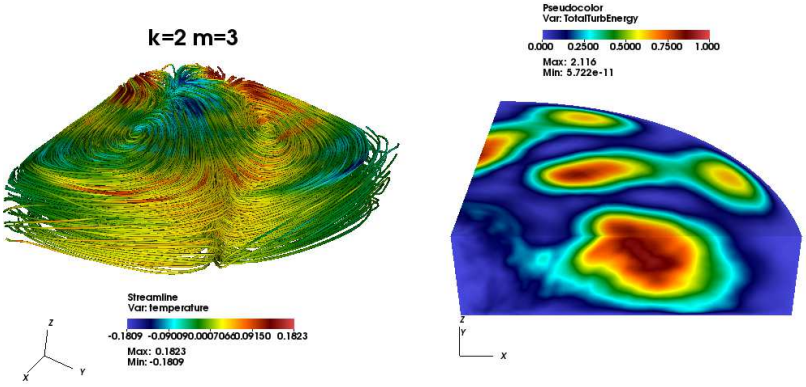

k=40 m=2
Streamline
Var: temperature
-1.733 -0.8737 -0.01414 0.8454 1.705
Max: 1.705
Min: -1.733
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.685
Min: 1.215e-15

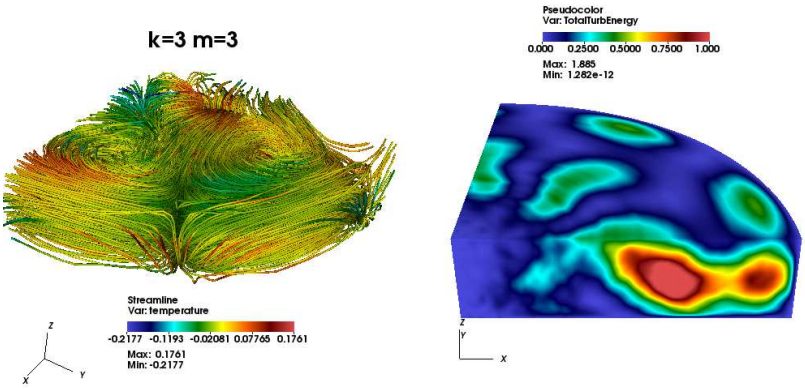

k=60 m=2
Streamline
Var: temperature
-2.319 -1.302 -0.2859 0.7306 1.747
Max: 1.747
Min: -2.319
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 8.354
Min: 7.200e-16

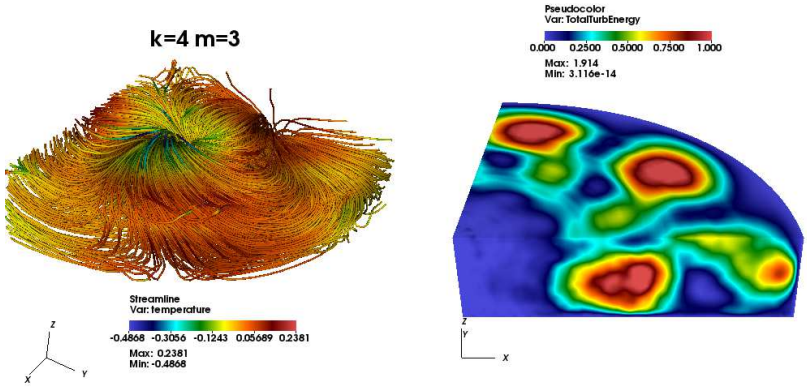

k=100 m=2
Streamline
Var: temperature
-3.353 -1.676 0.0001506 1.677 3.353
Max: 3.353
Min: -3.353
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 15.37
Min: 2.442e-16

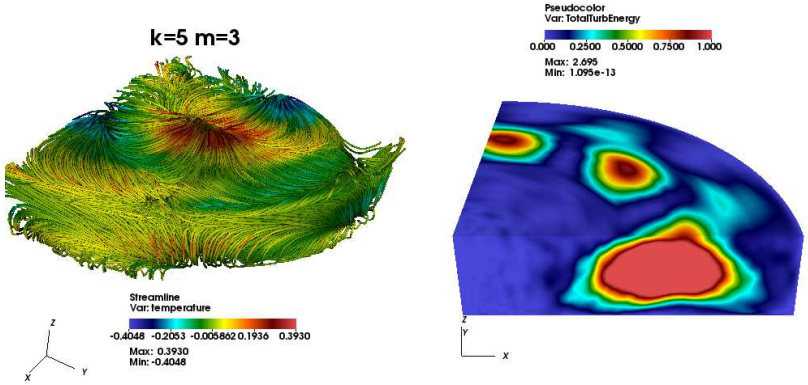

k=1000 m=2
Streamline
Var: temperature
-5.740 -2.736 0.2688 3.273 6.278
Max: 6.278
Min: -5.740
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 397.5
Min: 2.571e-15

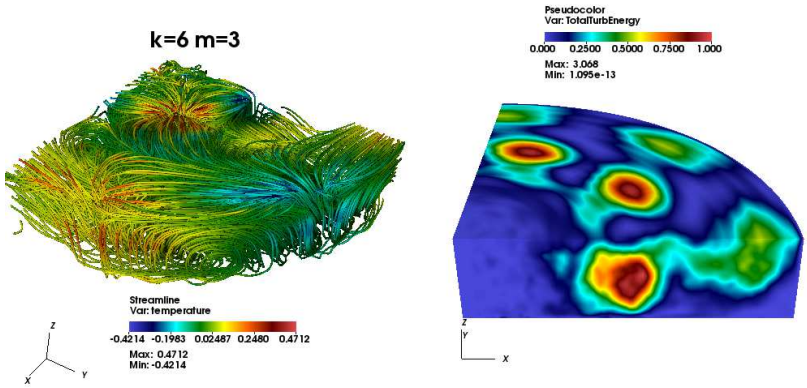

k=1 m=3
Streamline
Var: temperature
-0.1605 -0.07983 0.000861 0.08155 0.1622
Max: 0.1622
Min: -0.1605
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.480
Min: 3.379e-10

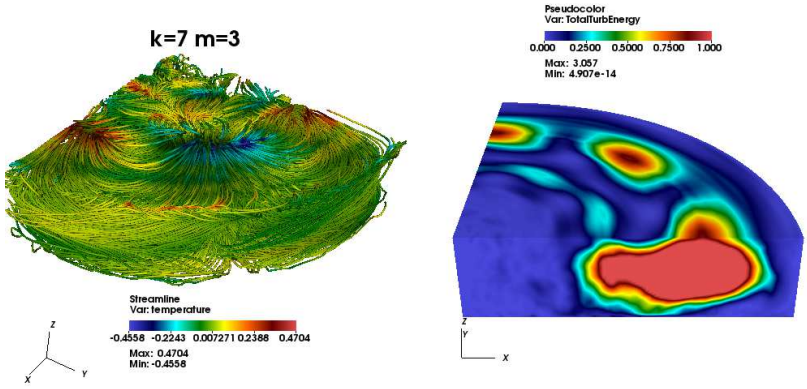

k=2 m=3
Streamline
Var: temperature
-0.1809 -0.09009 0.000706 0.09150 0.1823
Max: 0.1823
Min: -0.1809
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.116
Min: 5.722e-11

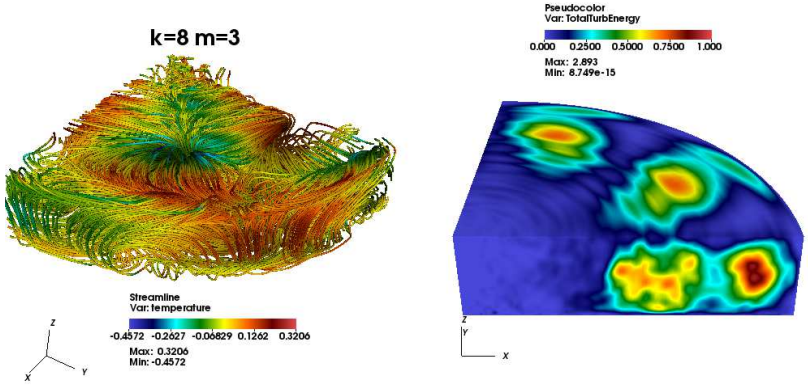

k=3 m=3
Streamline
Var: temperature
-0.2177 -0.1193 -0.02081 0.07765 0.1761
Max: 0.1761
Min: -0.2177
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.885
Min: 1.282e-12

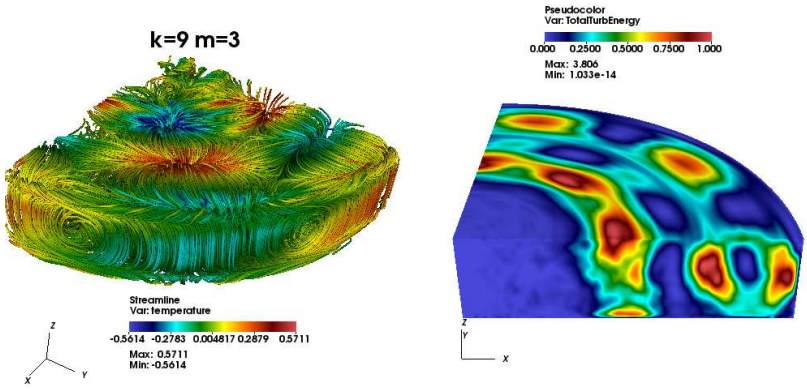

k=4 m=3
Streamline
Var: temperature
-0.4868 -0.3056 -0.1243 0.05689 0.2381
Max: 0.2381
Min: -0.4868
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.914
Min: 3.116e-14

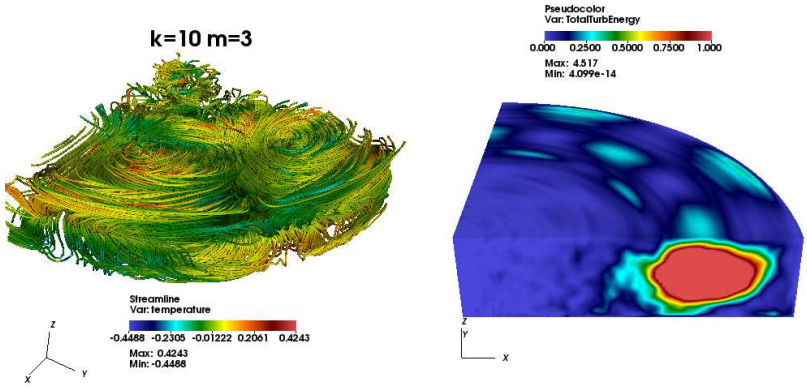

k=5 m=3
Streamline
Var: temperature
-0.4048 -0.2053 -0.005862 0.1936 0.3930
Max: 0.3930
Min: -0.4048
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.695
Min: 1.095e-13

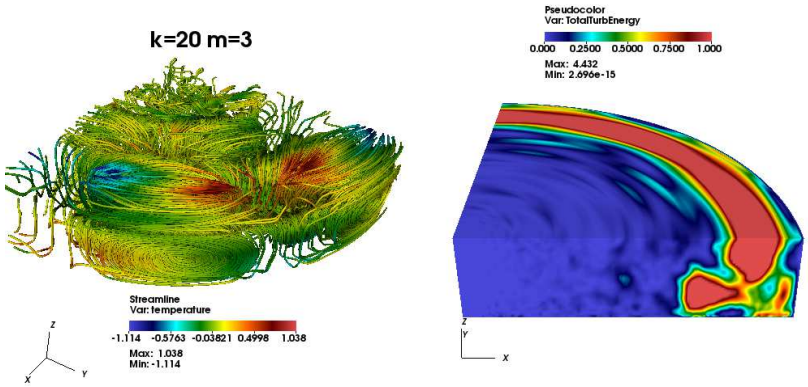

k=6 m=3
Streamline
Var: temperature
-0.4214 -0.1983 0.02487 0.2480 0.4712
Max: 0.4712
Min: -0.4214
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.068
Min: 1.095e-13

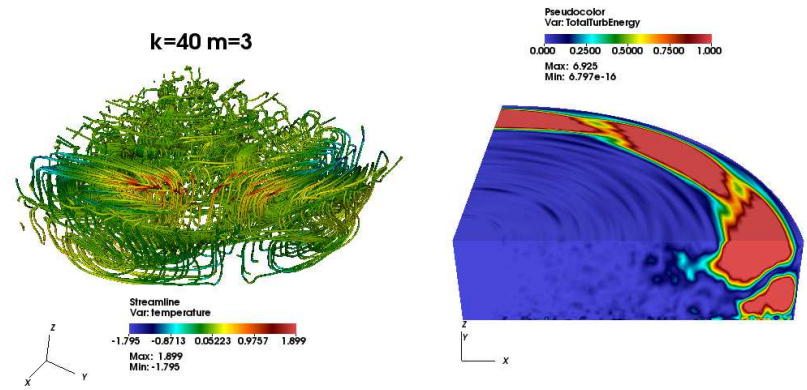

k=7 m=3
Streamline
Var: temperature
-0.4558 -0.2243 0.007271 0.2388 0.4704
Max: 0.4704
Min: -0.4558
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.057
Min: 4.907e-14

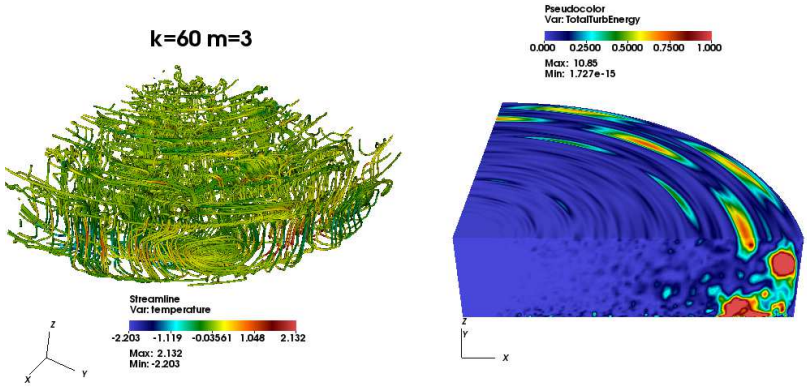

k=8 m=3
Streamline
Var: temperature
-0.4572 -0.2627 -0.06829 0.1262 0.3206
Max: 0.3206
Min: -0.4572
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.893
Min: 8.749e-15

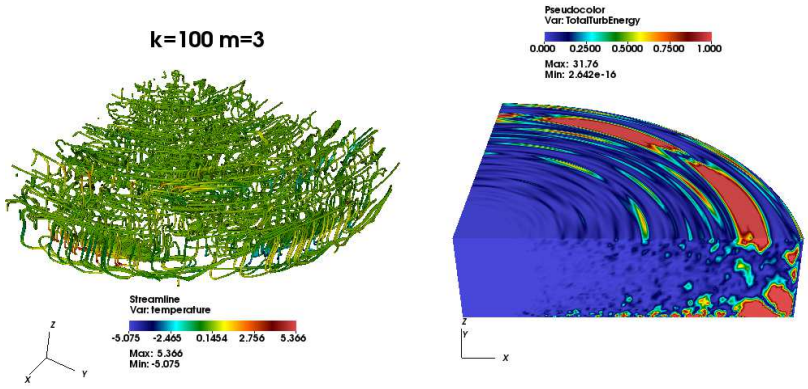

k=9 m=3
Streamline
Var: temperature
-0.5614 -0.2783 0.004817 0.2879 0.5711
Max: 0.5711
Min: -0.5614
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.806
Min: 1.033e-14

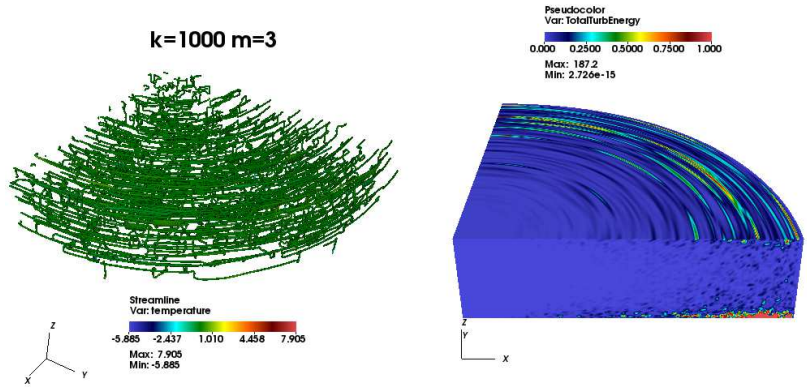

k=10 m=3
Streamline
Var: temperature
-0.4488 -0.2305 -0.01222 0.2061 0.4243
Max: 0.4243
Min: -0.4488
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.517
Min: 4.099e-14

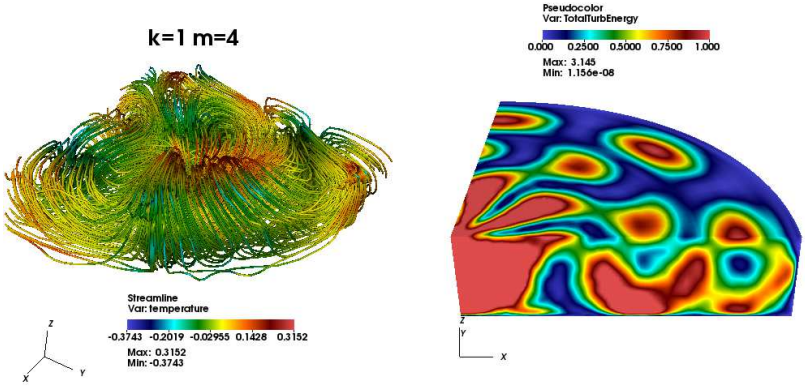

k=20 m=3
Streamline
Var: temperature
-1.114 -0.5763 -0.03821 0.4998 1.038
Max: 1.038
Min: -1.114
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.432
Min: 2.696e-15

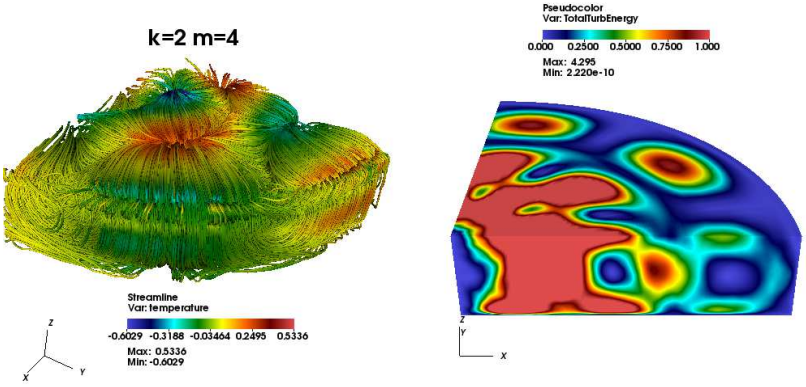

k=40 m=3
Streamline
Var: temperature
-1.795 -0.8713 0.05223 0.9757 1.899
Max: 1.899
Min: -1.795
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 6.925
Min: 6.797e-16

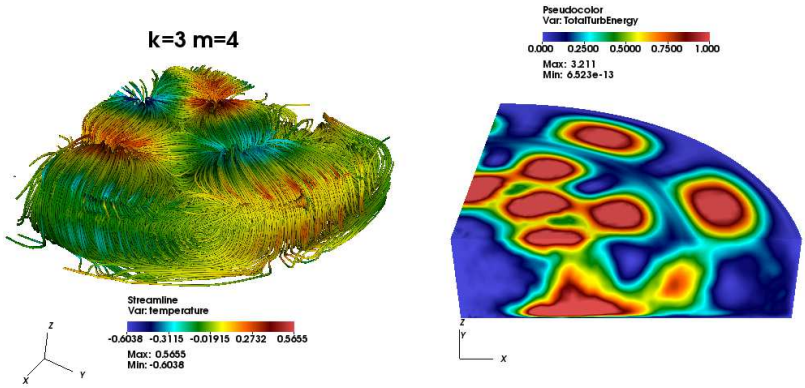

k=60 m=3
Streamline
Var: temperature
-2.203 -1.119 -0.03561 1.048 2.132
Max: 2.132
Min: -2.203
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 10.85
Min: 1.727e-15

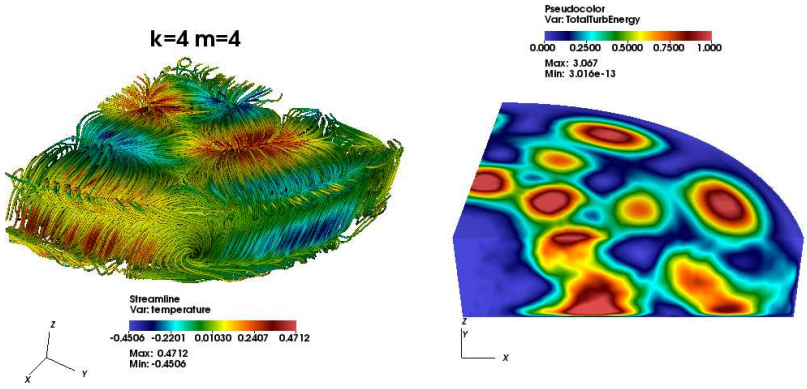

k=100 m=3
Streamline
Var: temperature
-5.075 -2.465 0.1454 2.756 5.366
Max: 5.366
Min: -5.075
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 31.76
Min: 2.642e-16

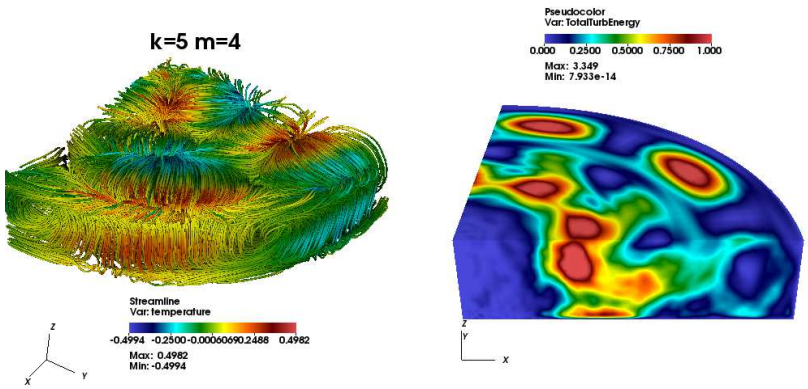

k=1000 m=3
Streamline
Var: temperature
-5.885 -2.437 1.010 4.458 7.905
Max: 7.905
Min: -5.885
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 187.2
Min: 2.726e-15

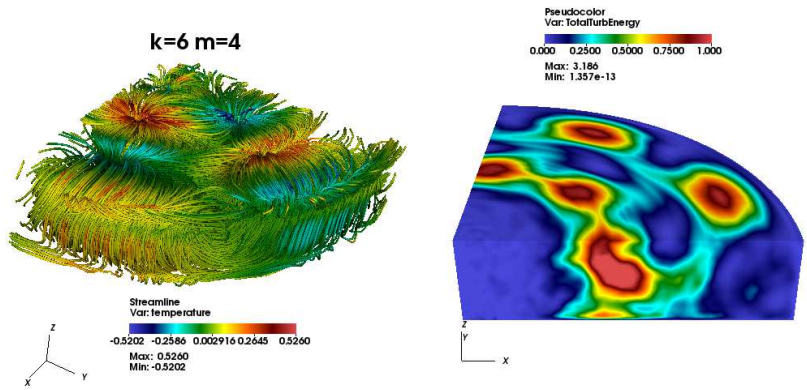

k=1 m=4
Streamline
Var: temperature
-0.3743 -0.2019 -0.02955 0.1428 0.3152
Max: 0.3152
Min: -0.3743
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.145
Min: 1.156e-08

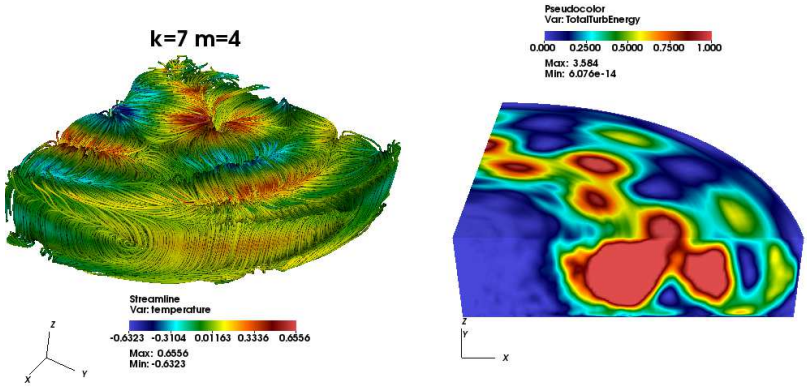

k=2 m=4
Streamline
Var: temperature
-0.6029 -0.3188 -0.03464 0.2495 0.5336
Max: 0.5336
Min: -0.6029
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.295
Min: 2.220e-10

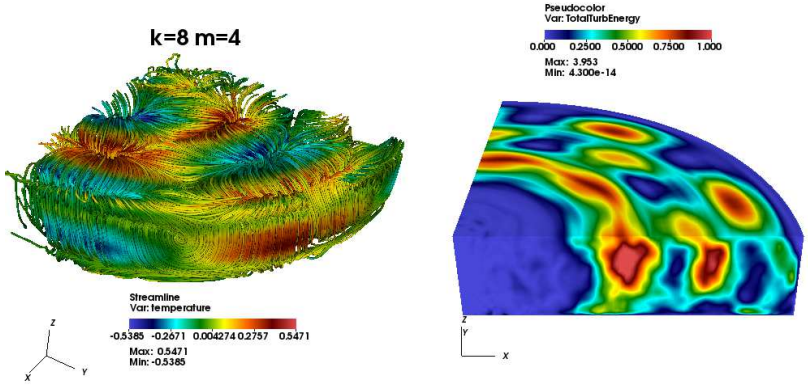

k=3 m=4
Streamline
Var: temperature
-0.6038 -0.3115 -0.01915 0.2732 0.5655
Max: 0.5655
Min: -0.6038
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.211
Min: 6.523e-13

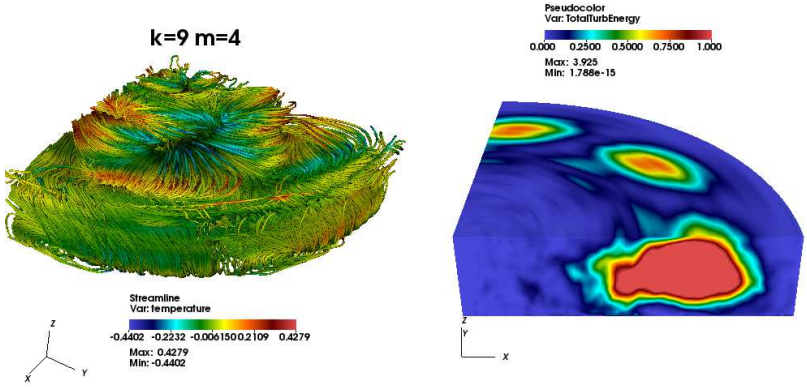

k=4 m=4
Streamline
Var: temperature
-0.4506 -0.2201 0.01030 0.2407 0.4712
Max: 0.4712
Min: -0.4506
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.067
Min: 3.016e-13

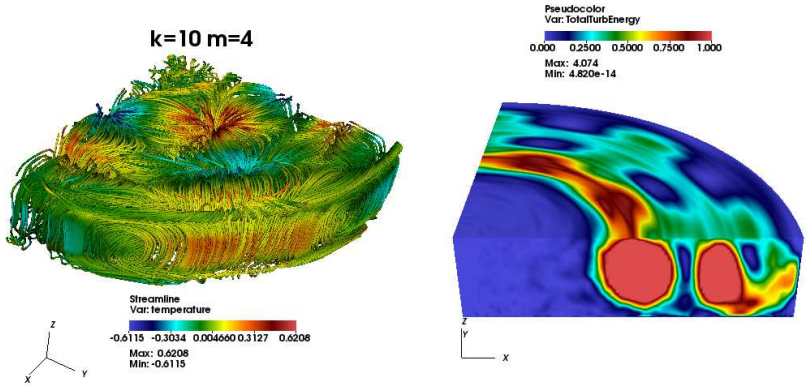

k=5 m=4
Streamline
Var: temperature
-0.4994 -0.2500 -0.00060690.2488 0.4982
Max: 0.4982
Min: -0.4994
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.349
Min: 7.933e-14

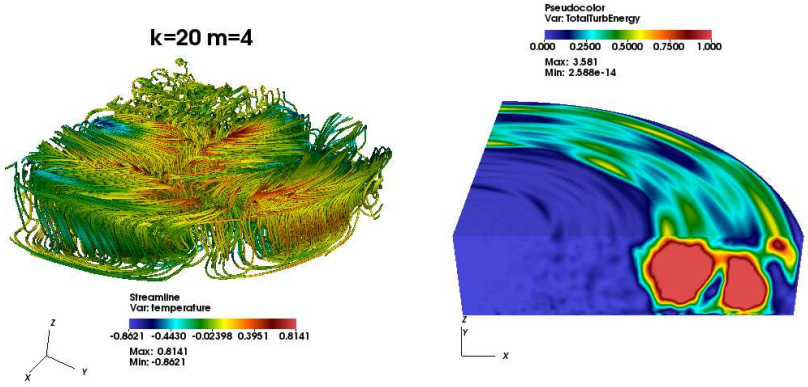

k=6 m=4
Streamline
Var: temperature
-0.5202 -0.2586 0.002916 0.2645 0.5260
Max: 0.5260
Min: -0.5202
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.186
Min: 1.357e-13

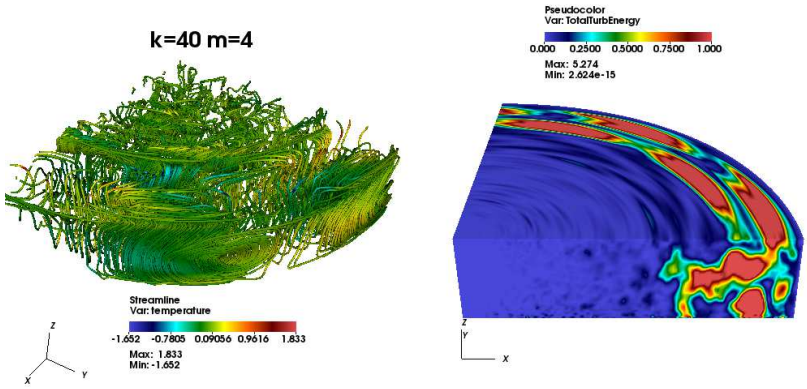

k=7 m=4
Streamline
Var: temperature
-0.6323 -0.3104 0.01163 0.3336 0.6556
Max: 0.6556
Min: -0.6323
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.584
Min: 6.076e-14

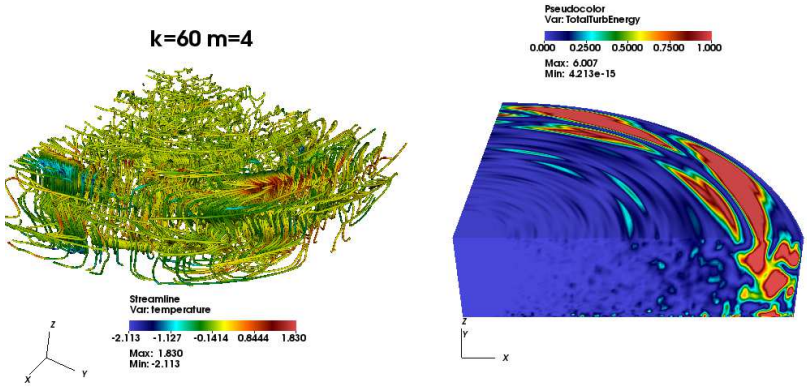

k=8 m=4
Streamline
Var: temperature
-0.5385 -0.2671 0.004274 0.2757 0.5471
Max: 0.5471
Min: -0.5385
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.953
Min: 4.300e-14

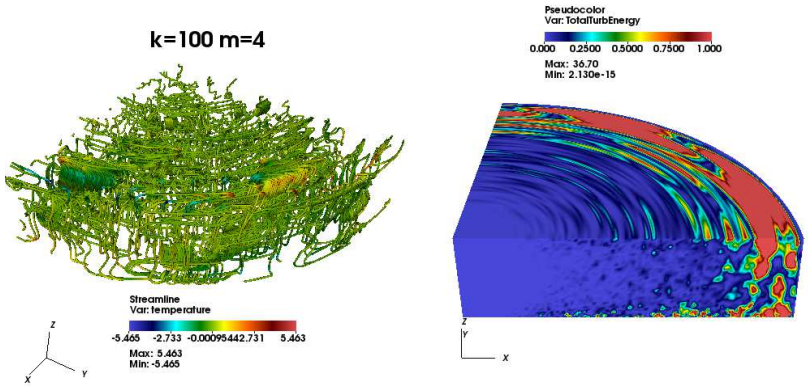

k=9 m=4
Streamline
Var: temperature
-0.4402 -0.2232 -0.0061500.2109 0.4279
Max: 0.4279
Min: -0.4402
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.925
Min: 1.788e-15

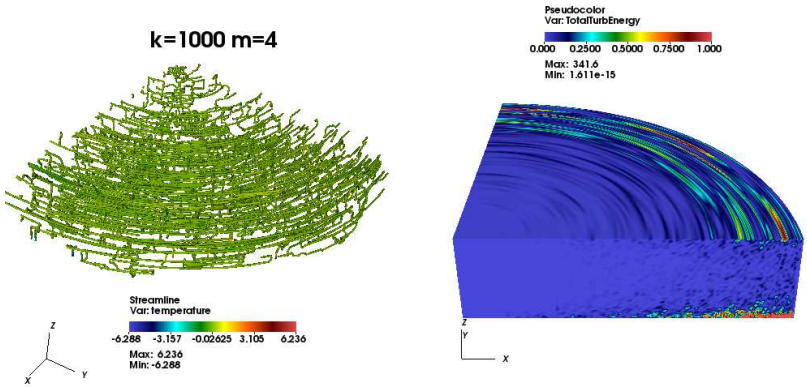

k=10 m=4
Streamline
Var: temperature
-0.6115 -0.3034 0.004660 0.3127 0.6208
Max: 0.6208
Min: -0.6115
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.074
Min: 4.820e-14

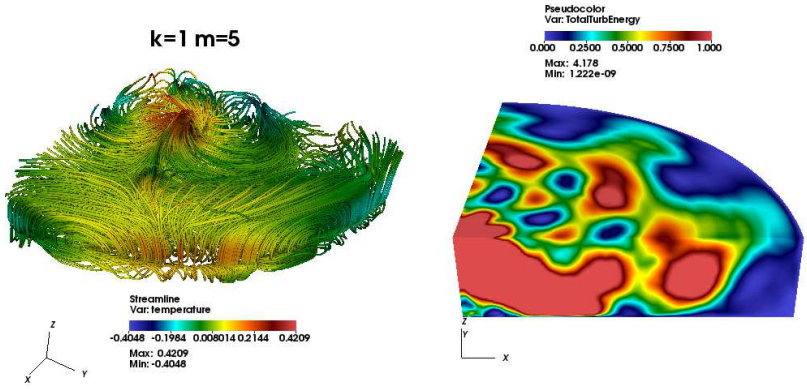

k=20 m=4
Streamline
Var: temperature
-0.8621 -0.4430 -0.02398 0.3951 0.8141
Max: 0.8141
Min: -0.8621
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.581
Min: 2.588e-14

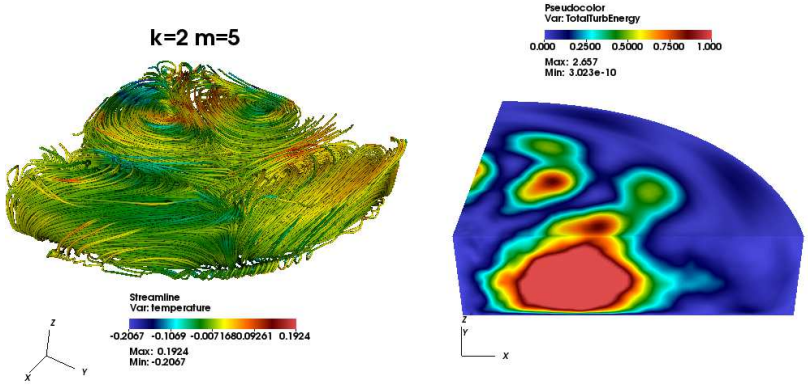

k=40 m=4
Streamline
Var: temperature
-1.652 -0.7805 0.09056 0.9616 1.833
Max: 1.833
Min: -1.652
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.274
Min: 2.624e-15

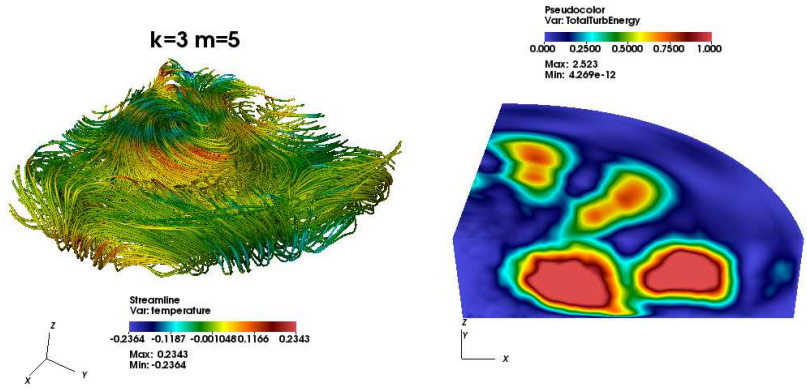

k=60 m=4
Streamline
Var: temperature
-2.113 -1.127 -0.1414 0.8444 1.830
Max: 1.830
Min: -2.113
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 6.007
Min: 4.213e-15

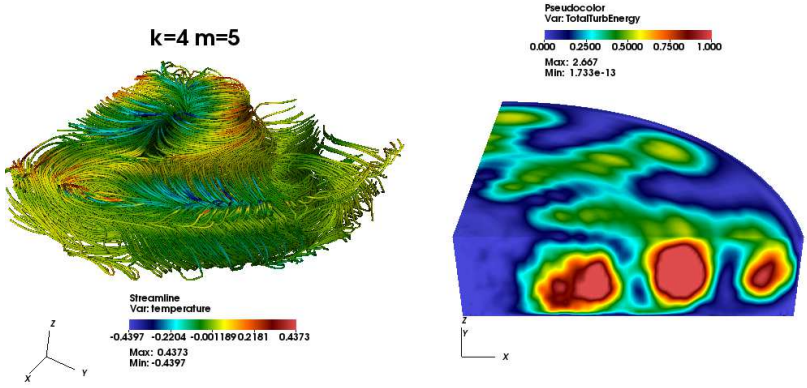

k=100 m=4
Streamline
Var: temperature
-5.465 -2.733 -0.00095442.731 5.463
Max: 5.463
Min: -5.465
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 36.70
Min: 2.130e-15

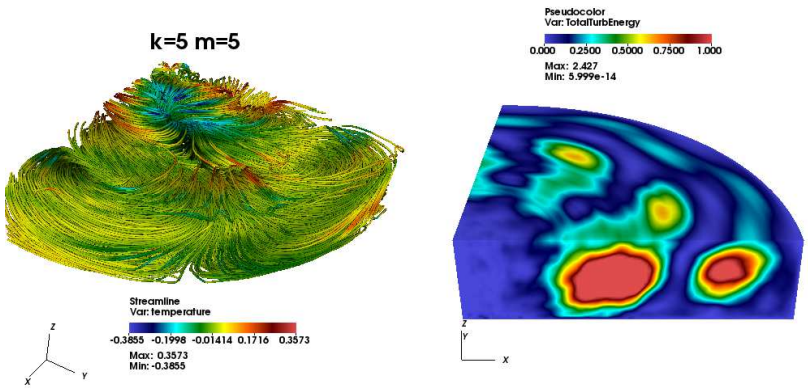

k=1000 m=4
Streamline
Var: temperature
-6.288 -3.157 -0.02625 3.105 6.236
Max: 6.236
Min: -6.288
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 341.6
Min: 1.611e-15

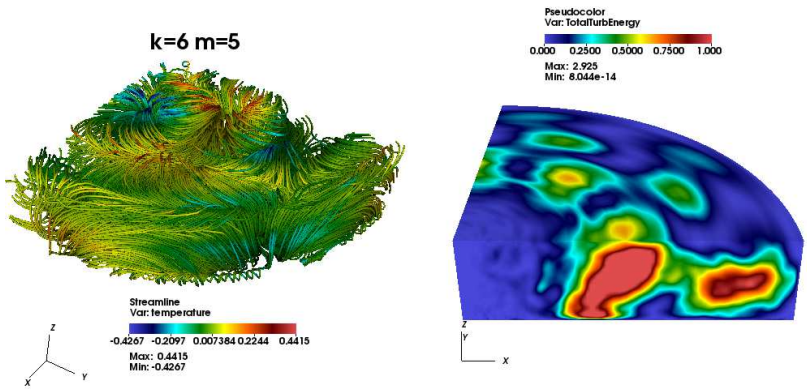

k=1 m=5
Streamline
Var: temperature
-0.4048 -0.1984 0.008014 0.2144 0.4209
Max: 0.4209
Min: -0.4048
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.178
Min: 1.222e-09

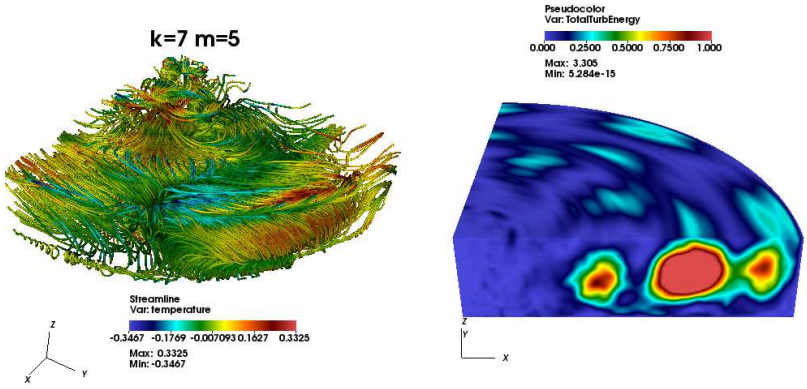

k=2 m=5
Streamline
Var: temperature
-0.2067 -0.1069 -0.007168 0.09261 0.1924
Max: 0.1924
Min: -0.2067
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.657
Min: 3.023e-10

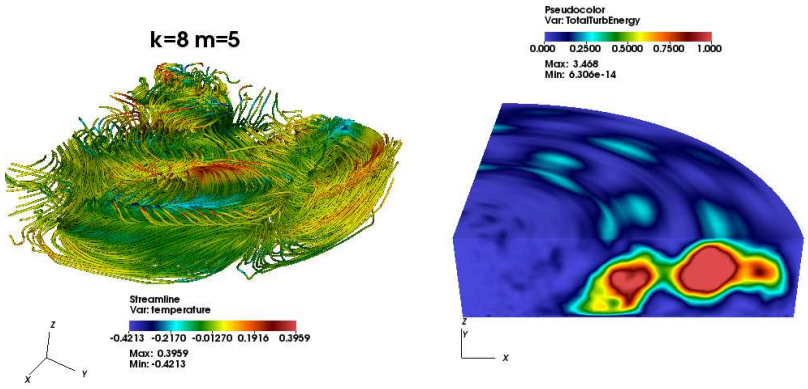

k=3 m=5
Streamline
Var: temperature
-0.2364 -0.1187 -0.001048 0.1166 0.2343
Max: 0.2343
Min: -0.2364
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.523
Min: 4.269e-12

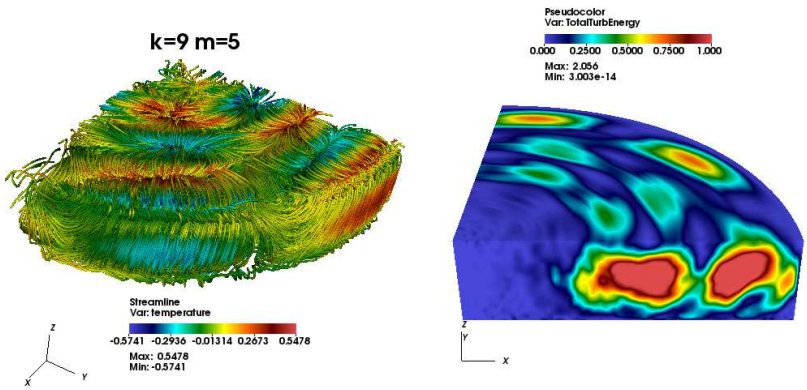

k=4 m=5
Streamline
Var: temperature
-0.4397 -0.2204 -0.001189 0.2181 0.4373
Max: 0.4373
Min: -0.4397
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.667
Min: 1.733e-13

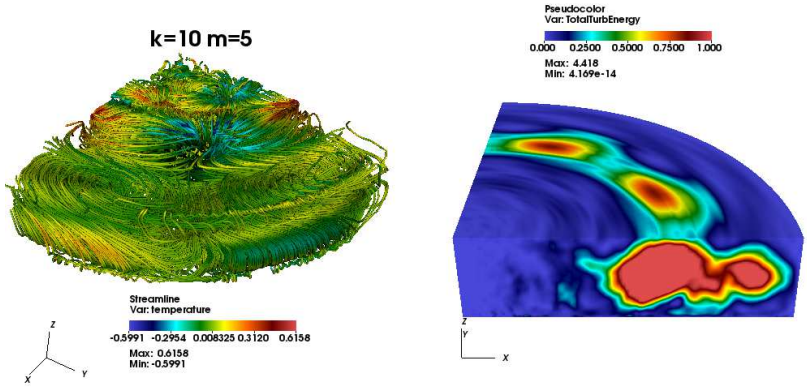

k=5 m=5
Streamline
Var: temperature
-0.3855 -0.1998 -0.01414 0.1716 0.3573
Max: 0.3573
Min: -0.3855
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.427
Min: 5.999e-14

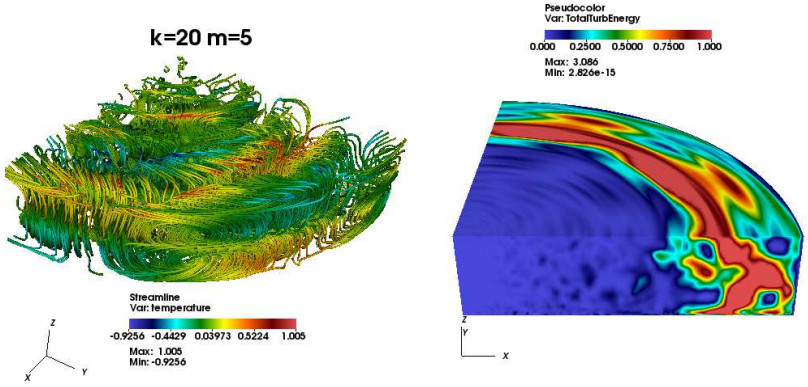

k=6 m=5
Streamline
Var: temperature
-0.4267 -0.2097 0.007384 0.2244 0.4415
Max: 0.4415
Min: -0.4267
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.925
Min: 8.044e-14

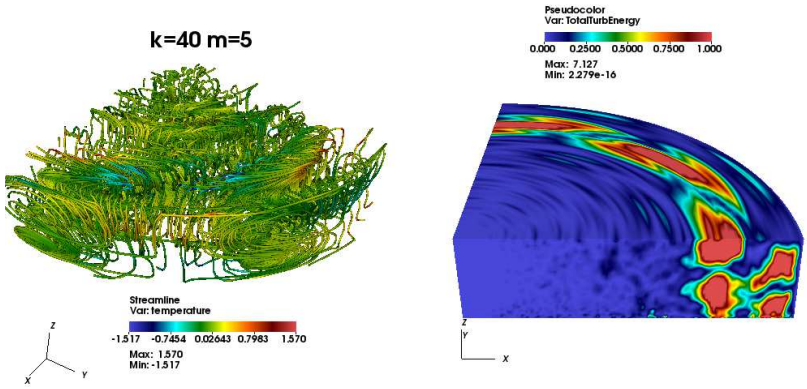

k=7 m=5
Streamline
Var: temperature
-0.3467 -0.1769 -0.007093 0.1627 0.3325
Max: 0.3325
Min: -0.3467
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.305
Min: 5.284e-15

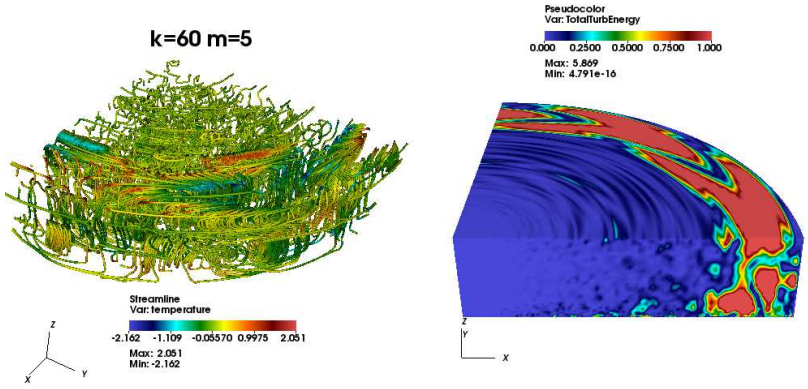

k=8 m=5
Streamline
Var: temperature
-0.4213 -0.2170 -0.01270 0.1916 0.3959
Max: 0.3959
Min: -0.4213
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.468
Min: 6.306e-14

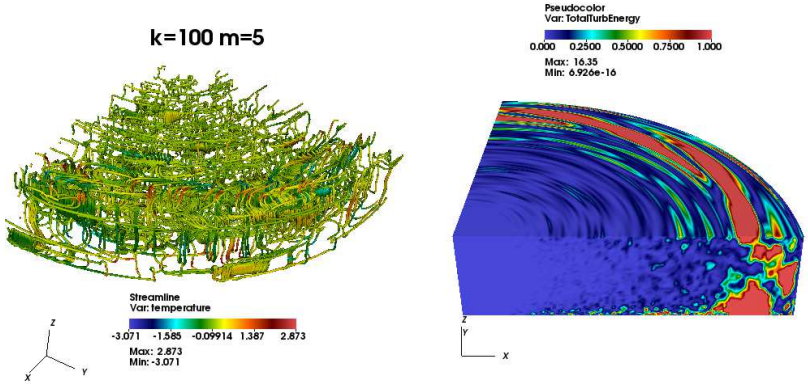

k=9 m=5
Streamline
Var: temperature
-0.5741 -0.2936 -0.01314 0.2673 0.5478
Max: 0.5478
Min: -0.5741
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.056
Min: 3.003e-14

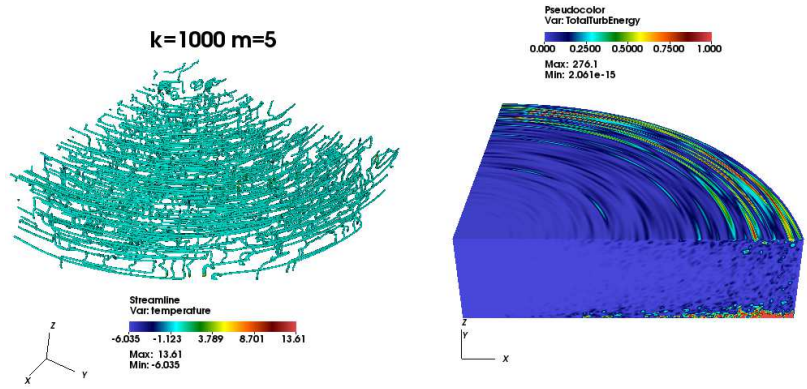

k=10 m=5
Streamline
Var: temperature
-0.5991 -0.2954 0.008325 0.3120 0.6158
Max: 0.6158
Min: -0.5991
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.418
Min: 4.169e-14

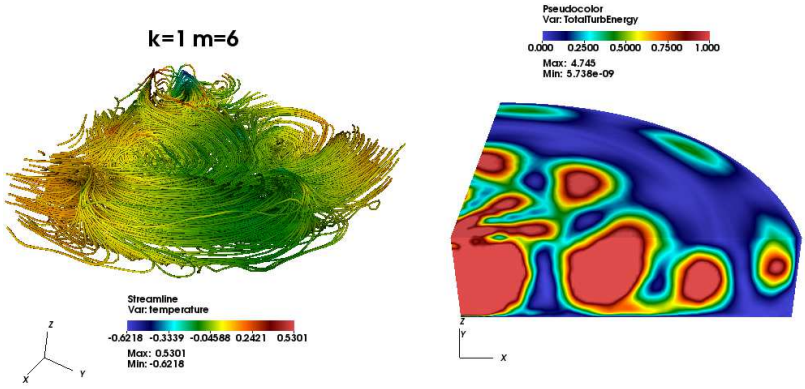

k=20 m=5
Streamline
Var: temperature
-0.9256 -0.4429 0.03973 0.5224 1.005
Max: 1.005
Min: -0.9256
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.086
Min: 2.826e-15

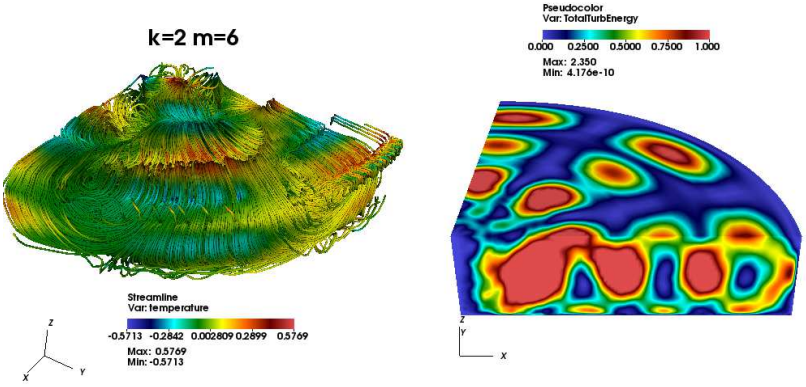

k=40 m=5
Streamline
Var: temperature
-1.517 -0.7454 0.02643 0.7983 1.570
Max: 1.570
Min: -1.517
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 7.127
Min: 2.279e-16

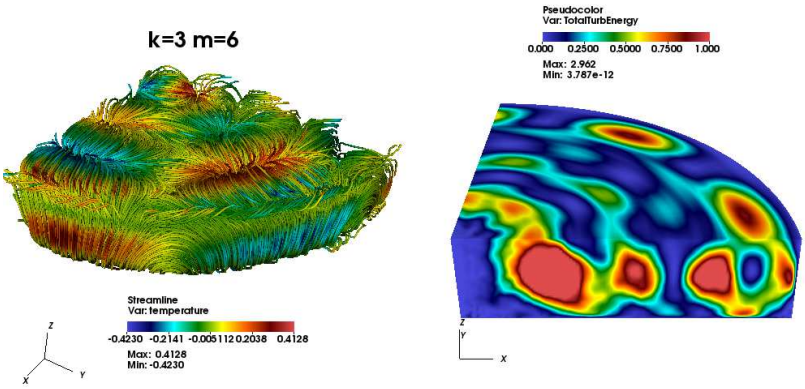

k=60 m=5
Streamline
Var: temperature
-2.162 -1.109 -0.05570 0.9975 2.051
Max: 2.051
Min: -2.162
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.869
Min: 4.791e-16

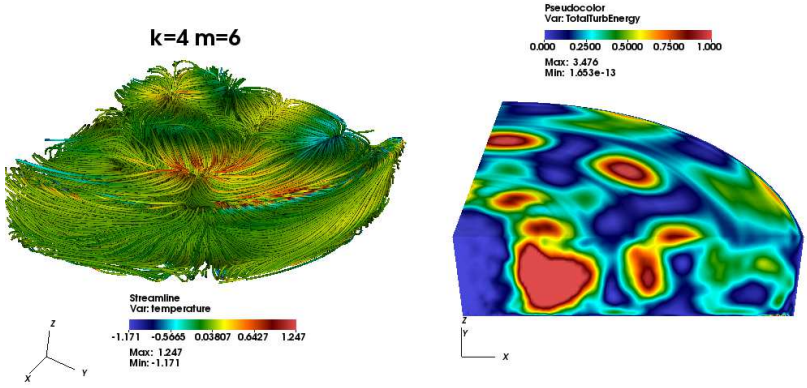

k=100 m=5
Streamline
Var: temperature
-3.071 -1.585 -0.09914 1.387 2.873
Max: 2.873
Min: -3.071
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 16.35
Min: 6.926e-16

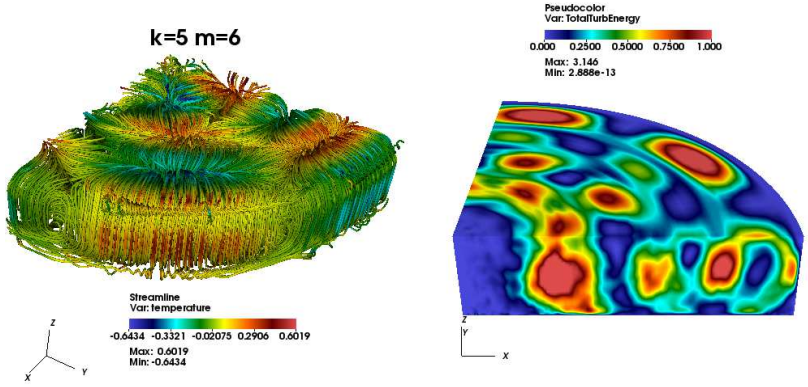

k=1000 m=5
Streamline
Var: temperature
-6.035 -1.123 3.789 8.701 13.61
Max: 13.61
Min: -6.035
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 276.1
Min: 2.061e-15

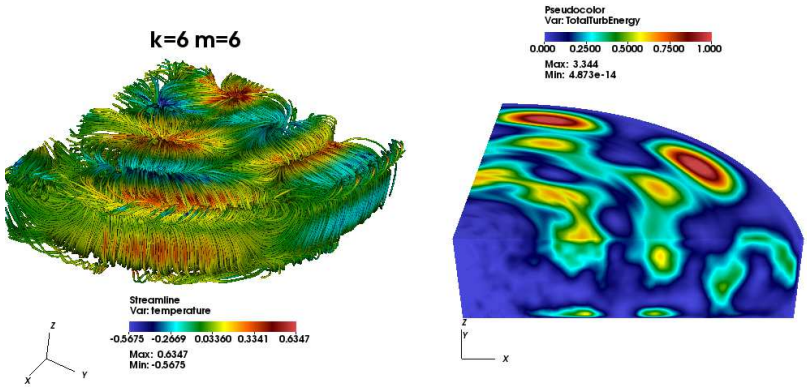

k=1 m=6
Streamline
Var: temperature
-0.6218 -0.3339 -0.04588 0.2421 0.5301
Max: 0.5301
Min: -0.6218
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.745
Min: 5.738e-09

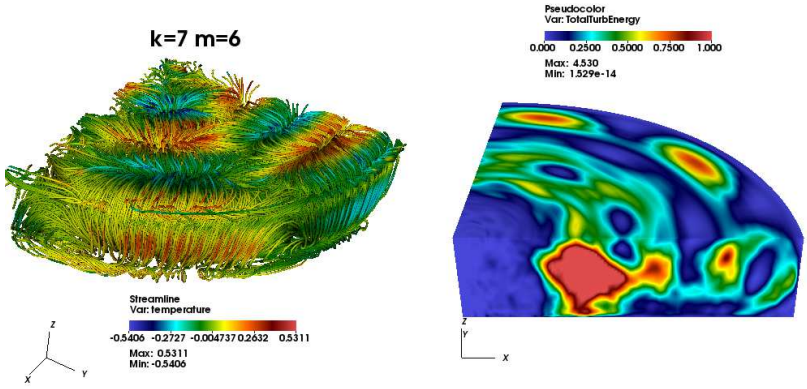

k=2 m=6
Streamline
Var: temperature
-0.5713 -0.2842 0.002809 0.2899 0.5769
Max: 0.5769
Min: -0.5713
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.350
Min: 4.176e-10

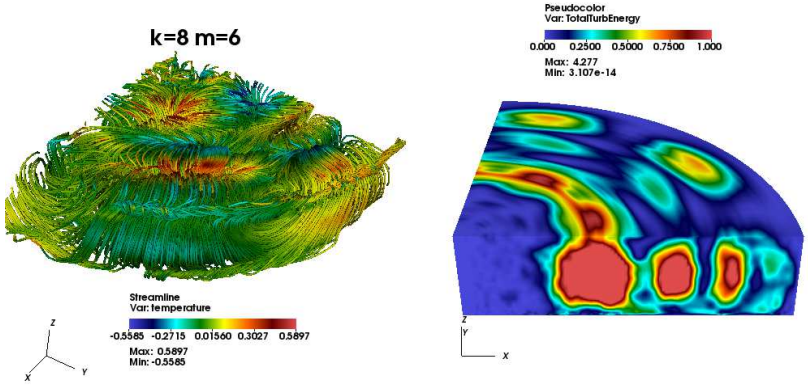

k=3 m=6
Streamline
Var: temperature
-0.4230 -0.2141 -0.005112 0.2038 0.4128
Max: 0.4128
Min: -0.4230
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.962
Min: 3.787e-12

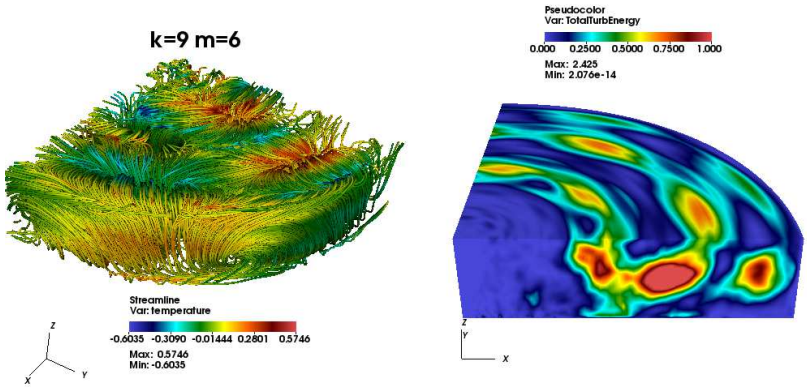

k=4 m=6
Streamline
Var: temperature
-1.171 -0.5665 0.03807 0.6427 1.247
Max: 1.247
Min: -1.171
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.476
Min: 1.653e-13

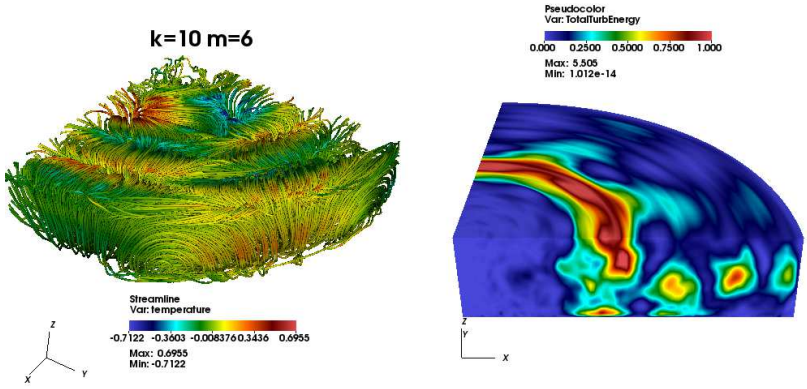

k=5 m=6
Streamline
Var: temperature
-0.6434 -0.3321 -0.02075 0.2906 0.6019
Max: 0.6019
Min: -0.6434
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.146
Min: 2.888e-13

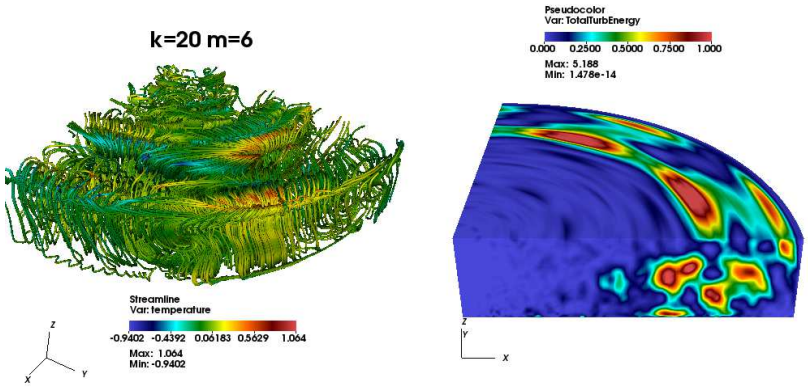

k=6 m=6
Streamline
Var: temperature
-0.5675 -0.2669 0.03360 0.3341 0.6347
Max: 0.6347
Min: -0.5675
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.344
Min: 4.873e-14

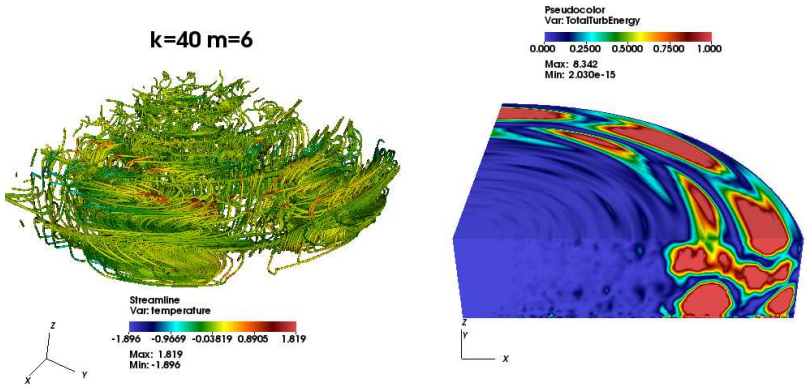

k=7 m=6
Streamline
Var: temperature
-0.5406 -0.2727 -0.004737 0.2632 0.5311
Max: 0.5311
Min: -0.5406
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.530
Min: 1.529e-14

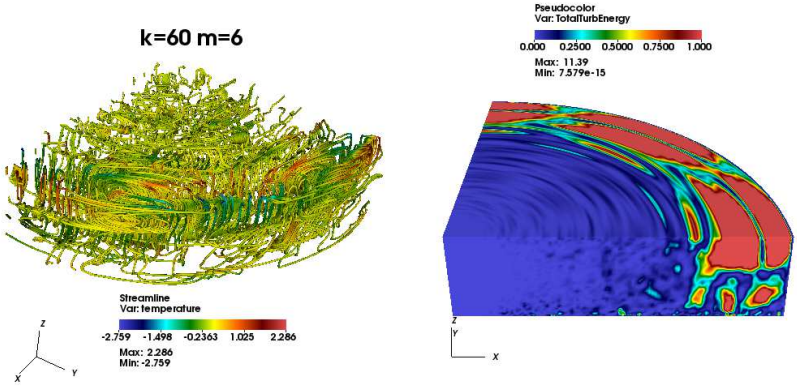

k=8 m=6
Streamline
Var: temperature
-0.5585 -0.2715 0.01560 0.3027 0.5897
Max: 0.5897
Min: -0.5585
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.277
Min: 3.107e-14

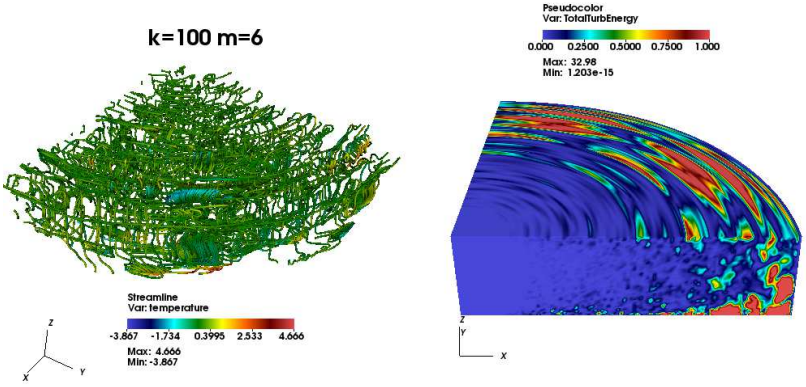

k=9 m=6
Streamline
Var: temperature
-0.6035 -0.3090 -0.01444 0.2801 0.5746
Max: 0.5746
Min: -0.6035
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.425
Min: 2.076e-14

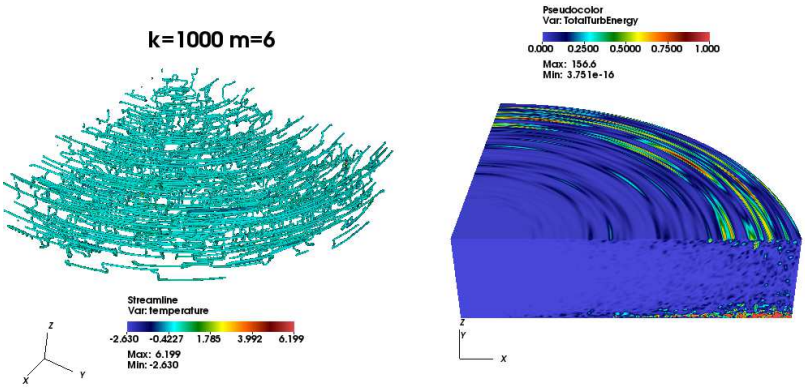

k=10 m=6
Streamline
Var: temperature
-0.7122 -0.3603 -0.008376 0.3436 0.6955
Max: 0.6955
Min: -0.7122
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.505
Min: 1.012e-14

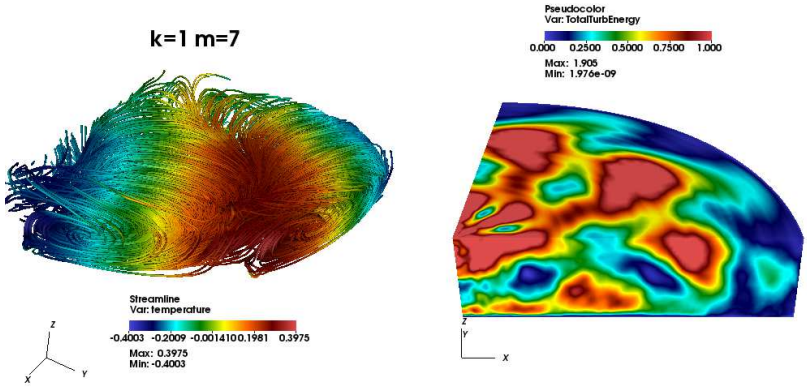

k=20 m=6
Streamline
Var: temperature
-0.9402 -0.4392 0.06183 0.5629 1.064
Max: 1.064
Min: -0.9402
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.188
Min: 1.478e-14

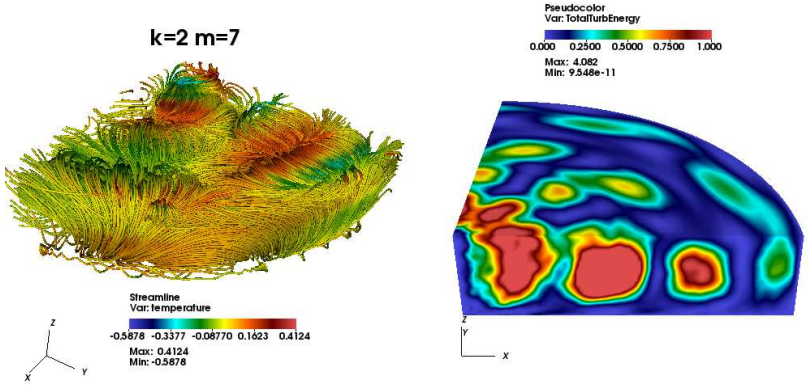

k=40 m=6
Streamline
Var: temperature
-1.896 -0.9669 -0.03819 0.8905 1.819
Max: 1.819
Min: -1.896
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 8.342
Min: 2.030e-15

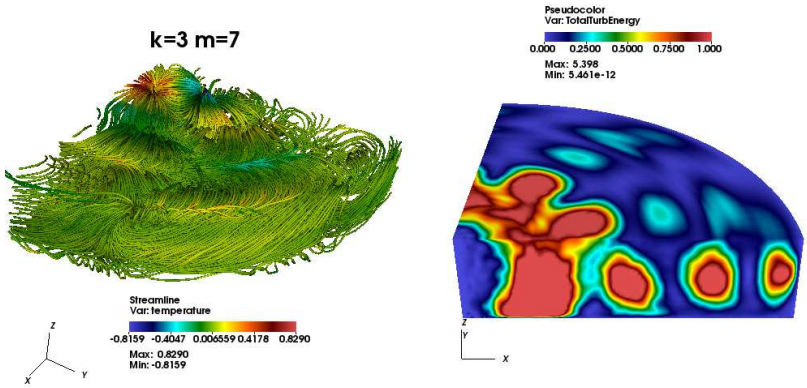

k=60 m=6
Streamline
Var: temperature
-2.759 -1.498 -0.2363 1.025 2.286
Max: 2.286
Min: -2.759
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 11.39
Min: 7.579e-15

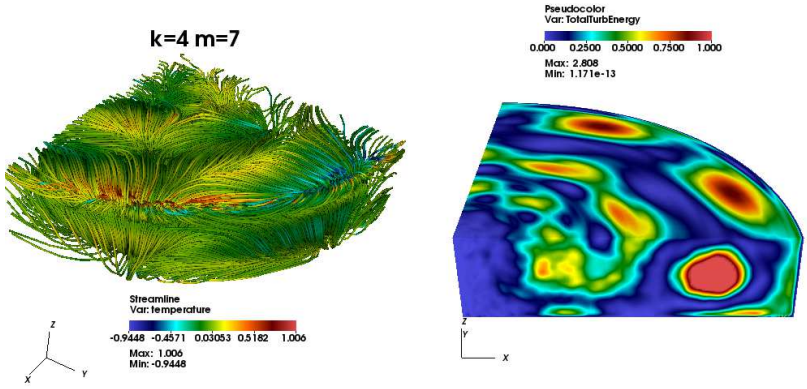

k=100 m=6
Streamline
Var: temperature
-3.867 -1.734 0.3995 2.533 4.666
Max: 4.666
Min: -3.867
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 32.98
Min: 1.203e-15

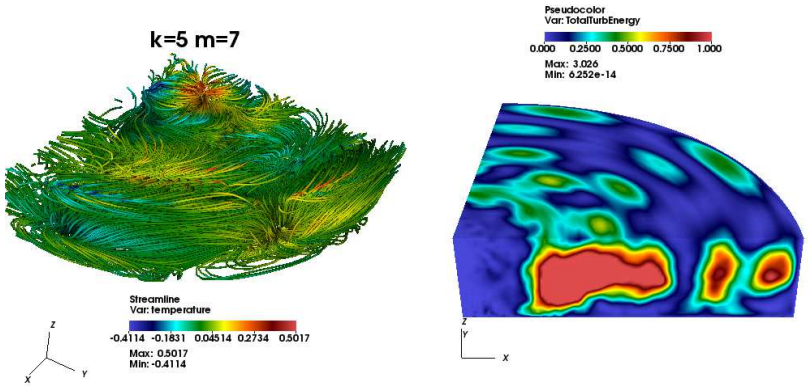

k=1000 m=6
Streamline
Var: temperature
-2.630 -0.4227 1.785 3.992 6.199
Max: 6.199
Min: -2.630
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 156.6
Min: 3.751e-16

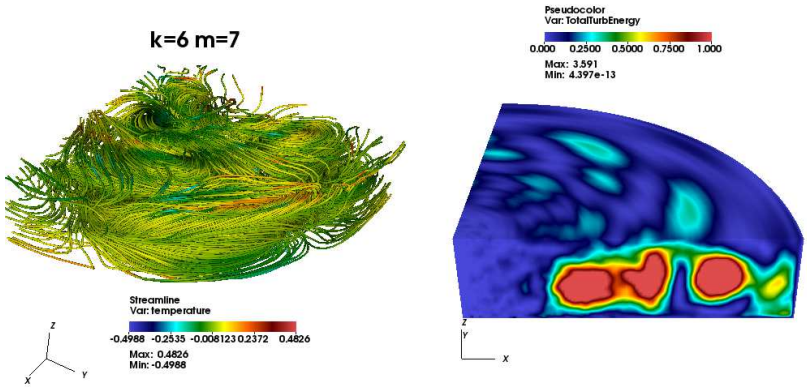

k=1 m=7
Streamline
Var: temperature
-0.4003 -0.2009 -0.0014100.1981 0.3975
Max: 0.3975
Min: -0.4003
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.905
Min: 1.976e-09

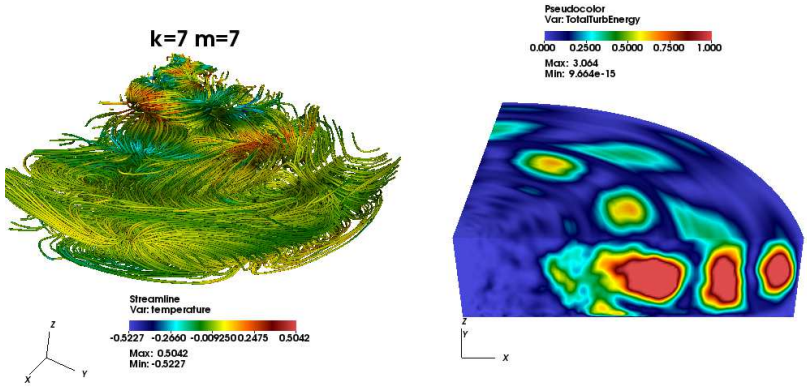

k=2 m=7
Streamline
Var: temperature
-0.5878 -0.3377 -0.08770 0.1623 0.4124
Max: 0.4124
Min: -0.5878
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.082
Min: 9.548e-11

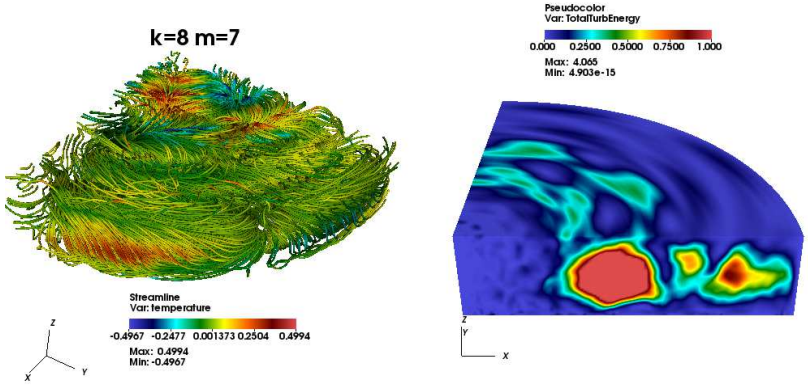

k=3 m=7
Streamline
Var: temperature
-0.8159 -0.4047 0.006559 0.4178 0.8290
Max: 0.8290
Min: -0.8159
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.398
Min: 5.461e-12

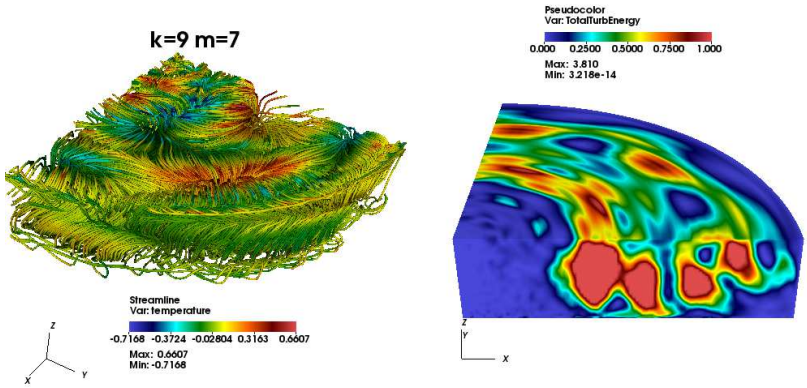

k=4 m=7
Streamline
Var: temperature
-0.9448 -0.4571 0.03053 0.5182 1.006
Max: 1.006
Min: -0.9448
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.808
Min: 1.171e-13

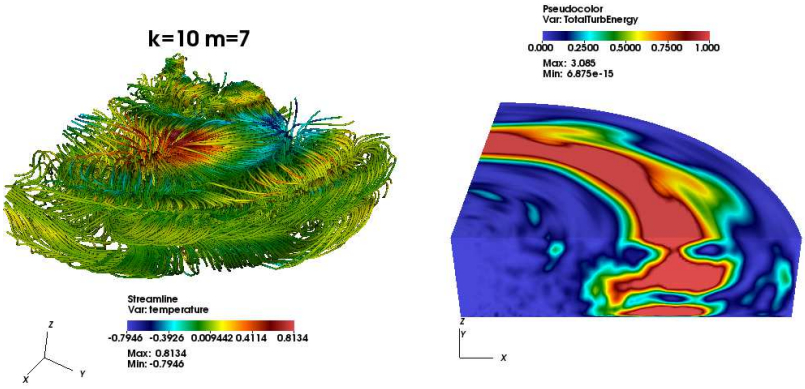

k=5 m=7
Streamline
Var: temperature
-0.4114 -0.1831 0.04514 0.2734 0.5017
Max: 0.5017
Min: -0.4114
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.026
Min: 6.252e-14

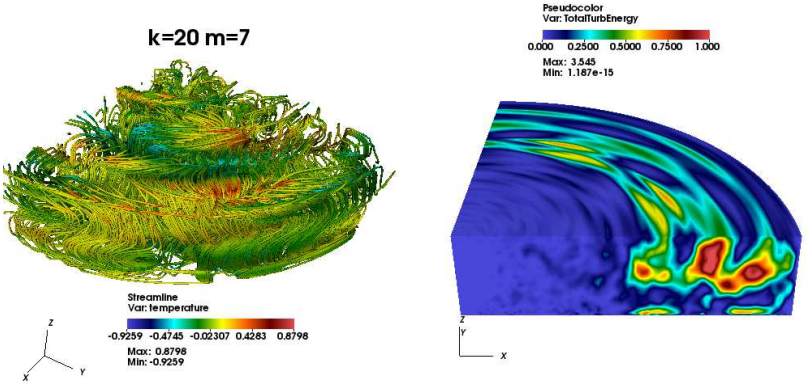

k=6 m=7
Streamline
Var: temperature
-0.4988 -0.2535 -0.008123 0.2372 0.4826
Max: 0.4826
Min: -0.4988
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.591
Min: 4.397e-13

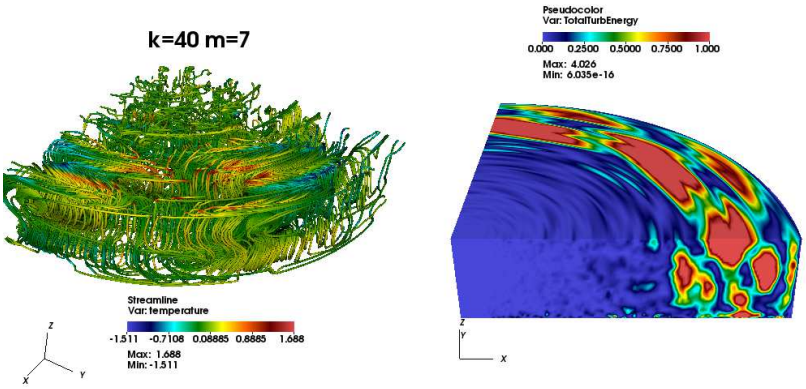

k=7 m=7
Streamline
Var: temperature
-0.5227 -0.2660 -0.009250 0.2475 0.5042
Max: 0.5042
Min: -0.5227
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.064
Min: 9.664e-15

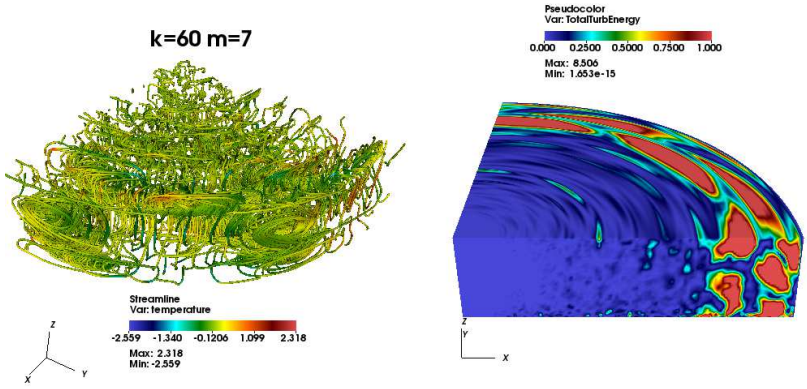

k=8 m=7
Streamline
Var: temperature
-0.4967 -0.2477 0.001373 0.2504 0.4994
Max: 0.4994
Min: -0.4967
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.065
Min: 4.903e-15

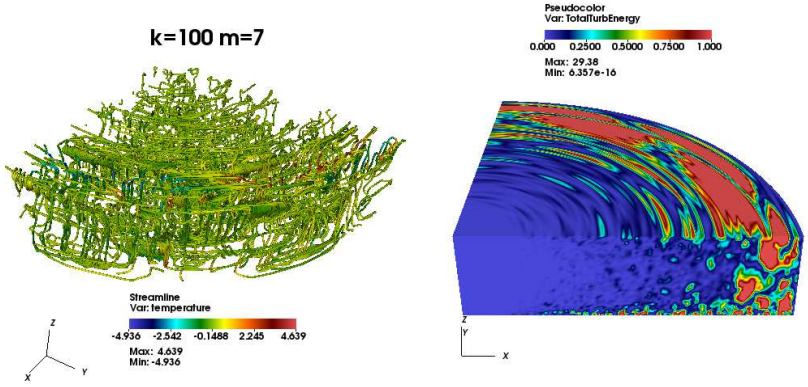

k=9 m=7
Streamline
Var: temperature
-0.7168 -0.3724 -0.02804 0.3163 0.6607
Max: 0.6607
Min: -0.7168
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.810
Min: 3.218e-14

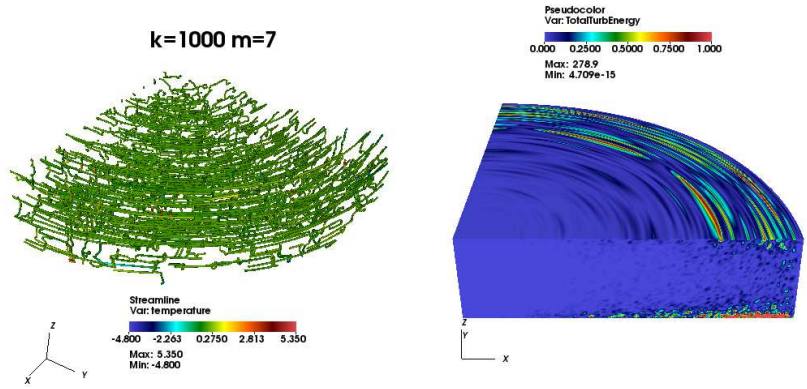

k=10 m=7
Streamline
Var: temperature
-0.7946 -0.3926 0.009442 0.4114 0.8134
Max: 0.8134
Min: -0.7946
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.085
Min: 6.875e-15

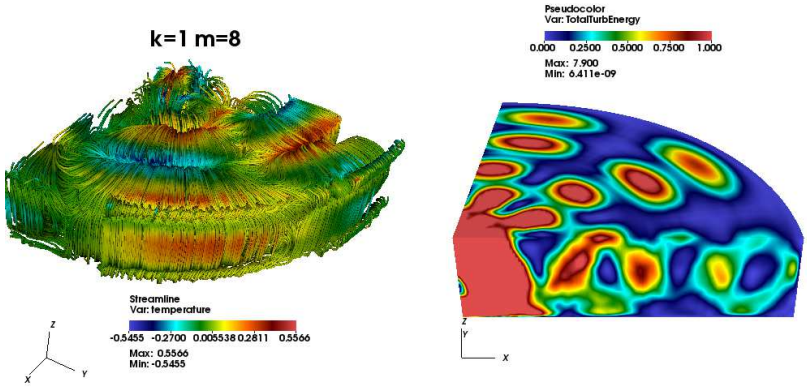

k=20 m=7
Streamline
Var: temperature
-0.9259 -0.4745 -0.02307 0.4283 0.8798
Max: 0.8798
Min: -0.9259
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.545
Min: 1.187e-15

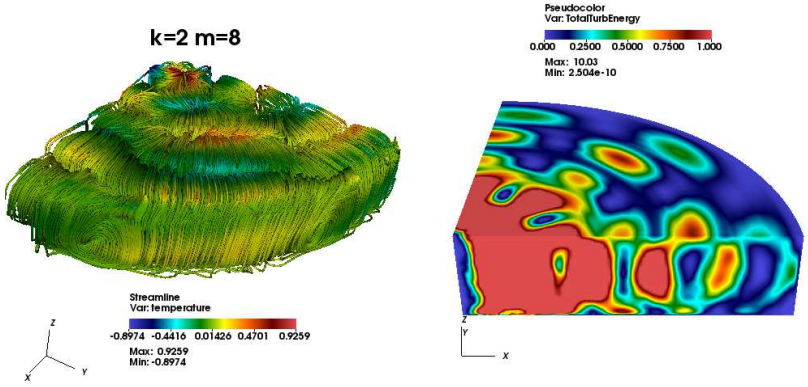

k=40 m=7
Streamline
Var: temperature
-1.511 -0.7108 0.08885 0.8885 1.688
Max: 1.688
Min: -1.511
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.026
Min: 6.035e-16

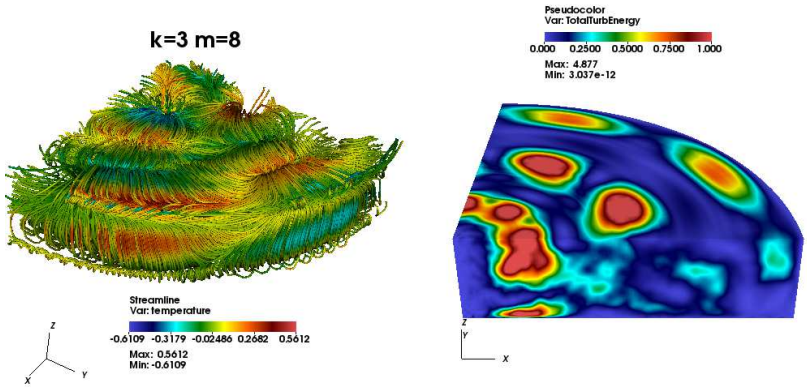

k=60 m=7
Streamline
Var: temperature
-2.559 -1.340 -0.1206 1.099 2.318
Max: 2.318
Min: -2.559
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 8.506
Min: 1.653e-15

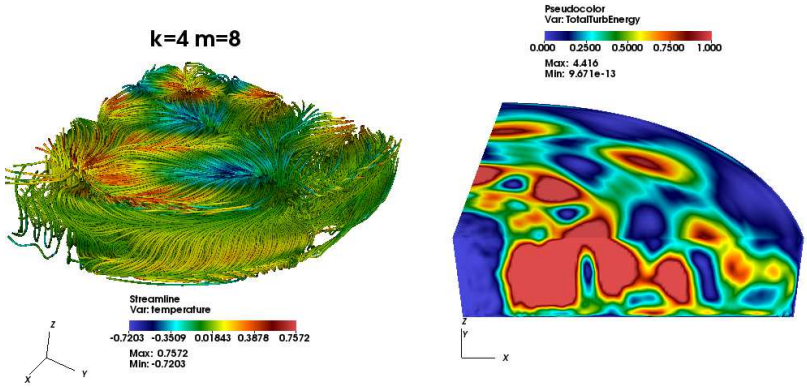

k=100 m=7
Streamline
Var: temperature
-4.936 -2.542 -0.1488 2.245 4.639
Max: 4.639
Min: -4.936
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 29.38
Min: 6.357e-16

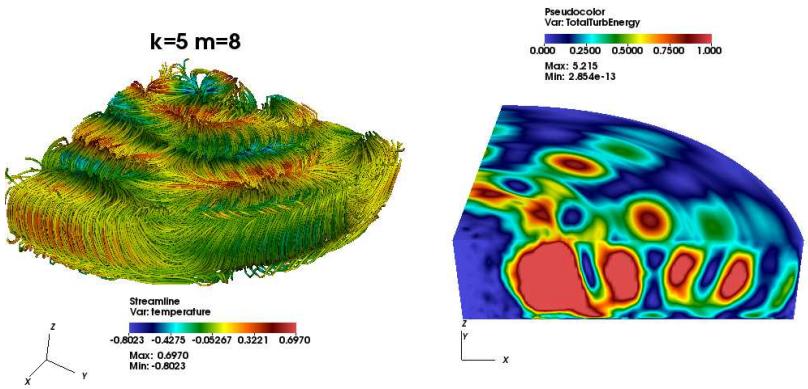

k=1000 m=7
Streamline
Var: temperature
-4.800 -2.263 0.2750 2.813 5.350
Max: 5.350
Min: -4.800
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 278.9
Min: 4.709e-15

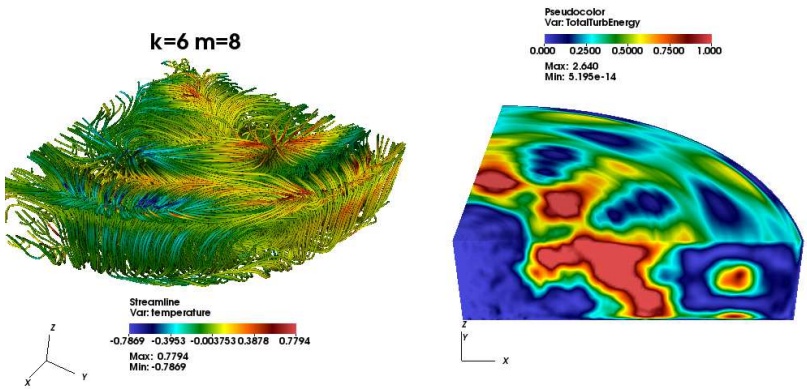

k=1 m=8
Streamline
Var: temperature
-0.5455 -0.2700 0.005538 0.2811 0.5566
Max: 0.5566
Min: -0.5455
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 7.900
Min: 6.411e-09

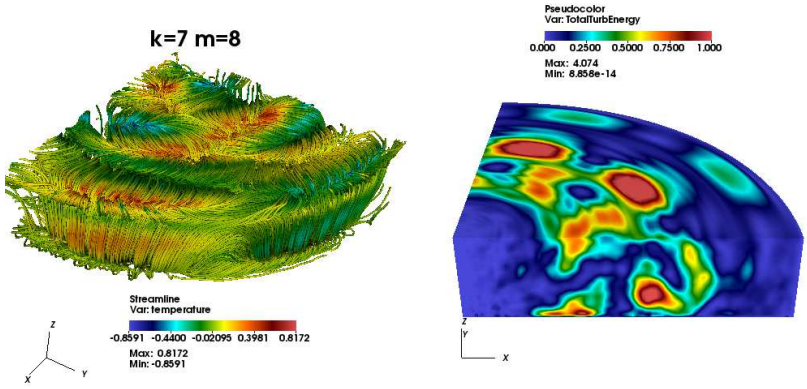

k=2 m=8
Streamline
Var: temperature
-0.8974 -0.4416 0.01426 0.4701 0.9259
Max: 0.9259
Min: -0.8974
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 10.03
Min: 2.504e-10

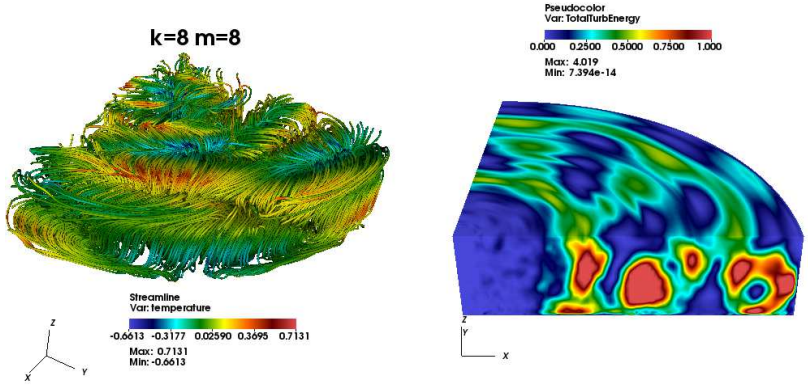

k=3 m=8
Streamline
Var: temperature
-0.6109 -0.3179 -0.02486 0.2682 0.5612
Max: 0.5612
Min: -0.6109
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.877
Min: 3.037e-12

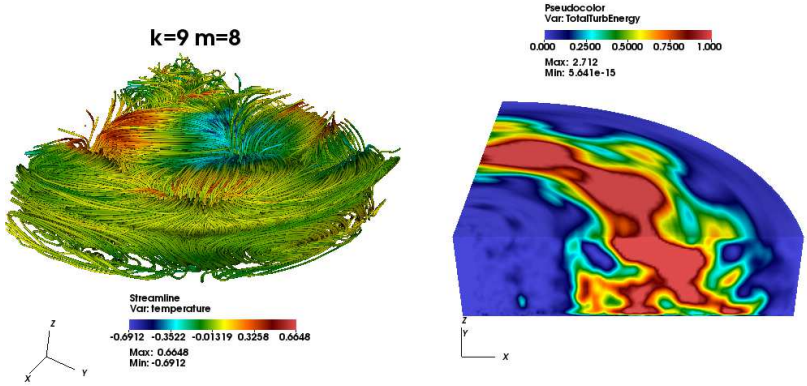

k=4 m=8
Streamline
Var: temperature
-0.7203 -0.3509 0.01843 0.3878 0.7572
Max: 0.7572
Min: -0.7203
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.416
Min: 9.671e-13

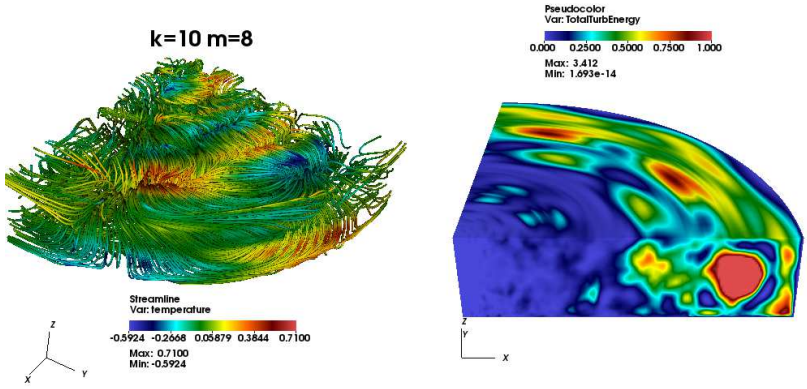

k=5 m=8
Streamline
Var: temperature
-0.8023 -0.4275 -0.05267 0.3221 0.6970
Max: 0.6970
Min: -0.8023
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.215
Min: 2.854e-13

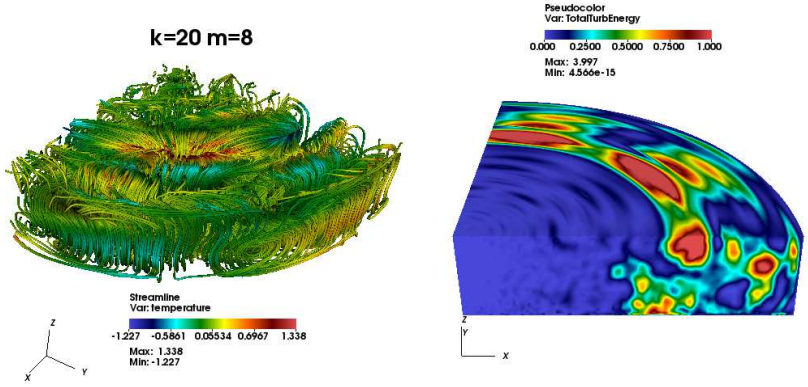

k=6 m=8
Streamline
Var: temperature
-0.7869 -0.3953 -0.003753 0.3878 0.7794
Max: 0.7794
Min: -0.7869
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.640
Min: 5.195e-14

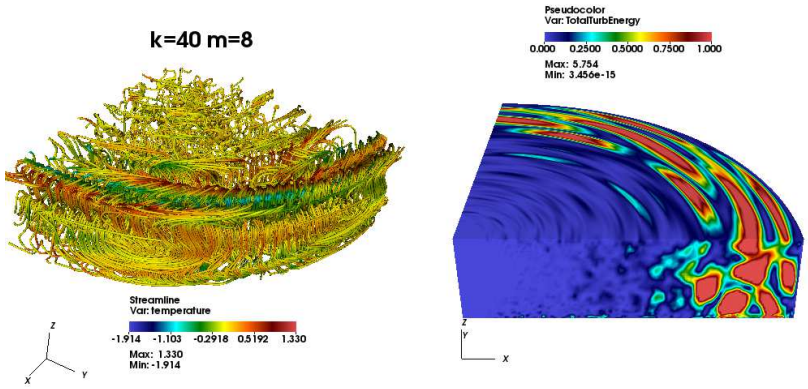

k=7 m=8
Streamline
Var: temperature
-0.8591 -0.4400 -0.02095 0.3981 0.8172
Max: 0.8172
Min: -0.8591
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.074
Min: 8.858e-14

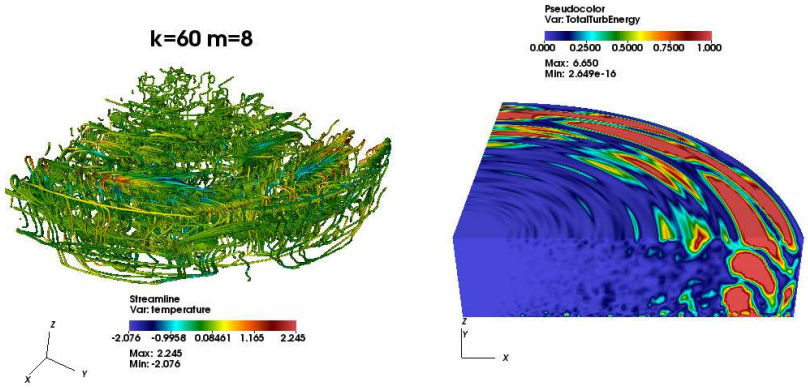

k=8 m=8
Streamline
Var: temperature
-0.6613 -0.3177 0.02590 0.3695 0.7131
Max: 0.7131
Min: -0.6613
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.019
Min: 7.394e-14

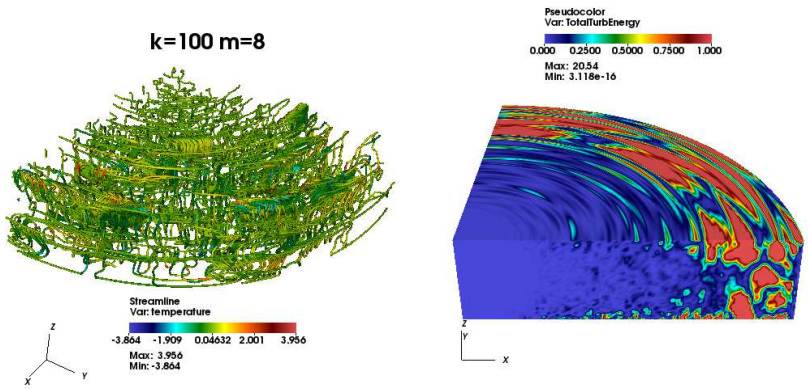

k=9 m=8
Streamline
Var: temperature
-0.6912 -0.3522 -0.01319 0.3258 0.6648
Max: 0.6648
Min: -0.6912
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.712
Min: 5.641e-15

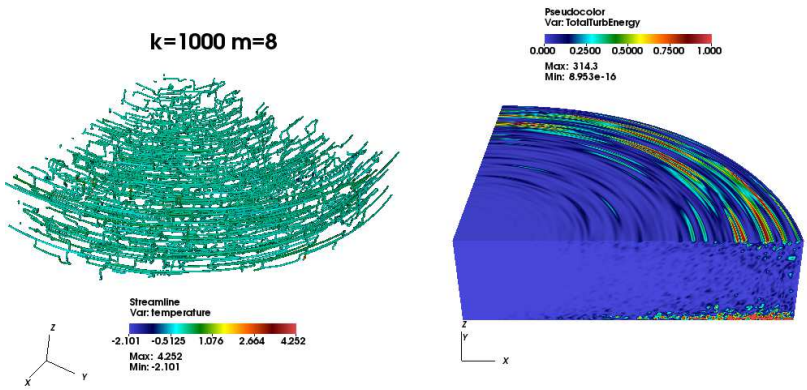

k=10 m=8
Streamline
Var: temperature
-0.5924 -0.2668 0.05879 0.3844 0.7100
Max: 0.7100
Min: -0.5924
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.412
Min: 1.693e-14

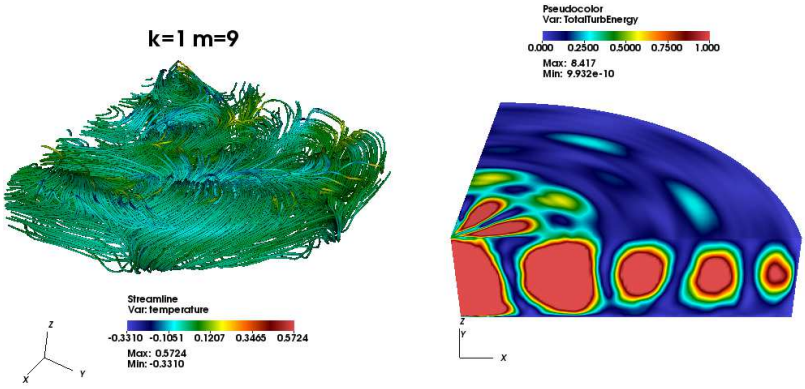

k=20 m=8
Streamline
Var: temperature
-1.227 -0.5861 0.05534 0.6967 1.338
Max: 1.338
Min: -1.227
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.997
Min: 4.566e-15

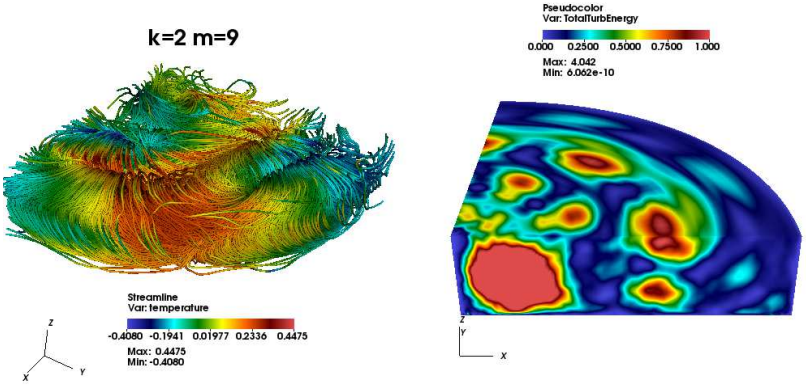

k=40 m=8
Streamline
Var: temperature
-1.914 -1.103 -0.2918 0.5192 1.330
Max: 1.330
Min: -1.914
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.754
Min: 3.456e-15

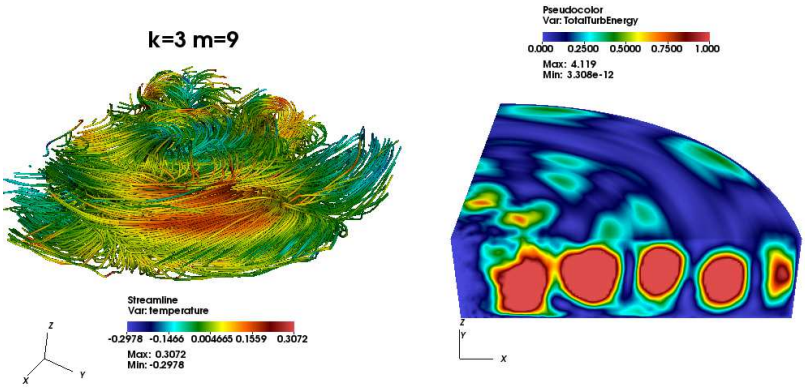

k=60 m=8
Streamline
Var: temperature
-2.076 -0.9958 0.08461 1.165 2.245
Max: 2.245
Min: -2.076
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 6.650
Min: 2.649e-16

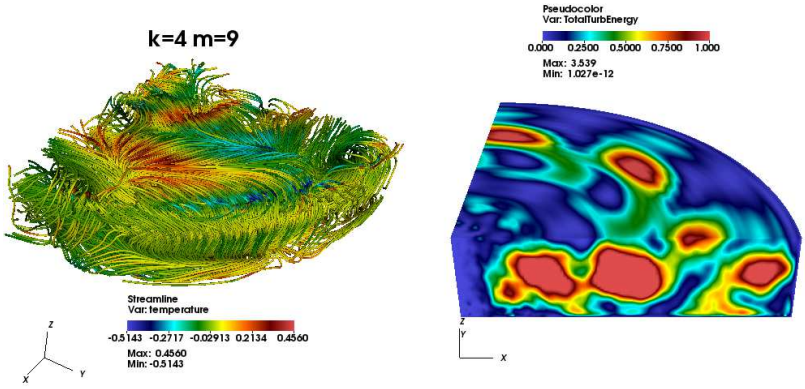

k=100 m=8
Streamline
Var: temperature
-3.864 -1.909 0.04632 2.001 3.956
Max: 3.956
Min: -3.864
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 20.54
Min: 3.118e-16

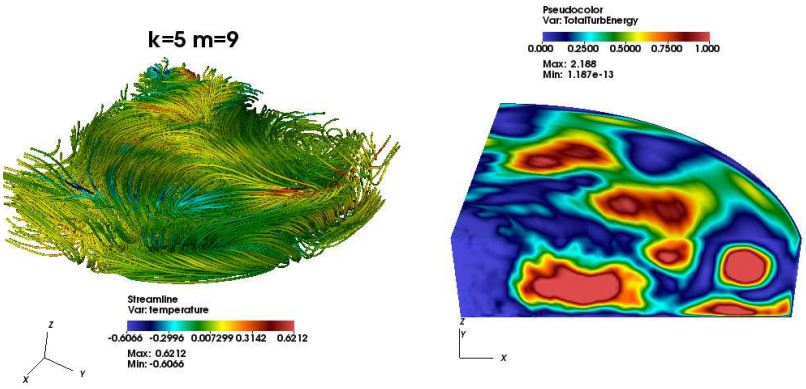

k=1000 m=8
Streamline
Var: temperature
-2.101 -0.5125 1.076 2.664 4.252
Max: 4.252
Min: -2.101
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 314.3
Min: 8.953e-16

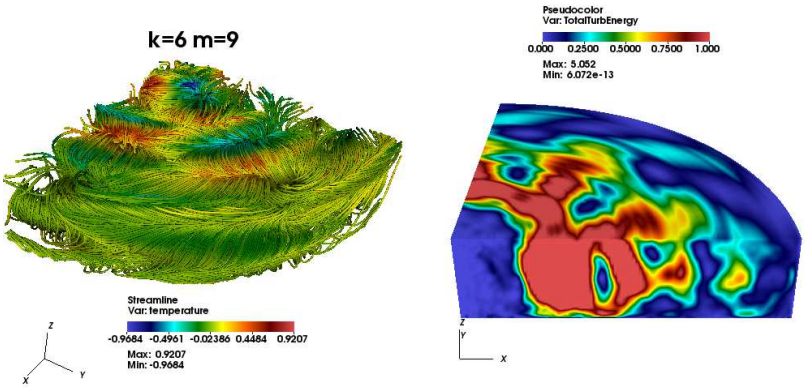

k=1 m=9
Streamline
Var: temperature
-0.3310 -0.1051 0.1207 0.3465 0.5724
Max: 0.5724
Min: -0.3310
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 8.417
Min: 9.932e-10

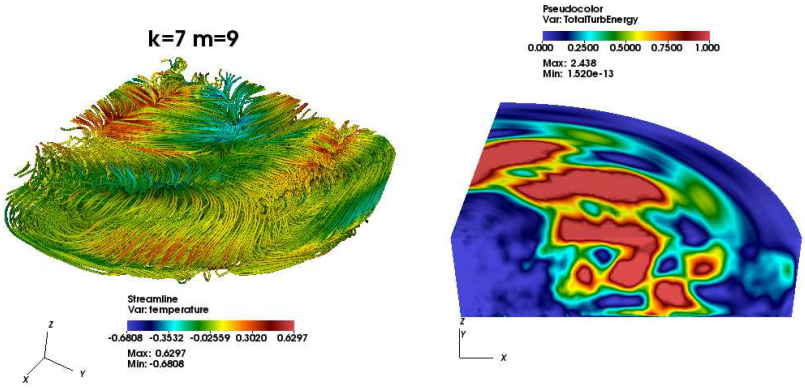

k=2 m=9
Streamline
Var: temperature
-0.4080 -0.1941 0.01977 0.2336 0.4475
Max: 0.4475
Min: -0.4080
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.042
Min: 6.062e-10

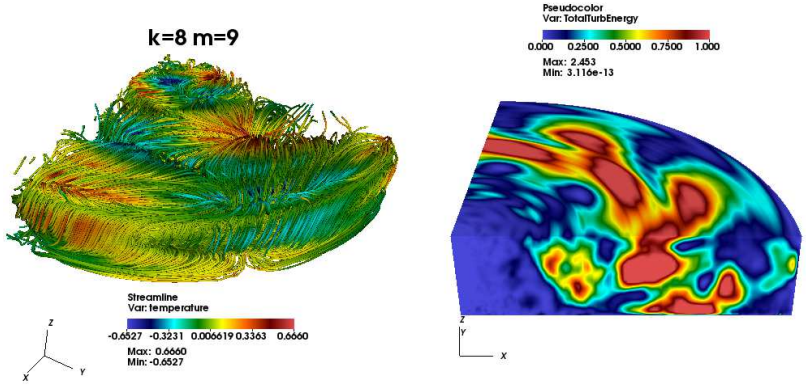

k=3 m=9
Streamline
Var: temperature
-0.2978 -0.1466 0.004665 0.1559 0.3072
Max: 0.3072
Min: -0.2978
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.119
Min: 3.308e-12

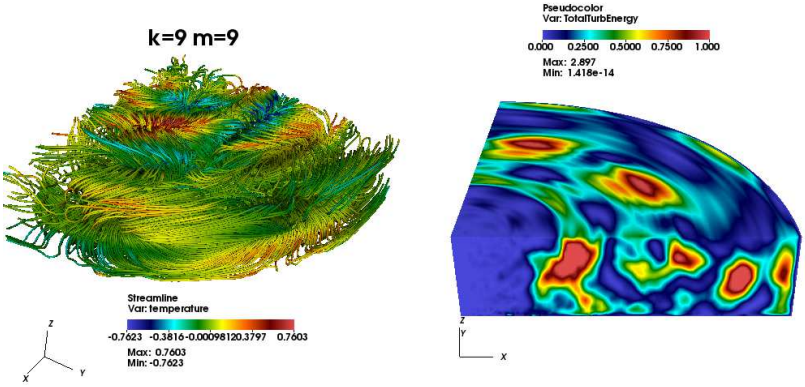

k=4 m=9
Streamline
Var: temperature
-0.5143 -0.2717 -0.02913 0.2134 0.4560
Max: 0.4560
Min: -0.5143
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.539
Min: 1.027e-12

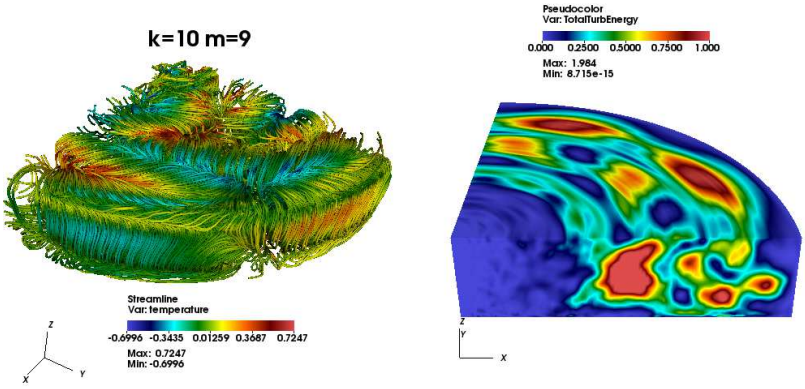

k=5 m=9
Streamline
Var: temperature
-0.6066 -0.2996 0.007299 0.3142 0.6212
Max: 0.6212
Min: -0.6066
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.188
Min: 1.187e-13

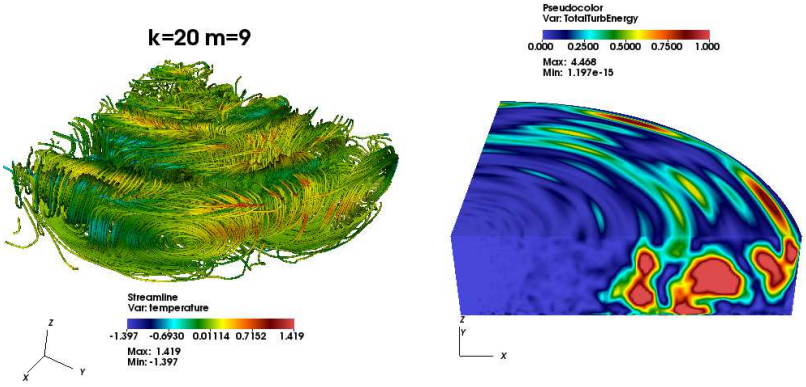

k=6 m=9
Streamline
Var: temperature
-0.9684 -0.4961 -0.02386 0.4484 0.9207
Max: 0.9207
Min: -0.9684
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.052
Min: 6.072e-13

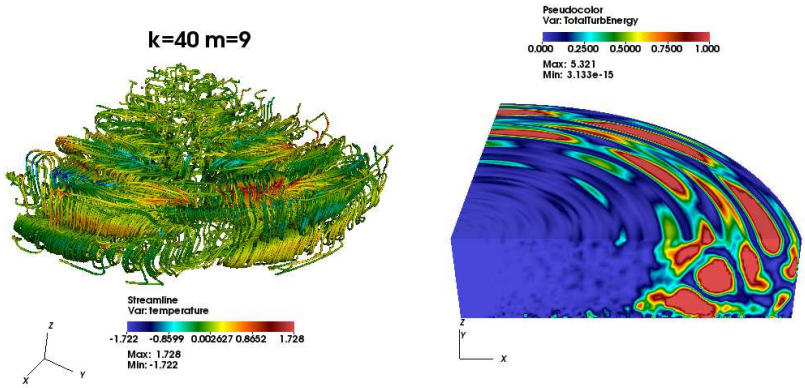

k=7 m=9
Streamline
Var: temperature
-0.6808 -0.3532 -0.02559 0.3020 0.6297
Max: 0.6297
Min: -0.6808
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.438
Min: 1.520e-13

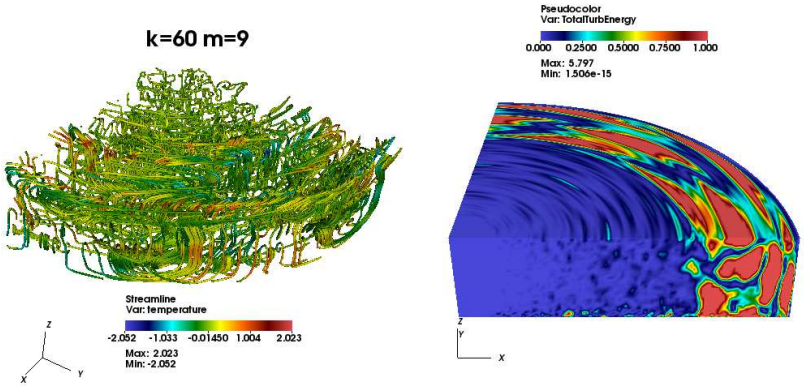

k=8 m=9
Streamline
Var: temperature
-0.6527 -0.3231 0.006619 0.3363 0.6660
Max: 0.6660
Min: -0.6527
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.453
Min: 3.116e-13

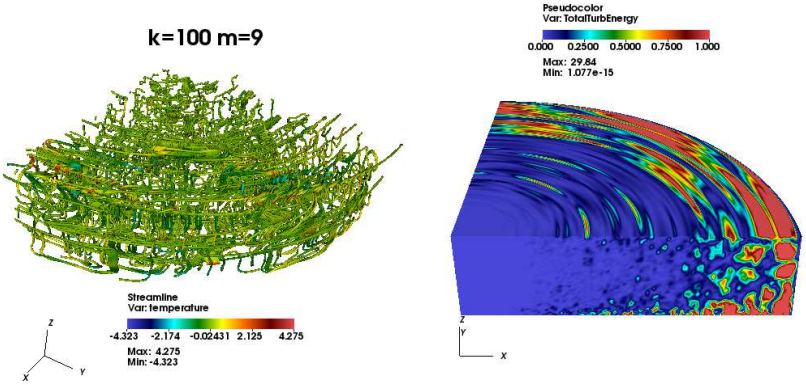

k=9 m=9
Streamline
Var: temperature
-0.7623 -0.3816 -0.0009812 0.3797 0.7603
Max: 0.7603
Min: -0.7623
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.897
Min: 1.418e-14

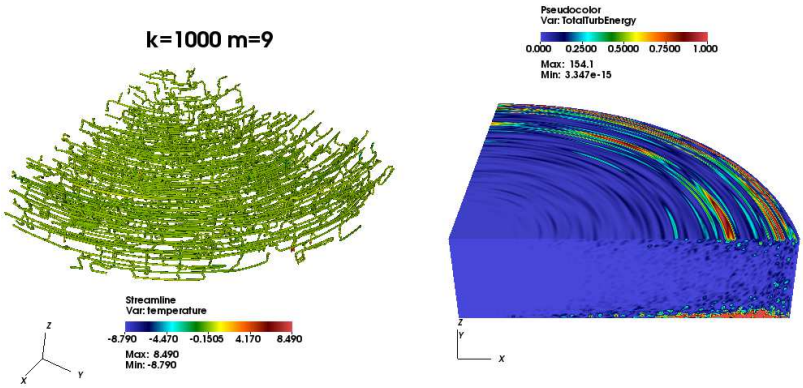

k=10 m=9
Streamline
Var: temperature
-0.6996 -0.3435 0.01259 0.3687 0.7247
Max: 0.7247
Min: -0.6996
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 1.984
Min: 8.715e-15

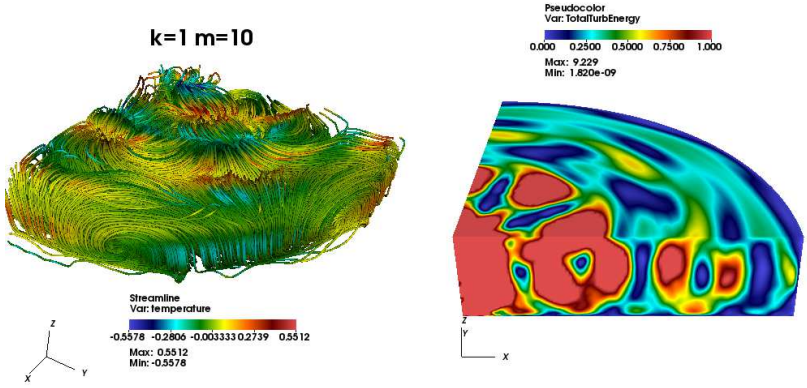

k=20 m=9
Streamline
Var: temperature
-1.397 -0.6930 0.01114 0.7152 1.419
Max: 1.419
Min: -1.397
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.468
Min: 1.197e-15

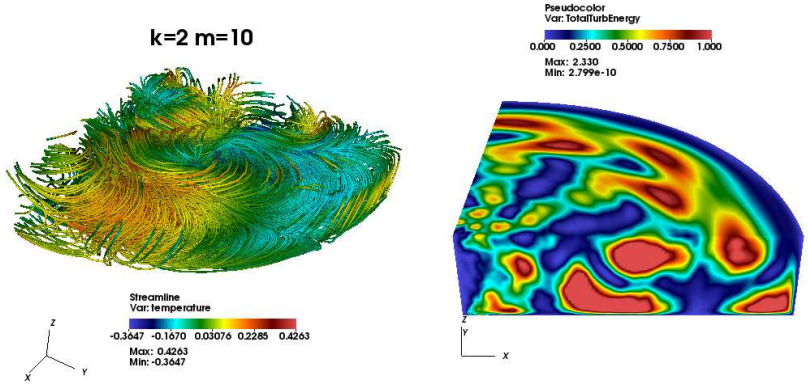

k=40 m=9
Streamline
Var: temperature
-1.722 -0.8599 0.002627 0.8652 1.728
Max: 1.728
Min: -1.722
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.321
Min: 3.133e-15

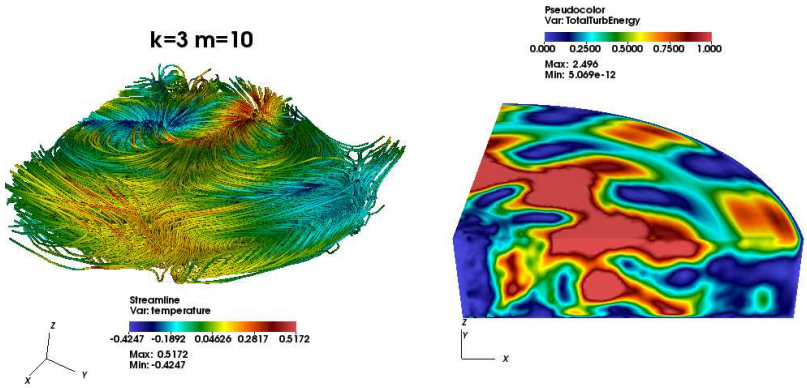

k=60 m=9
Streamline
Var: temperature
-2.052 -1.033 -0.01450 1.004 2.023
Max: 2.023
Min: -2.052
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 5.797
Min: 1.506e-15

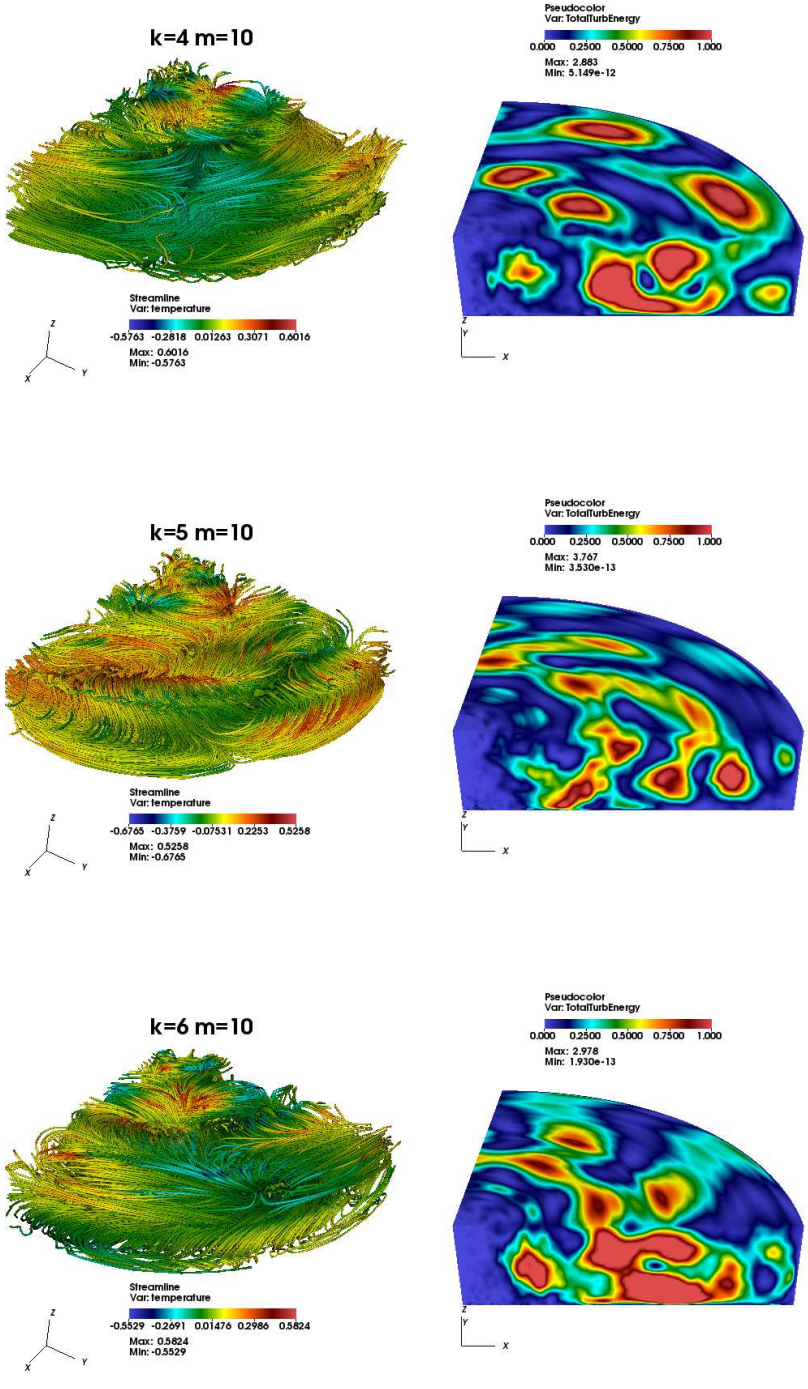

k=100 m=9
Streamline
Var: temperature
-4.323 -2.174 -0.02431 2.125 4.275
Max: 4.275
Min: -4.323
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 29.84
Min: 1.077e-15

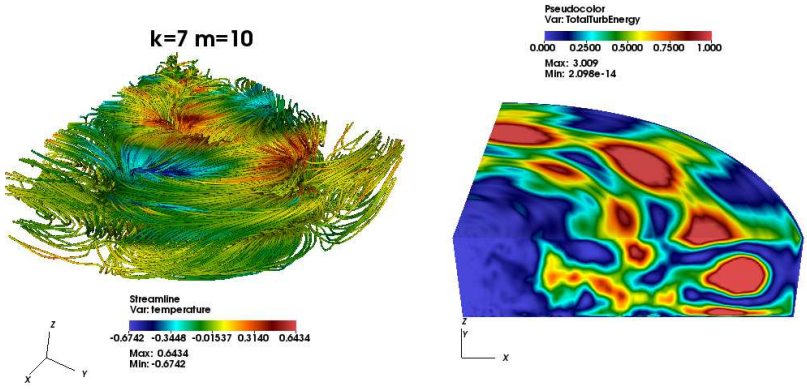

k=1000 m=9
Streamline
Var: temperature
-8.790 -4.470 -0.1505 4.170 8.490
Max: 8.490
Min: -8.790
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 154.1
Min: 3.347e-15

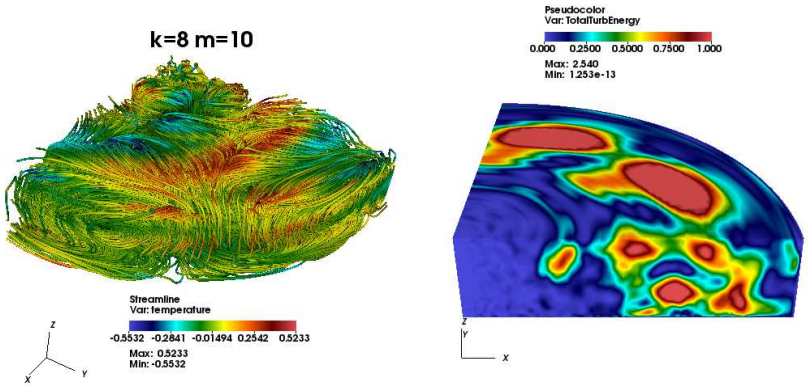

k=1 m=10
Streamline
Var: temperature
-0.5578 -0.2806 -0.003333 0.2739 0.5512
Max: 0.5512
Min: -0.5578
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 9.229
Min: 1.820e-09

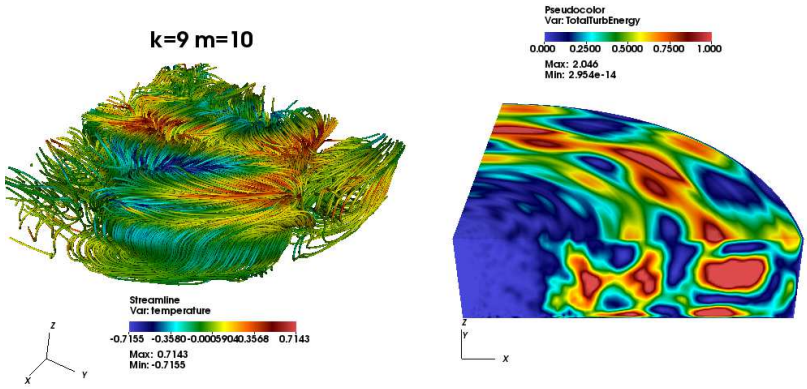

k=2 m=10
Streamline
Var: temperature
-0.3647 -0.1670 0.03076 0.2285 0.4263
Max: 0.4263
Min: -0.3647
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.330
Min: 2.799e-10

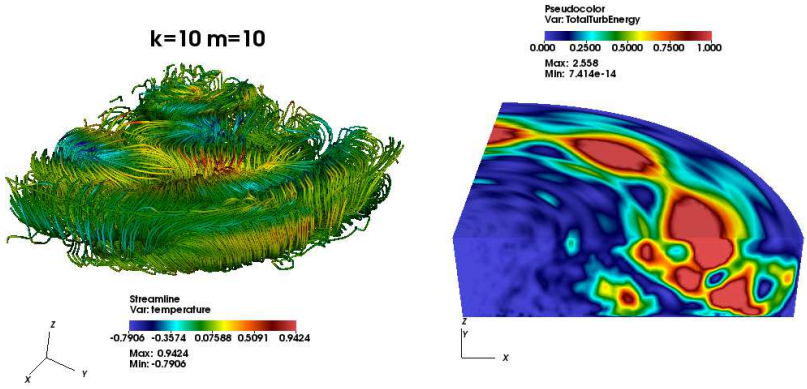

k=3 m=10
Streamline
Var: temperature
-0.4247 -0.1892 0.04626 0.2817 0.5172
Max: 0.5172
Min: -0.4247
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.496
Min: 5.069e-12

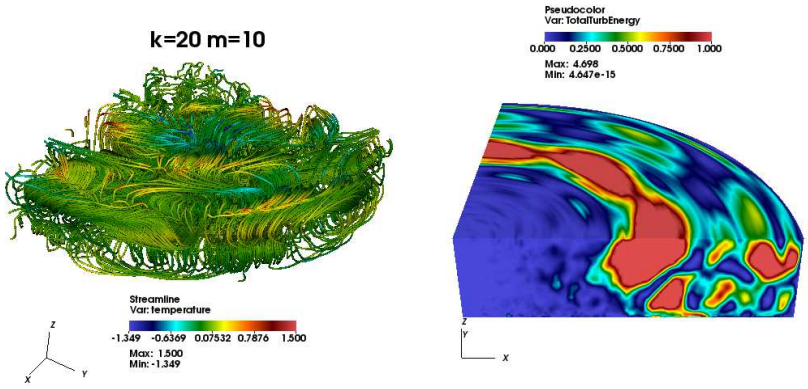

k=4 m=10
Streamline
Var: temperature
-0.5763 -0.2818 0.01263 0.3071 0.6016
Max: 0.6016
Min: -0.5763
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.883
Min: 5.149e-12
k=5 m=10
Streamline
Var: temperature
-0.6765 -0.3759 -0.07531 0.2253 0.5258
Max: 0.5258
Min: -0.6765
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.767
Min: 3.530e-13
k=6 m=10
Streamline
Var: temperature
-0.5529 -0.2691 0.01476 0.2986 0.5824
Max: 0.5824
Min: -0.5529
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.978
Min: 1.930e-13

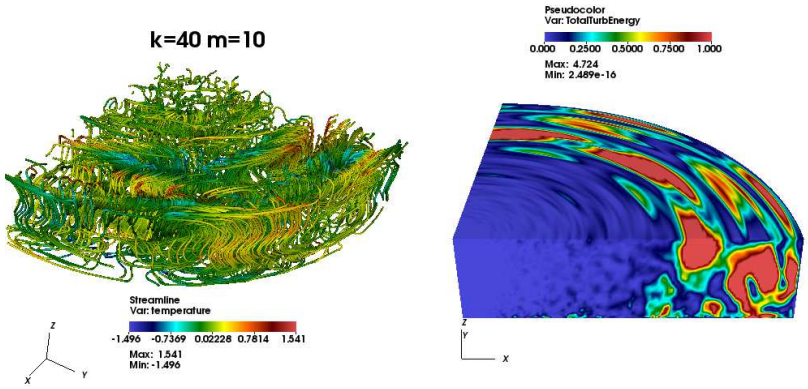

k=7 m=10
Streamline
Var: temperature
-0.6742 -0.3448 -0.01537 0.3140 0.6434
Max: 0.6434
Min: -0.6742
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 3.009
Min: 2.098e-14

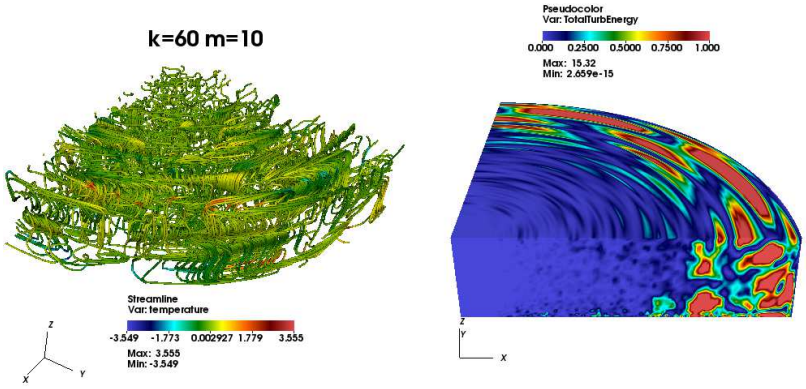

k=8 m=10
Streamline
Var: temperature
-0.5532 -0.2841 -0.01494 0.2542 0.5233
Max: 0.5233
Min: -0.5532
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.540
Min: 1.253e-13

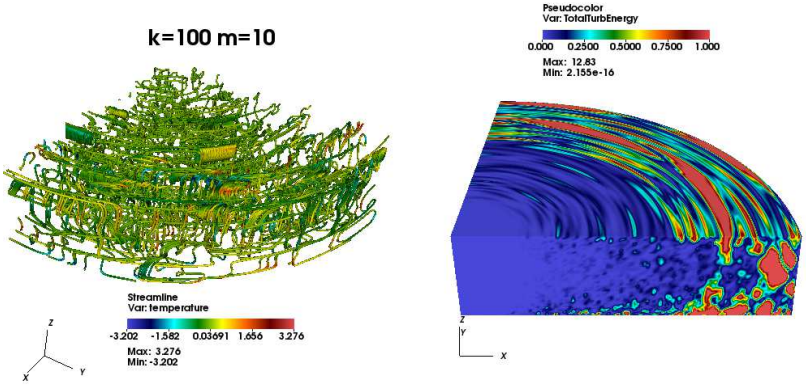

k=9 m=10
Streamline
Var: temperature
-0.7155 -0.3580 -0.0005904 0.3568 0.7143
Max: 0.7143
Min: -0.7155
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.046
Min: 2.954e-14

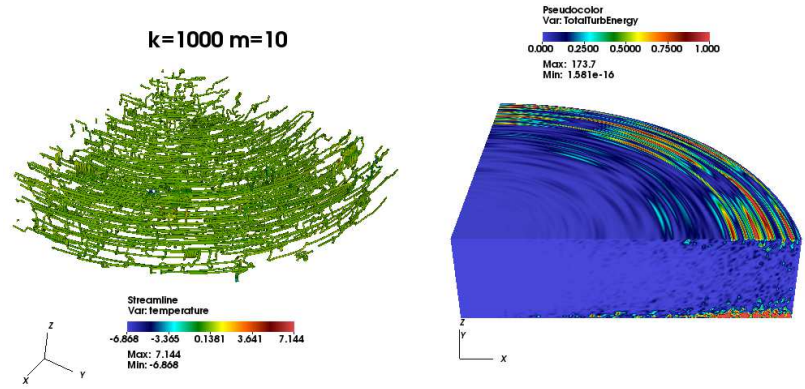

k=10 m=10
Streamline
Var: temperature
-0.7906 -0.3574 0.07588 0.5091 0.9424
Max: 0.9424
Min: -0.7906
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 2.558
Min: 7.414e-14


k=20 m=10
Streamline
Var: temperature
-1.349 -0.6369 0.07532 0.7876 1.500
Max: 1.500
Min: -1.349
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.698
Min: 4.647e-15


k=40 m=10
Streamline
Var: temperature
-1.496 -0.7369 0.02228 0.7814 1.541
Max: 1.541
Min: -1.496
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 4.724
Min: 2.489e-16


k=60 m=10
Streamline
Var: temperature
-3.549 -1.773 0.002927 1.779 3.555
Max: 3.555
Min: -3.549
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 15.32
Min: 2.659e-15


k=100 m=10
Streamline
Var: temperature
-3.202 -1.582 0.03691 1.656 3.276
Max: 3.276
Min: -3.202
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 12.83
Min: 2.155e-16


k=1000 m=10
Streamline
Var: temperature
-6.868 -3.365 0.1381 3.641 7.144
Max: 7.144
Min: -6.868
Pseudocolor
Var: TotalTurbEnergy
0.000 0.2500 0.5000 0.7500 1.000
Max: 173.7
Min: 1.581e-16

APPENDIX B

POST PROCESSING CODE

```
c-----------------------------------------------------------------------
c SUBROUTINES DEVELOPED BY PHIL SAKIEVICH
c FOR RAYLEIGH-BENARD CONVECTION ANALYSIS
c
c-----------------------------------------------------------------------
c
      subroutine ps_Test
c     TEST that the file compiles correctly
c     with the nek make routines
       write(6,*),"PHIL ROUTINES ARE WORKING"
      end subroutine ps_Test
c-----------------------------------------------------------------------
      subroutine ps_GridSpacing(gMin,gMax,gMean)
c
c    This routine finds and returns the global min max and mean grid spacing
c
      include 'SIZE'
      include 'TOTAL'
      real gMin(ndim), gMax(ndim), gMean(ndim)
c
      real lMin(ndim), lMax(ndim), lMean(ndim)
      real dX(lx1-1),dY(lx1-1),dZ(lx1-1)
      integer ncount,lcount;

      lMin=10000.;
      lMax=0;
      lMean=0;
      lcount=0;

c     find the mean for each processor
      do i=1,nelv
          do j=2,lx1
c             x spacing
              dX(j-1)=sqrt((xm1(j,1,1,i)-xm1(j-1,1,1,i))**2+
     $                     (ym1(j,1,1,i)-ym1(j-1,1,1,i))**2+
     $                     (zm1(j,1,1,i)-zm1(j-1,1,1,i))**2)
              if(dX(j-1).gt.lMax(1))lMax(1)=dX(j-1)
              if(dX(j-1).lt.lMin(1))lMin(1)=dX(j-1)
              lMean(1)=lMean(1)+dX(j-1)
c             y spacing
              dY(j-1)=sqrt((xm1(1,j,1,i)-xm1(1,j-1,1,i))**2+
     $                     (ym1(1,j,1,i)-ym1(1,j-1,1,i))**2+
     $                     (zm1(1,j,1,i)-zm1(1,j-1,1,i))**2)
              if(dY(j-1).gt.lMax(2))lMax(2)=dY(j-1)
              if(dY(j-1).lt.lMin(2))lMin(2)=dY(j-1)
              lMean(2)=lMean(2)+dY(j-1)
              if(ndim.eq.3)then
c             z spacing
              dZ(j-1)=sqrt((xm1(1,1,j,i)-xm1(1,1,j-1,i))**2+
     $                     (ym1(1,1,j,i)-ym1(1,1,j-1,i))**2+
     $                     (zm1(1,1,j,i)-zm1(1,1,j-1,i))**2)
              if(dZ(j-1).gt.lMax(3))lMax(3)=dZ(j-1)
              if(dZ(j-1).lt.lMin(3))lMin(3)=dZ(j-1)
              lMean(3)=lMean(3)+dZ(j-1)
              endif
c             add to counter
              lcount=lcount+1
          enddo
       enddo
c
c     find global mean
      call gop(lMean,gMean,'+  ',nDim)
      call igop(lCount,ncount,'+  ',1)
      do i=1,ndim
         gMean(i)=gMean(i)/dble(nCount)
      enddo
c
c     find global min
      call gop(lMin,gMin,'m  ',nDim)
c
c     find global max
      call gop(lMax,gMax,'M  ',nDim)
```

```
      return
      end subroutine ps_GridSpacing
c-----------------------------------------------------------------------
      subroutine ps_T_Diss(psi)
c     this subroutine computes the thermal dissipation grad T:grad T at each
c     grid point
c
c
      include 'SIZE'
      include 'TOTAL'
c
      integer psi !which passive scalar to use to store variable
      real dTx(lx1,ly1,lz1,lelt),
     $     dTy(lx1,ly1,lz1,lelt),
     $     dTz(lx1,ly1,lz1,lelt)
c
      nt=nx1*ny1*nz1*nelt
c     compute gradients
      call gradm1(dTx,dTy,dTz,T(1,1,1,1,1))
c
c     zero out the passive scalar
      call rzero(T(1,1,1,1,psi),nt)
c
c
      call add2col2(T(1,1,1,1,psi),dTx,dTx,nt)
      call add2col2(T(1,1,1,1,psi),dTy,dTy,nt)
      call add2col2(T(1,1,1,1,psi),dTz,dTz,nt)
      return
      end subroutine ps_T_Diss
c-----------------------------------------------------------------------
      subroutine ps_KE_Diss(psi)
c     this subroutine computes (grad U+ grad U^T)^2 at each grid point
c     *note additional scaling will be required based on dimensional form
c     of equations and puts dissipation in passive scalar #psi
c
c
      include 'SIZE'
      include 'TOTAL'
      integer psi
c     derivatives of velocity field
      real ddux(lx1,ly1,lz1,lelt),
     $     dduy(lx1,ly1,lz1,lelt),
     $     dduz(lx1,ly1,lz1,lelt),
     $     ddvx(lx1,ly1,lz1,lelt),
     $     ddvy(lx1,ly1,lz1,lelt),
     $     ddvz(lx1,ly1,lz1,lelt),
     $     ddwx(lx1,ly1,lz1,lelt),
     $     ddwy(lx1,ly1,lz1,lelt),
     $     ddwz(lx1,ly1,lz1,lelt),
     $     work(lx1,ly1,lz1,lelt)
      integer nv
c
      nv=nx1*ny1*nz1*nelv
      nt=nx1*ny1*nz1*nelt
      call rzero(T(1,1,1,1,psi),nt) ! zero out epsilon
c
c    compute velocity gradients
      call gradm1(ddux,dduy,dduz,vx)
      call gradm1(ddvx,ddvy,ddvz,vy)
      call gradm1(ddwx,ddwy,ddwz,vz)
c    sum up terms that contribute to dissipation
      call add2col2(T(1,1,1,1,psi),ddux,ddux,nt) !ux^2
      call add2col2(T(1,1,1,1,psi),ddvy,ddvy,nt) !vy^2
      call add2col2(T(1,1,1,1,psi),ddwz,ddwz,nt) !wz^2
      call add2col2(T(1,1,1,1,psi),dduy,ddvx,nt) !u_y*v_x (12*21)
      call add2col2(T(1,1,1,1,psi),ddvz,ddwy,nt) !v_z*w_y (23*32)
      call add2col2(T(1,1,1,1,psi),ddwx,dduz,nt) !w_x*u_z (31*13)
c   terms above contribute twice
      call cmult(T(1,1,1,1,psi),2.0,nt)
c   add the rest of the terms                    !(ij)
      call add2col2(T(1,1,1,1,psi),dduy,dduy,nt) !(12)
      call add2col2(T(1,1,1,1,psi),dduz,dduz,nt) !(13)
      call add2col2(T(1,1,1,1,psi),ddvx,ddvx,nt) !(21)
```

```
      call add2col2(T(1,1,1,1,psi),ddvz,ddvz,nt) !(23)
      call add2col2(T(1,1,1,1,psi),ddwx,ddwx,nt) !(31)
      call add2col2(T(1,1,1,1,psi),ddwy,ddwy,nt) !(32)
c
c      call rzero(T(1,1,1,1,psi),nt) ! zero out epsilon
      return
      end subroutine ps_KE_Diss
c-----------------------------------------------------------------------
      subroutine ps_Dissipation(eps,n)
c     this subroutine computes (grad U+ grad U^T)^2 at each grid point
c     *note additional scaling will be required based on dimensional form
c     of equations
c
c     parameters: eps- real array for storing the dissipation
c                 n - integer for size of eps
c
      include 'SIZE'
      include 'TOTAL'
      integer n
      real eps(n)
c     derivatives of velocity field
      real dux(lx1,ly1,lz1,lelv),
     $     duy(lx1,ly1,lz1,lelv),
     $     duz(lx1,ly1,lz1,lelv),
     $     dvx(lx1,ly1,lz1,lelv),
     $     dvy(lx1,ly1,lz1,lelv),
     $     dvz(lx1,ly1,lz1,lelv),
     $     dwx(lx1,ly1,lz1,lelv),
     $     dwy(lx1,ly1,lz1,lelv),
     $     dwz(lx1,ly1,lz1,lelv),
     $     work(lx1,ly1,lz1,lelv)
      integer nv
c
      nv=nx1*ny1*nz1*nelv
      if(n.ne.nv)then
         write(6,*)"Error in divergence input size",n,nv
         return
      endif
      call rzero(eps,n) ! zero out epsilon
c
c    compute velocity gradients
      call gradm1(dux,duy,duz,vx,nv)
      call gradm1(dvx,dvy,dvz,vy,nv)
      call gradm1(dwx,dwy,dwz,vz,nv)
c    sum up terms that contribute to dissipation
      call add2col2(eps,dux,dux,nv) !ux^2
      call add2col2(eps,dvy,dvy,nv) !vy^2
      call add2col2(eps,dwz,dwz,nv) !wz^2
      call add2col2(eps,duy,dvx,nv)
      call add2col2(eps,dvz,dwy,nv)
      call add2col2(eps,dwx,duz,nv)
c   terms above contribute twice
      call cmult(eps,2.0,nv)
c   add the rest of the terms
      call add2col2(eps,duy,duy,nv)
      call add2col2(eps,duz,duz,nv)
      call add2col2(eps,dvx,dvx,nv)
      call add2col2(eps,dvz,dvz,nv)
      call add2col2(eps,dwx,dwx,nv)
      call add2col2(eps,dwy,dwy,nv)
c
      return
      end subroutine ps_Dissipation
c-----------------------------------------------------------------------
c
c
c-----------------------------------------------------------------------
      subroutine ps_hpts(prefix)
c    ******************************************
c    ***** MODIFIED VERSION OF HPTS IN REPO******
c    ******************************************
C
c     evaluate velocity, temperature, pressure and ps-scalars
```

```
c     for list of points (read from hpts.in) and dump results
c     into a file (hpts.out).
c     note: read/write on rank0 only
c
c     ASSUMING LHIS IS MAX NUMBER OF POINTS TO READ IN ON ONE PROCESSOR

      include 'SIZE'
      include 'TOTAL'

      parameter(nfldm=2*ldim+ldimt+1)


      common /c_hptsr/ pts      (ldim,lhis)
     $               , fieldout (nfldm,lhis)
     $               , dist     (lhis)
     $               , rst      (lhis*ldim)


      common /c_hptsi/ rcode(lhis),elid(lhis),proc(lhis)

      common /scrcg/  pm1 (lx1,ly1,lz1,lelv) ! mapped pressure
      common /outtmp/ wrk (lx1*ly1*lz1*lelt,nfldm)
      character*3    prefix

      logical iffind

      integer icalld,npoints,npts
      save    icalld,npoints,npts
      data    icalld  /0/
      data    npoints /0/

      save    inth_hpts

      nxyz  = nx1*ny1*nz1
      ntot  = nxyz*nelt
      nbuff = lhis      ! point to be read in on 1 proc.

      if(nio.eq.0) write(6,*) 'dump history points'

      if(icalld.eq.0) then
        npts  = lhis      ! number of points per proc
        call ps_hpts_in(pts,npts,npoints) !npoints is initially zero
        call intpts_setup(-1.0,inth_hpts) ! use default tolerance
      endif


      call prepost_map(0)  ! maps axisymm and pressure

      ! pack working array
      ! modified to dump out corrdinates as well
      nflds = ndim
      if(ifvo) then
        call copy(wrk(1,ndim+1),vx,ntot)
        call copy(wrk(1,ndim+2),vy,ntot)
        if(if3d) call copy(wrk(1,ndim+3),vz,ntot)
        nflds = ndim+ndim
      endif
      if(ifpo) then
        nflds = nflds + 1
        call copy(wrk(1,nflds),pm1,ntot)
      endif
      if(ifto) then
        nflds = nflds + 1
        call copy(wrk(1,nflds),t,ntot)
      endif
      do i = 1,ldimt
         if(ifpsco(i)) then
           nflds = nflds + 1
           call copy(wrk(1,nflds),T(1,1,1,1,i+1),ntot)
         endif
      enddo

      ! interpolate
```

```
      if(icalld.eq.0) then
        call findpts(inth_hpts,rcode,1,
     &                proc,1,
     &                elid,1,
     &                rst,ndim,
     &                dist,1,
     &                pts(1,1),ndim,
     &                pts(2,1),ndim,
     &                pts(3,1),ndim,npts)

        do i=1,npts
           ! check return code
           if(rcode(i).eq.1) then
             if (dist(i).gt.1e-12) then
                nfail = nfail + 1
                IF (NFAIL.LE.5) WRITE(6,'(a,1p4e15.7)')
     &      ' WARNING: point on boundary or outside the mesh xy[z]d^2:'
     &      ,(pts(k,i),k=1,ndim),dist(i)
             endif
           elseif(rcode(i).eq.2) then
             nfail = nfail + 1
             if (nfail.le.5) write(6,'(a,1p3e15.7)')
     &        ' WARNING: point not within mesh xy[z]: !',
     &         (pts(k,i),k=1,ndim)
           endif
        enddo
        icalld = 1
      endif
      if(nflds.ne.nfldm.and.nid.eq.0)write(6,*)"Error nflds ",nflds,
     $   nfldm
      ! evaluate input field at given points
      do ifld = ndim+1,nflds
         call findpts_eval(inth_hpts,fieldout(ifld,1),nfldm,
     &                     rcode,1,
     &                     proc,1,
     &                     elid,1,
     &                     rst,ndim,npts,
     &                     wrk(1,ifld))
      enddo
      !copy coordinates
      do i=1,ndim
        do ii=1,npts
          fieldout(i,ii)=pts(i,ii)
        enddo
      enddo

      ! write interpolation results to file
c      call ps_hpts_out(fieldout,nflds,nfldm,npoints,nbuff)
      call ps_hpts_out_fld(prefix,fieldout,nflds,nfldm,
     $                                 npoints,nbuff)

      call prepost_map(1)  ! maps back axisymm arrays

      if(nio.eq.0) write(6,*) 'done :: dump history points'

      return
      end
c-----------------------------------------------------------------------
      subroutine ps_buffer_in(buffer,npp,npoints,nbuf)

      include 'SIZE'
      include 'PARALLEL'

      common/hpts_to_elm/NELGH,NXH,NYH,NZH !sizes from hpts_fld
      real    buffer(ldim,nbuf)

      ierr = 0
c    read in the total number of history points from hpts.in
      if(nid.eq.0) then
        write(6,*) 'reading ps_hpts.in'
        open(50,file='ps_hpts.in',status='old',err=100)
        read(50,*,err=100) npoints,NELGH,NXH,NYH,NZH
        goto 101
```

```
 100    ierr = 1
 101    continue
      endif
c   check all processors for an error
      ierr=iglsum(ierr,1)
      if(ierr.gt.0) then
        write(6,*) 'Cannot open ps_hpts.in in
     $                    subroutine hpts()'
        call exitt
      endif
c     send total number of points to all processors and
c     check to see if there is enough memory allocated
      call bcast(npoints,isize)
      call bcast(nelgh,isize)
      call bcast(nxh,isize)
      call bcast(nyh,isize)
      call bcast(nzh,isize)
      if(npoints.gt.lhis*np) then
        if(nid.eq.0) write(6,*) 'ABORT: Too many pts to read in hpts()!'
        call exitt
      endif
      if(nid.eq.0) write(6,*) 'found ', npoints, ' points'

c     nbuf=2*lx1*ly1*lz1*lelt
      npass =  npoints/nbuf +1  !number of passes to cover all pts
      n0    =  mod(npoints,nbuf)!remainder
      if(n0.eq.0) then
         npass = npass-1
         n0    = nbuf
      endif

      len = wdsize*ndim*nbuf
c     put all processors except processor 0 into receive mode
      if (nid.gt.0.and.nid.lt.npass) msg_id=irecv(nid,buffer,len)
      call nekgsync
c     read in data with processor 0 from hts and send to other
c     processors
      npp=0
      if(nid.eq.0) then
        i1 = nbuf
        do ipass = 1,npass
           if(ipass.eq.npass) i1 = n0
           do i = 1,i1
              read(50,*) (buffer(j,i),j=1,ndim)
           enddo
           if(ipass.lt.npass)call csend(ipass,buffer,len,ipass,0)
        enddo
        close(50)
        npp = n0
c        open(50,file='hpts.out')!,status='new')
c        write(50,'(A)')
c     &       '# time  vx  vy  [vz]  pr  T  PS1  PS2  ...'
      elseif (nid.lt.npass)  then !processors receiving data
        call msgwait(msg_id)
        npp=nbuf
      endif

      return
      end
c-----------------------------------------------------------------------
      subroutine ps_hpts_in(pts,npts,npoints)
c                       npts=local count; npoints=total count

      include 'SIZE'
      include 'PARALLEL'

      parameter (lt2=2*lx1*ly1*lz1*lelt)
      common /scrns/ xyz(ldim,lt2)
      common /scruz/ mid(lt2)  ! Target proc id
      common/hpts_to_elm/NELGH,NXH,NYH,NZH !sizes from hpts_fld
      real    pts(ldim,npts)

c     I think that if this conditional is false the routine
```

```
c     puts all of the points on processor 0
      if (lt2.gt.npts) then

         call ps_buffer_in(xyz,npp,npoints,lt2) !lt2 is the size of buffer
         if(npoints.gt.np*npts) then
           if(nid.eq.0)write(6,*)'ABORT in hpts(): npoints > NP*lhis!!'
           if(nid.eq.0)write(6,*)'Change SIZE: ',np,npts,npoints
           call exitt
         endif
         if(npoints.ne.NELGH*NXH*NYH*NZH)then
           if(nid.eq.0)write(6,*)'HPNTS dosnt match the given dims'
     $                    ,npoints,NELGH*NXH*NYH*NZH
           call exitt
         endif

         npmax = (npoints/npts)
         if(mod(npoints,npts).eq.0) npmax=npmax+1

         if(nid.gt.0.and.npp.gt.0) then
          npts_b = lt2*(nid-1)                ! # pts  offset(w/o 0)
          nprc_b = npts_b/npts                ! # proc offset(w/o 0)

          istart = mod(npts_b,npts)           ! istart-->npts pts left
          ip     = nprc_b + 1                 ! PID offset
          icount = istart                     ! point offset
         elseif(nid.eq.0) then
          npts0   = mod1(npoints,lt2)         ! Node 0 pts
          npts_b  = npoints - npts0           ! # pts before Node 0
          nprc_b  = npts_b/npts

          istart  = mod(npts_b,npts)
          ip      = nprc_b + 1
          icount  = istart
         endif

         do i =1,npp
            icount = icount + 1
            if(ip.gt.npmax) ip = 0
            mid(i) = ip
            if (icount.eq.npts) then
               ip     = ip+1
               icount = 0
            endif
         enddo

         call crystal_tuple_transfer
     &       (cr_h,npp,lt2,mid,1,pts,0,xyz,ldim,1)

         call copy(pts,xyz,ldim*npp)
      else
         call ps_buffer_in(pts,npp,npoints,npts)
      endif
      npts = npp


      return
      end
c-----------------------------------------------------------------------
      subroutine ps_hpts_out(fieldout,nflds,nfldm,npoints,nbuff)
c     ******************************************
c     *** MODIFIED VERSION OF HPTS_OUT IN REPO****
c     ******************************************

      include 'SIZE'
      include 'TOTAL'
      common/hpts_to_elm/NELGH,NXH,NYH,NZH !sizes from hpts_fld
      real buf(nfldm,nbuff),fieldout(nfldm,nbuff)
      character*80 filename
      character*1 excode(30)
      integer iFileNum
      save iFileNum
      data iFileNum /0/
```

```
      len = wdsize*nfldm*nbuff

      npass = npoints/nbuff + 1
      il = mod(npoints,nbuff)
      if(il.eq.0) then
         il = nbuff
         npass = npass-1
      endif
      !setup the header
      call BLANK(EXCODE,30)
         IF(IFXYO) then
            EXCODE(1)='X'
            EXCODE(2)=' '
            EXCODE(3)='Y'
            EXCODE(4)=' '
            i = 5
            IF(IF3D) THEN
              EXCODE(i)  ='Z'
              EXCODE(i+1)=' '
              i = i + 2
            ENDIF
         ENDIF
         IF(IFVO) then
            EXCODE(i)  ='U'
            EXCODE(i+1)=' '
            i = i + 2
         ENDIF
         IF(IFPO) THEN
           EXCODE(i)='P'
           EXCODE(i+1)=' '
           i = i + 2
         ENDIF
         IF(IFTO) THEN
           EXCODE(i)='T '
           EXCODE(i+1)=' '
           i = i + 1
         ENDIF
         do iip=1,ldimt1
            if (ifpsco(iip)) then
              write(excode(iip+I)  ,'(i1)') iip
              write(excode(iip+I+1),'(a1)') ' '
              i = i + 1
            endif
         enddo


        if(nid.eq.0)then
          write(filename,"('udfpnts.fld',I2.2)")iFileNum+1
          open(unit=50,file=filename)!,status='new')
          WRITE(50,'(4i4,1pe14.7,I5,1X,30A1,1X,A12)')
     $      NELGH,NXH,NYH,NZH,TIME,iFileNum,(EXCODE(I),I=1,30),
     $           'NELT,NX,NY,N'
          CDRROR=0.0
          WRITE(50,'(6G11.4)')(CDRROR,I=1,NELGH)    ! dummy

        endif
        call nekgsync
      do ipass = 1,npass

        if(ipass.lt.npass) then
          if(nid.eq.0) then
            call crecv(ipass,buf,len)
            do ip = 1,nbuff
              write(50,'(1p20E14.6)'),
c     &          (pts(i,ip), i=1,ndim),
     &          (buf(i,ip), i=1,nflds)
            enddo
          elseif(nid.eq.ipass) then
            call csend(ipass,fieldout,len,0,nid)
          endif

        else  !ipass.eq.npass
```

```
          if(nid.eq.0) then
            do ip = 1,il
              write(50,'(1p20E14.6)'),
c     &          (pts(i,ip), i=1,ndim),
     &          (fieldout(i,ip), i=1,nflds)
            enddo
          endif

        endif
      enddo
      call nekgsync
      if(nid.eq.0)then
        close(unit=50)
        iFileNum=iFileNum+1
      endif
      return
      end
c-----------------------------------------------------------------------
      subroutine ps_hpts_out_fld(prefix,fieldout,nflds,
     $                           nfldm,npoints,nbuff)

c     output .fld file

      include 'SIZE'
      include 'TOTAL'
      include 'RESTART'
C
C     Work arrays and temporary arrays
C
      common /scrcg/ pm1 (lx1,ly1,lz1,lelv)
      common/hpts_to_elm/NELGH,NXH,NYH,NZH
c
c     note, this usage of CTMP1 will be less than elsewhere if NELT ~> 3.
      parameter (lxyz=lx1*ly1*lz1)
      parameter (lpsc9=ldimt1+9)
c      parameter (lxyz=NXH*NYH*NZH)
c      common /cbuff1/ tbuf(lxyz,lpsc9)
      real*4         tbuf(NXH*NYH*NZH,lpsc9)

      real*4         test_pattern

      character*3    prefix
      real buf(nfldm,nbuff),fieldout(nfldm,nbuff)!from hpts_out
      character*1    fhdfle1(132)
      character*132   fhdfle
      equivalence   (fhdfle,fhdfle1)
      character*1    fldfile2(120)
      integer        fldfilei( 60)
      equivalence   (fldfilei,fldfile2)

      character*1 excode(30)
      character*10 frmat

      common /nopenf/ nopen(99)

      common /rdump/ ntdump
      data ndumps / 0 /

      logical ifxyo_s
      integer ncount,hxyz
      hxyz=NXH*NYH*NZH
      len = wdsize*nfldm*nbuff!from hpts_out
      npass = npoints/nbuff + 1
      il = mod(npoints,nbuff)
      if(il.eq.0) then
         il = nbuff
         npass = npass-1
      endif

c  Write to logfile that you're outputting data
      if(nio.eq.0) then
        WRITE(6,1001) istep,time
 1001   FORMAT(/,i9,1pe12.4,' Write checkpoint:')
```

```
      endif
      call nekgsync()

c  Check for file type
c  If filetype =6 then use multi-file-output
      p66 = abs(param(66))
c      p66=0
c      if (p66.eq.6) then
c         call mfo_outfld(prefix)
c         call nekgsync               ! avoid race condition w/ outfld
c         return
c      endif

      ifxyo_s = ifxyo              ! Save ifxyo

c  Check given prefix against the database of prefixes
      iprefix = i_find_prefix(prefix,99)

      ierr = 0
      if (nid.eq.0) then

c       Open new file for each dump on /cfs
        nopen(iprefix)=nopen(iprefix)+1

        if (prefix.eq.'   '.and.nopen(iprefix).eq.1) ifxyo = .true. ! 1st file

        if (prefix.eq.'rst'.and.max_rst.gt.0)
     $         nopen(iprefix) = mod1(nopen(iprefix),max_rst) ! restart

        call file2(nopen(iprefix),prefix)
c     if file type is 0 or negative then open using statement for ASCII
        if (p66.lt.1.0) then
           open(unit=24,file=fldfle,form='formatted',status='unknown')
        else
c     open binary file
           call  izero    (fldfilei,33)
           len1 = ltrunc   (fldfle,131)
           call chcopy    (fldfile2,fldfle,len1)
           call byte_open (fldfile2,ierr)
c          write header as character string
           call blank(fhdfle,132)
        endif
      endif
c    broadcast if you are dumping the grid
      call bcast(ifxyo,lsize)
c    check to see if there was an error when byte_open was called
      if(p66.ge.1.0)
     $   call err_chk(ierr,'Error opening file in outfld. Abort. $')

C     Figure out what goes in EXCODE (header)
      CALL BLANK(EXCODE,30)
      NDUMPS=NDUMPS+1
      i=1
      if (mod(p66,1.0).eq.0.0) then !old header format
         IF(IFXYO) then
            EXCODE(1)='X'
            EXCODE(2)=' '
            EXCODE(3)='Y'
            EXCODE(4)=' '
            i = 5
            IF(IF3D) THEN
              EXCODE(i)  ='Z'
              EXCODE(i+1)=' '
              i = i + 2
            ENDIF
         ENDIF
         IF(IFVO) then
            EXCODE(i)  ='U'
            EXCODE(i+1)=' '
            i = i + 2
         ENDIF
         IF(IFPO) THEN
           EXCODE(i)='P'
```

```
            EXCODE(i+1)=' '
            i = i + 2
          ENDIF
          IF(IFTO) THEN
            EXCODE(i)='T '
            EXCODE(i+1)=' '
            i = i + 1
          ENDIF
          do iip=1,ldimt1
             if (ifpsco(iip)) then
               write(excode(iip+I)  ,'(i1)') iip
               write(excode(iip+I+1),'(a1)') ' '
               i = i + 1
             endif
          enddo
       else
          !new header format
          IF (IFXYO) THEN !dumping grid information
             EXCODE(i)='X'
             i = i + 1
          ENDIF
          IF (IFVO) THEN !dumping velocity information
             EXCODE(i)='U'
             i = i + 1
          ENDIF
          IF (IFPO) THEN !dumping pressure information
             EXCODE(i)='P'
             i = i + 1
          ENDIF
          IF (IFTO) THEN !dumping Temperature information
             EXCODE(i)='T'
             i = i + 1
          ENDIF
          IF (LDIMT.GT.1) THEN !dumping passive scalar information
             NPSCALO = 0
             do k = 1,ldimt-1
               if(ifpsco(k)) NPSCALO = NPSCALO + 1
             enddo
             IF (NPSCALO.GT.0) THEN
                EXCODE(i) = 'S'
                WRITE(EXCODE(i+1),'(I1)') NPSCALO/10
                WRITE(EXCODE(i+2),'(I1)') NPSCALO-(NPSCALO/10)*10
             ENDIF
          ENDIF
       endif

c^^^^^^^^^^^^^^^^^^^^^^^No changes necessary^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c !!!!!!!!!!!!!!!!!!!!!Begin Changes!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
c
c      Begining from hpts_out set up amount to pass
c       npass = npoints/nbuff + 1
c       il = mod(npoints,nbuff)
c       if(il.eq.0) then
c          il = nbuff
c          npass = npass-1
c       endif
C      Dump header based on phil files
       ierr = 0
       if (nid.eq.0) call ps_dump_header(excode,p66,ierr)
       call err_chk(ierr,'Error dumping header in outfld. Abort. $')

c    Get number of fields to write to file (xyzuvwpTt1 etc)
       call get_id(id)

       ierr = 0
       ncount=1
c      Dump out hpts in terms of elements
       call nekgsync
       do ipass = 1,npass

         if(ipass.lt.npass) then
           if(nid.eq.0) then
             call crecv(ipass,buf,len)
```

```
            do ip = 1,nbuff
              do i=1,nflds
                tbuf(ncount,i)=buf(i,ip)
              enddo
              ncount=ncount+1
c              if(ncount-1.eq.lxyz.or.ip.eq.nbuff)then
              if(ncount-1.eq.hxyz)then
                 call ps_out_buff(id,p66,tbuf,ierr)
                 ncount=1
              endif
            enddo
          elseif(nid.eq.ipass) then
            call csend(ipass,fieldout,len,0,nid)
          endif

        else  !ipass.eq.npass

          if(nid.eq.0) then
            do ip = 1,il
              do i=1,nflds
               tbuf(ncount,i)=fieldout(i,ip)
              enddo
              ncount=ncount+1
c              if(ncount-1.eq.lxyz.or.ip.eq.il)then
              if(ncount-1.eq.hxyz)then
c                 call ps_out_buff(id,p66,ncount-1,ierr)
                 call ps_out_buff(id,p66,tbuf,ierr)
                 ncount=1
              endif
            enddo
          endif

        endif
      enddo
      call nekgsync

      call err_chk(ierr,'Error writing file in outfld. Abort. $')

      ifxyo = ifxyo_s           ! restore ifxyo

      if (nid.eq.0) call close_fld(p66,ierr)
      call err_chk(ierr,'Error closing file in outfld. Abort. $')

      return
      end
c-----------------------------------------------------------------------
      subroutine ps_dump_header(excodein,p66,ierr)

      include 'SIZE'
      include 'TOTAL'
      common/hpts_to_elm/NELGH,NXH,NYH,NZH

      character*30  excodein

      character*30 excode
      character*1  excode1(30)
      equivalence (excode,excode1)

      real*4         test_pattern

      character*1 fhdfle1(132)
      character*132 fhdfle
      equivalence (fhdfle,fhdfle1)

      write(excode,'(A30)') excodein

      ikstep = istep
      do ik=1,10
         if (ikstep.gt.9999) ikstep = ikstep/10
      enddo

      call blank(fhdfle,132)
```

```
c       write(6,111)               !       print on screen
c     $     nelgt,nx1,ny1,nz1,time,istep,excode
c
      if (mod(p66,1.0).eq.0.0) then !       old header format
         if (p66.lt.1.1) then       !ASCII
           if(nelgh.lt.10000) then
            WRITE(24,'(4i4,1pe14.7,I5,1X,30A1,1X,A12)')
     $           NELGH,NXH,NYH,NZH,TIME,ikstep,(EXCODE1(I),I=1,30),
     $           'NELT,NX,NY,N'
           else
            WRITE(24,'(i10,3i4,1pe18.9,I9,1X,30A1,1X,A12)')
     $           NELGH,NXH,NYH,NZH,TIME,ikstep,(EXCODE1(I),I=1,30),
     $           'NELT,NX,NY,N'
           endif
         else                       !Binary
            if (nelgh.lt.10000) then
               WRITE(fhdfle,'(4I4,1pe14.7,I5,1X,30A1,1X,A12)')
     $              NELGH,NXH,NYH,NZH,TIME,ikstep,(EXCODE1(I),I=1,30),
     $              'NELT,NX,NY,N'
            else
               write(fhdfle,'(i10,3i4,1P1e18.9,i9,1x,30a1)')
     $         nelgh,nxh,nyh,nzh,time,istep,(excode1(i),i=1,30)
            endif
            call byte_write(fhdfle,20,ierr)
         endif
       else                         !       new header format
         if (p66.eq.0.1) then
            write(24,111)
     $           nelgh,nxh,nyh,nzh,time,istep,excode
         else
             write(fhdfle,111)
     $            nelgh,nxh,nyh,nzh,time,istep,excode
             call byte_write(fhdfle,20,ierr)
         endif
 111     FORMAT(i10,1x,i2,1x,i2,1x,i2,1x,1P1e18.9,1x,i9,1x,a)
      endif

      if(ierr.ne.0) return

      CDRROR=0.0
      if (p66.LT.1.1) then       !       formatted i/o
         WRITE(24,'(6G11.4)')(CDRROR,I=1,NELGH)   ! dummy
      else
C       write byte-ordering test pattern to byte file...
        test_pattern = 6.54321
        call byte_write(test_pattern,1,ierr)
      endif

      return
      end
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      subroutine ps_out_buff(id,p66,tbuf,ierr)

      include 'SIZE'
      include 'TOTAL'

      common/hpts_to_elm/NELGH,NXH,NYH,NZH
      parameter (lpsc9=ldimt1+9)
      parameter (lxyz=lx1*ly1*lz1)
c      common /cbuff1/ tbuf(lxyz,lpsc9)
      real*4          tbuf(NXH*NYH*NXH,lpsc9)
      integer nxyz
      character*11 frmat

      nxyz=NXH*NYH*NZH
      call blank(frmat,11)
      if (id.le.9) then
         WRITE(FRMAT,1801) ID
 1801    FORMAT('(1p',I1,'e14.6)')
      else
         WRITE(FRMAT,1802) ID
 1802    FORMAT('(1p',I2,'e14.6)')
```

```
      endif

      if (p66.lt.1.0) then
C       formatted i/o
        WRITE(24,FRMAT)
     $       ((TBUF(I,II),II=1,ID),I=1,nxyz)
      else
C        C binary i/o
         do ii=1,id
c            call byte_reverse(tbuf,id,ierr)
            call byte_write(tbuf(1,ii),nxyz,ierr)
            if(ierr.ne.0) goto 101
         enddo
      endif
 101  continue

      return
      end
c-----------------------------------------------------------------------
cc*************************************************************************
c      subroutine ps_hpts_create_vtk(nIter, bBinary, chFileNameBase)
cc     ROUTINE DEFINITION
cc     This routine writes out the data from hpts in an
cc     curvilinear vtk file format
cc
c      !COMMON BLOCKS
c      include 'SIZE'
c      include 'TOTAL'
c      !INPUT VARIABLE DEFINITIONS
c      character*50 chFileNameBase
c      integer nIter
c      logical bBinary
c      !LOCAL VARIABLE DEFINITIONS
c      character*80 chFileNameFull
c      !STEP 1) OPEN VTK FILE
c      if(nid.eq.0)
c         write(chFileNameFull,"(A50,I0,'.vtk')")chFileNameBase,nIter
c         if (bBinary) then
c            call byte_open(chFileNameFull,ierr)
c         else
c            open(unit=50,file=chFileNameFull,form='formatted',
c     $       status='unknown')
c         endif
c      endif
c      !STEP 2) WRITE HEADER
c      !STEP 3) WRITE MESH INFO
c      !STEP 4) WRITE VARIABLES
c      !STEP 4a) WRITE SCALARS
c      !STEP 4b) WRITE VECTORS
c      !STEP 5) CLOSE FILE
c      if(bBinary) then
c         call byte_close(ierr)
c      else
c         close(50)
c      endif
c      end subroutine
c*************************************************************************
c-----------------------------------------------------------------------
c     This subroutine is used to retreive the specific z values for the mesh
      subroutine ps_GetZVal
c
      include 'SIZE'
      include 'TOTAL'
      include 'mpif.h'
      common /nekmpi/nid_,np_,nekcomm,nekgroup,nekreal
      common /myzval/ zval,zvaltol
c
c   variable list
c        INTEGERS:
c        -- i,j -- counter variables
c        -- n   -- number of gll points
c        -- NumVals-- number of values per each spatial location size levels
c        -- levels--  number of spatial locations
```

```
c        -- last  --  place holder for last value to be updated in vector
c        REALS:
c        -- rmax -- max radius to average out to
c        -- r    -- radius of current value
c        -- ztest -- temp value for comparing
c        -- zval -- array of spatial locations size levels
c        -- scalar-- array of scalars to output size levels
c
c   variable declerations
      integer,parameter:: levels=24*(lz1-1)+1
      integer:: n,i,j,k,last,dest,sita,npes
      integer:: oddball, buffSize
      character(13):: filename
      logical::used
      real*8:: ztest,zvaltol
      real*8,dimension(levels)::zval,tempArray
      real::locMax,locMin
      character*80::fout
c
c   1) initialize values
c      call MPI_Comm_Size(nekcomm,npes,ierr)
      npes=np_
      oddball=npes-npes/2*2
      write(filename,"('node',I0,'.dat')")nid
c      open(unit=nid,file=filename)
c
      n=nx1*ny1*nz1*nelv
      zvaltol=1.e-9
      buffSize=levels*8
c
      do i=1,levels
        tempArray(i)=0.0
        zval(i)=-10.0
      enddo
c
c   2) intialize zval
c
      last=0
c   -2a) initialize local zval
      do i=1,nelv!nz1*nelv
      do k=1,nz1
        ztest=zm1(1,1,k,i)
        do j=1,levels
c          if (zval(j,me).eq.ztest)then
         if (abs(zval(j)-zm1(1,1,k,i)).lt.zvaltol)then
            !repeated value, exit loop
              exit
          elseif(j.gt.last)then
             !original value add to the end of the vector
              zval(j)=zm1(1,1,k,i)
              last=j
              exit
         endif
        enddo
c        write(6,*),"ACTVAL",zm1(1,1,k,i)
      enddo
      enddo
c   -2aa) Set all unused values to zero
      last=last+1
      do i=last,levels
         zval(i)=0.0
      enddo
      last=last-1

c   -2b) MPI communication to send zval to all processors
c   -2b1) Set up give and receive processors
      if(nid.gt.npes/2-1)then
        dest=nid-(npes/2-1)
        dest=nid-2*dest+1
      else
        dest=npes/2-nid
        dest=nid+2*dest-1
      endif
```

```
c      call nekgsync
      do sita=1,npes/2+oddball
c
         if(dest.ge.npes.and.nid.lt.npes/2)then
            dest=dest-npes/2-oddball
         endif
         if(dest.ge.npes/2.and.nid.ge.npes/2)then
            dest=0-oddball
         endif
c         write(6,*),"ME",nid,"DEST",dest,npes

c         call nekgsync
         if(nid.le.npes/2-1)then
            call csend(dest,zval,buffSize,dest,dest)
         elseif(dest.gt.-1)then
            call crecv(nid,tempArray,buffSize)
         endif
         call nekgsync
         if(nid.gt.npes/2-1.and.dest.gt.-1)then
            call csend(dest,zval,buffSize,dest,dest)
         else
            call crecv(nid,tempArray,buffSize)
         endif
         call nekgsync

c    -2c) Sort values and remove duplicates
         do j=1,levels
           used=.false.
           do k=1,last
           if(abs(zval(k)-tempArray(j)).lt.zvaltol)then
                used=.true.
                exit
           endif
           enddo
           if(used.eqv..false.)then
             zval(last+1)=tempArray(j)
             last=last+1
           endif
         enddo
c
         dest=dest+1
c
      enddo
c   -2d) Bubble sort values to put them all in the same order on each processor
      do i=1,levels
         do j=1,i
            if(zval(i).lt.zval(j))then
               tempArray(1)=zval(i)
               zval(i)=zval(j)
               zval(j)=tempArray(1)
             endif
         enddo
      enddo
c      do i=1,levels
c         if(nid.eq.0)then
c            write(6,*),"ZVAL", i, zval(i)
c         endif
c            write(nid,*),"ZVAL", i, zval(i)
c      enddo
c      call nekgsync
c      close(unit=nid)
      end
c-----------------------------------------------------------------------
      subroutine ps_PlanarAverage(iter,pwr,prefix)
      include 'SIZE'
      include 'TOTAL'
      include 'mpif.h'
      common /nekmpi/ nid_,np_,nekcomm,nekgroup,nekreal
      common /myzval/ zval,zvaltol
      common /mystuff/ tx(lx1,ly1,lz1,lelt)
     $               , ty(lx1,ly1,lz1,lelt)
     $               , tz(lx1,ly1,lz1,lelt)
```

```
      integer,parameter:: levels=24*(lz1-1)+1
      real*8:: zvaltol
      real*8,dimension(levels)::zval

      real,dimension(levels,7)::scalar,tempScalar
      real,dimension(levels,1)::flucs,tempFlucs
      real,dimension(levels):: wght,tempWght
      real::myWght,dTheta
      integer::e,i,j,k,ii,n,nt
      integer::f,nflds,nflucs,iter,pwr
      character*80 filename
      character*3  prefix

      n=nx1*ny1*nz1*nelv
      nt=nx1*ny1*nz1*nelt
      nflds=7
      nflucs=1

      do i=1,levels
      do j=1,nflds
      scalar(i,j)=0.
      tempScalar(i,j)=0.
      enddo
      wght(i)=0.
      tempWght(i)=0.
      enddo

      do i=1,levels
      do j=1,nflucs
      flucs(i,j)=0.
      tempFlucs(i,j)=0.
      enddo
      enddo
      do e=1,nelv
      !---Find the desired face via normal
        f=1
        do while(unz(1,1,f,e).ne.-1.0.and.f.lt.6)
          f=f+1
        enddo
      !---March over horizontal planes of face
        do k=1,nz1
      !-----Find the appropriate height
          ii=1
          do while(abs(zval(ii)-zm1(1,1,k,e))
     $             .gt.zvaltol.and.ii.lt.levels)
            ii=ii+1
c       if(nid.eq.0)then
c        write(6,*)f,ii,zval(ii),zm1(1,1,k,e),zvaltol
c     $   ,abs(zval(ii)-zm1(1,1,k,e))
c       endif
          enddo
c       if(nid.eq.0)then
c        write(6,*)f,ii,zval(ii),zm1(1,1,k,e),zvaltol
c     $   ,abs(zval(ii)-zm1(1,1,k,e))
c       endif
      !-----March over the face and weight the points
          do i=1,nx1*ny1
           if(xm1(i,1,k,e).ne.0.)then
              dTheta=atan(ym1(i,1,k,e)/xm1(i,1,k,e))
           else
              dTheta=0.
           endif
           myWght=area(i,1,f,e)
           wght(ii)=wght(ii)+myWght
           scalar(ii,1)=scalar(ii,1)+(vx(i,1,k,e)*cos(dTheta)
     $                  +vy(i,1,k,e)*sin(dTheta))**pwr*myWght
           scalar(ii,2)=scalar(ii,2)+vz(i,1,k,e)**pwr*myWght
           scalar(ii,3)=scalar(ii,3)+t(i,1,k,e,1)**pwr*myWght
           scalar(ii,4)=scalar(ii,4)+t(i,1,k,e,2)**pwr*myWght
           scalar(ii,5)=scalar(ii,5)+t(i,1,k,e,3)**pwr*myWght
           scalar(ii,6)=scalar(ii,6)+(t(i,1,k,e,1)*
     $                                vz(i,1,k,e))**pwr*myWght
           scalar(ii,7)=scalar(ii,7)+tz(i,1,k,e)**pwr*myWght
```

```
           enddo!--i-loop
         enddo!--k-loop
       enddo!---e-loop
       !---Sum over procesors
       call gop(scalar,tempScalar,'+  ',levels*nflds)
       call gop(wght,tempWght,'+  ',levels)
       call nekgsync
       !---Area average
       do j=1,nflds
       do i=1,levels
           scalar(i,j)=scalar(i,j)/wght(i)
       enddo
       enddo
       !----Compute Fluctuations
       do e=1,nelv
       !---Find the desired face via normal
         f=1
         do while(unz(1,1,f,e).ne.-1.0.and.f.lt.6)
           f=f+1
         enddo
       !---March over horizontal planes of face
         do k=1,nz1
       !-----Find the appropriate height
           ii=1
           do while(abs(zval(ii)-zm1(1,1,k,e))
      $               .gt.zvaltol.and.ii.lt.levels)
              ii=ii+1
           enddo
       !-----March over the face and weight the points
       !---!!!MAKE SURE YOU DOUBLE CHECK SIGNS ON MEAN PROFILE!!!
           do i=1,nx1*ny1
            myWght=area(i,1,f,e)
            flucs(ii,1)=flucs(ii,1)+(t(i,1,k,e,1)-scalar(ii,3)**
      $                      (1.0/dble(pwr)))**pwr*myWght
            enddo!--i-loop
         enddo!--k-loop
       enddo!---e-loop

       !---Sum over processors
       call gop(flucs,tempFlucs,'+  ',levels*nflucs)
       call nekgsync
       !---Area average
       do j=1,nflucs
       do i=1,levels
           flucs(i,j)=flucs(i,j)/wght(i)
       enddo
       enddo
       !----Dump to File
       if(nid.eq.0)then
          write(filename,"(A3,'prof',I0,'.dat')")prefix,iter
          open(unit=10,file=filename)
          do i=1,levels
             write(10,*)         zval(i)
      $                         ,scalar(i,1)
      $                         ,scalar(i,2)
      $                         ,scalar(i,3)
      $                         ,scalar(i,4)
      $                         ,scalar(i,5)
      $                         ,scalar(i,6)
      $                         ,scalar(i,7)
      $                         ,flucs(i,1)
      $                         ,wght(i)
          enddo
          close(10)
       endif
       end
c-----------------------------------------------------------------------
       subroutine psLoadProfile(dProfile,chFilename)
       !PARAMETERS
       integer,parameter:: nLevels=217
       !IO VARIABLES
       real*8 dProfile(nLevels)
       character*32 chFilename
```

```
      !LOCAL VARIABLES
      integer i
      write(6,*),"Loading profile from ",chFileName
      open(unit=30,file=chFileName)
      do i=1,nLevels
         read(30,*)dProfile(i)
      enddo
      close(30)
      end

c-----------------------------------------------------------------------
      subroutine psInitStateMinus(chProfile)
      include 'SIZE'
      include 'TOTAL'
      include 'mpif.h'
      common /nekmpi/ nid_,np_,nekcomm,nekgroup,nekreal
      common /myzval/ zval,zvaltol
      character*32 chProfile
      integer,parameter:: levels=24*(lz1-1)+1
      real*8,dimension(levels)::zval,dProfile,dWork

      integer i,j,k,nt,nv

      !Initialize
      do i=1,levels
        dProfile(i)=0.
        dWork(i)=0.
      enddo

      nt=lx1*ly1*lz1*lelt
      nv=lx1*ly1*lz1*lelv

      !Get position
      if(nid.eq.0) write(6,*)"Get Zvals"
      call ps_GetZVal
      !Get Profile on rank 0
      if(nid.eq.0) call psLoadProfile(dProfile,chProfile)

      if(nid.eq.0) then
      write(6,*)"Mean Profile"
      do i=1,levels
          write(6,*)zval(i),dProfile(i)
      enddo
      endif

      !Send profile to all ranks
      if(nid.eq.0) write(6,*)"Send profile to all ranks"
      call gop(dProfile,dWork,'+  ',levels)
      !Compute changes
      if(nid.eq.0) write(6,*)"Flip Velocity"
      do i=1,nv
         vx(i,1,1,1)=-vx(i,1,1,1)
         vy(i,1,1,1)=-vy(i,1,1,1)
         vz(i,1,1,1)=-vz(i,1,1,1)
      end do
      if(nid.eq.0) write(6,*)"Flip temperature"
      do i=1,nt
         do j=1,levels
            if(abs(zval(j)-zm1(i,1,1,1)).lt.zvaltol)then
               t(i,1,1,1,1)=2.0*dProfile(j)-t(i,1,1,1,1)
               exit
            end if
         end do
      end do

      end subroutine
c-----------------------------------------------------------------------
      subroutine psStateTransform
      include 'SIZE'
      include 'TOTAL'
      include 'mpif.h'

      !LOCAL VARIABLES
```

```
      integer nt, nv, nelp !points in t field, v field and p field
      integer i,j,k
      !DEFINE VARIABLES
      nt=lx1*ly1*lz1*nelt
      nv=lx1*ly1*lz1*nelv
      nelp=lx2*ly2*lz2*nelv

      !BEGIN CALCULATIONS
      do i=1,nelt
      !temperature shift and 180 degree rotation
         t(i,1,1,1,1)=-1.0*(t(i,1,1,1,1)-0.5)+0.5
      enddo
      do i=1,nelv
      !velocity shift and 180 degree rotation
         vx(i,1,1,1)=-vx(i,1,1,1)
         vy(i,1,1,1)=-vy(i,1,1,1)
         vz(i,1,1,1)=-vz(i,1,1,1)
      enddo
      do i=1,nelp
      !temperature shift and 180 degree rotation
         pr(i,1,1,1)=-pr(i,1,1,1)
      enddo
      call outpost(vx,vy,vz,pr,t(1,1,1,1,1),'FL1')
      !call prepost(.true.,'FLP')
      call exitt
      end subroutine
c-----------------------------------------------------------------------
       subroutine ps_usr_dt

c     Change timestep if courno exceeds specified limits

       include 'SIZE'
       include 'TOTAL'

      common /orthbi/ nprv(2)

      real dtmax,p93,p94,p95,p14
      save dtmax,p93,p94,p95,p14

      real mycourno,dtprev,mycmax,mycmin,myctarg,dt_temp
      real mycfl


      MYCTARG = 0.7
      MYCMAX  = 0.8
      MYCMIN  = 0.6

c     Save initial parameter
      IF (istep.le.5) THEN
         IF (istep.eq.0) THEN
            DTMAX = abs(param(12))
            p93   = param(93)
            p94   = param(94)
            p95   = param(95)
            p14   = param(14)
          ENDIF
            param(14)=0.0

          DT    = 3.E-03
          param(12) = -DT
          return
      ENDIF


      call compute_cfl(MYCOURNO,vx,vy,vz,DT)
C      call compute_ale_cfl(mycfl,vx,vy,vz,wx,wy,wz,1.0)
C      MYCOURNO = mycfl*DT
      IF (nid.eq.0) write(6,28) istep,time,"C=",mycourno,
     $    'My CFL ',session
   28  format(i7,1p1e14.7,2x,a3,2x,0p1F7.3,2x,a7,2x,a7)

      DTPREV = DT
```

```
      IF(MYCOURNO.lt.1e-3)then
         DT=DTPREV*2.0
         param(14)=0
         go to 101
      else
         param(14)=p14
      END IF

      IF (MYCOURNO.GE.MYCMAX .OR. MYCOURNO.LE.MYCMIN) THEN
         DT_TEMP=DT*(MYCTARG/MYCOURNO)
         IF (DT_TEMP.LT.DTMAX) THEN
            DT = DT_TEMP
         ELSE
            DT = DTMAX
         ENDIF
      ENDIF
  101 continue
c      write(6,*) 'DTCALC',istep,dt,myctarg,mycourno,DT_TEMP

c     Synchronize time step for multiple sessions
      if (ifneknek) dt=uglmin(dt,1)

c     Turn off projection if DT changed
      IF (ABS(DT-DTPREV).GT.1e-7) THEN
         param(93) = 0
         param(94) = 0
         param(95) = 0
         p95   = istep
         nprv(:) = 1
        IF (NID.eq.0) WRITE(6,39) "Change: DT = ",DT,session
      ELSE
         param(93) = p93   ! turn projection back on
         param(94) = p94
         param(95) = p95
      ENDIF
      param(12) = -DT


 39   FORMAT(A13,1pE14.7,2X,A10)

      return
      end
c-----------------------------------------------------------------------
```

```
c-----------------------------------------------------------------------
c SUBROUTINES DEVELOPED BY PHIL SAKIEVICH
c FOR TRANSFORMATIONS OF GRID
c
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      subroutine SymFlip(symConstant)
c    ******************************************
c    ***** MODIFIED VERSION OF HPTS IN REPO******
c     flips all fields about the mid plane of the simulation
c    ******************************************
C
c     evaluate velocity, temperature, pressure and ps-scalars
c     for list of points (read from hpts.in) and dump results
c     into a file (hpts.out).
c     note: read/write on rank0 only
c
c     ASSUMING LHIS IS MAX NUMBER OF POINTS TO READ IN ON ONE PROCESSOR

      include 'SIZE'
      include 'TOTAL'

      parameter(nfldm=5)
      parameter(lSYM=lx1*ly1*lz1)


      real pts      (ldim,lSYM)
     $               , fieldout (nfldm,lSYM)
     $               , dist     (lSYM)
     $               , rst      (lSYM*ldim)


      integer rcode(lSYM),elid(lSYM),proc(lSYM)

      common /scrcg/  pm1 (lx1,ly1,lz1,lelv) ! mapped pressure
      common /outtmp/ wrk (lx1*ly1*lz1*lelt,nfldm)
      character*3    prefix

      logical iffind
      real symConstant
      integer icalld,npoints,npts,nelmNum
      integer iEnd,iEndTotal
      save    icalld,npoints,npts
      data    icalld  /0/
      data    npoints /0/

      nxyz  = nx1*ny1*nz1
      ntot  = nxyz*nelt
      nbuff = lhis      ! point to be read in on 1 proc.
      npts = nxyz
      if(nio.eq.0) write(6,*) 'swap points based on symmetry'
      call prepost_map(0)  ! maps axisymm and pressure
      ! pack working array
      ! modified to dump out corrdinates as well
      nflds = ndim
      if(ifvo) then
        call copy(wrk(1,1),vx,ntot)
        call copy(wrk(1,2),vy,ntot)
        if(if3d) call copy(wrk(1,3),vz,ntot)
        nflds = ndim
      endif
      if(ifpo) then
        nflds = nflds + 1
        call copy(wrk(1,nflds),pm1,ntot)
      endif
      if(ifto) then
        nflds = nflds + 1
        call copy(wrk(1,nflds),t,ntot)
      endif
      do i = 1,ldimt
         if(ifpsco(i)) then
           nflds = nflds + 1
           call copy(wrk(1,nflds),T(1,1,1,1,i+1),ntot)
```

```
         endif
       enddo


       if(nflds.ne.nfldm.and.nid.eq.0)write(6,*)"Error nflds ",nflds,
      $   nfldm
       call intpts_setup(-1.0,inth_hpts) ! use default tolerance
       nelmNum=1 !initialize element
       iEnd=0 !set flag for end of elements to zero
       iEndTotal=0
c
c      BEGIN ELEMENT BASED LOOP
c
       do while (nelmNum.le.nelt.and.iEnd.eq.0)

         call load_element(pts,npts,npoints,nelmNum,symConstant)

       ! interpolate
         call findpts(inth_hpts,rcode,1,
     &                proc,1,
     &                elid,1,
     &                rst,ndim,
     &                dist,1,
     &                pts(1,1),ndim,
     &                pts(2,1),ndim,
     &                pts(3,1),ndim,npts)

         do i=1,npts
            ! check return code
            if(rcode(i).eq.1) then
              if (dist(i).gt.1e-12) then
                 nfail = nfail + 1
                 IF (NFAIL.LE.5) WRITE(6,'(a,1p4e15.7)')
     &      ' WARNING: point on boundary or outside the mesh xy[z]d^2:'
     &      ,(pts(k,i),k=1,ndim),dist(i)
              endif
            elseif(rcode(i).eq.2) then
              nfail = nfail + 1
              if (nfail.le.5) write(6,'(a,1p3e15.7)')
     &         ' WARNING: point not within mesh xy[z]: !',
     &         (pts(k,i),k=1,ndim)
            endif
         enddo
       ! evaluate input field at given points
       do ifld = 1,nflds
          call findpts_eval(inth_hpts,fieldout(ifld,1),nfldm,
     &                      rcode,1,
     &                      proc,1,
     &                      elid,1,
     &                      rst,ndim,npts,
     &                      wrk(1,ifld))
       enddo


       !Write results back to the current element
       do i=1,nxyz
          vx(i,1,1,nelmNum)=fieldout(1,i)
          vy(i,1,1,nelmNum)=fieldout(2,i)
          vz(i,1,1,nelmNum)=-fieldout(3,i)
          pm1(i,1,1,nelmNum)=-fieldout(4,i)
          t(i,1,1,nelmNum,1)=symConstant-fieldout(5,i)
       end do

       if(nelmNum.lt.nelt) then
         nelmNum=nelmNum+1
       else
         iEnd=1
       end if

       call igop(iEnd,iEndTotal,'*  ',1)
       if(nid.eq.0)then
          write(6,*),"Elm num",nelmNum,"of",nelt,"iEnd equals",iend
       end if
```

```
      end do

      call prepost_map(1)  ! maps back axisymm arrays

      if(nio.eq.0) write(6,*) 'done :: swap points based on symmetry'

      return
      end
c-----------------------------------------------------------------------
      subroutine load_element(pts,npts,npoints,nelmNum,symConstant)
c     npts=local count; npoints=total count

      include 'SIZE'
      include 'TOTAL'
      !include 'PARALLEL'

      parameter (lt2=2*lx1*ly1*lz1*lelt)
      common /scrns/ xyz(ldim,lt2)
      common /scruz/ mid(lt2)  ! Target proc id
      integer, INTENT(in):: nelmNum
      real    pts(ldim,npts)
      real    symConstant
      integer i

      !load pnts
      do i=1,lx1*ly1*lz1
         pts(1,i)=xm1(i,1,1,nelmNum)
         pts(2,i)=ym1(i,1,1,nelmNum)
         pts(3,i)=symConstant-zm1(i,1,1,nelmNum)
      end do


      return
      end
```

```
C**********************************************************************
C   VARIABLES USED IN MYFFT ROUTINE
C**********************************************************************
      include "fftw3.f" !lib with FFTW type definitions

C*****BOOLEAN DATA*****************************************
      ! Variable names and definitions
C----------------------------------------------------------------------
    !Physical dimensions to perform FFT
      logical bFFTd2t(3)
C----------------------------------------------------------------------
C----------User Input Start
C DETERMINE WHICH DIRECTIONS YOU WANT TO TRANFORM HERE:
      data bFFTd2t /.True.,.False.,.False./
C----------User Input End


C*****COMMON BLOCK FOR INTEGERS****************************************
      ! Variable names and definitions
C----------------------------------------------------------------------
C--------------Start user input for parameters
C-----------------------------------------------
    !Number of processors that will use FFT
      integer, parameter::nFFTp2c=32
    !Number of blocks you domain is divided into in each direction
      integer, parameter::nFFTblX=1,nFFTblY=8,nFFTblZ=nFFTp2c/nFFTbly

    !Local sampling of field (any FFT dim must span global domain)
      integer, parameter::nFFTlx1=32,nFFTly1=8,nFFTlz1=16
    !Order of FFT (1d,2d,3d)=1,2,3
      integer, parameter::nFFTorder=1
    !Number of fields to perform FFT on
      integer, parameter::nFFTflds=ldim+1+ldimt !plus 1 is for pressure
    !Total number of points in the local FFT domain
      integer, parameter::nFFTtotal=nFFTlx1*nFFTly1*nFFTlz1
    !Destroy fftw plan manually (1) or automatically (anything else)
      integer, parameter::nFFTdmanual=0
C-----------------------------------------------
C---------------End user input for parameters
C-----------------------------------------------
    !Parameter for error codes in findpnts
      integer nFFTrcode(nFFTtotal)
    !Parameter for global domain size
      integer, parameter::nFFTGx=nFFTblX*nFFTlx1,nFFTGy=nFFTblY*nFFTly1,
     $  nFFTGz=nFFTblZ*nFFTlz1
    !Refernce for dimensions of FFT grid in vector form
      integer nFFTdims(3)
    !Array for element id's where points exist
      integer nFFTelid(nFFTtotal)
    !Array for which processor the points are stored on
      integer nFFTproc(nFFTtotal)
    !Handle for intpts routines
      integer nFFTitp_handle
    !FFT plan used by FFTW
      integer*8 nFFTplan

      COMMON /INTMYFFT/ nFFTrcode,nFFTelid,nFFTproc,
     $ nFFTitp_handle
      COMMON /INT8MYFFT/ nFFTplan
C----------------------------------------------------------------------


C*****COMMON BLOCK FOR REALS*******************************************
      ! Variable names and definitions
C----------------------------------------------------------------------
    !Spatial localtion of points for interpolated values
      real rFFTpts(ldim,nFFTtotal)
    !Working array for findpnts
      real rFFTwrk(lx1*ly1*lz1*lelt,nFFTflds)
    !Values that are stored from findpnts
      real rFFTvals(nFFTflds,nFFTtotal)
    !Distance away from point
      real rFFTdist(nFFTtotal)
```

```
    !Location of point in the local coordinates for the given element
      real rFFTrst(nFFTtotal*ldim)

C-----------------------------------------------------------------------
      COMMON /REALMYFFT/ rFFTpts, rFFTwrk,rFFTvals, rFFTdist, rFFTrst
C-----------------------------------------------------------------------

C*****COMMON BLOCK FOR COMPLEX*****************************************
      ! Variable names and definitions
C-----------------------------------------------------------------------
      double complex cFFTvals(nFFTtotal,nFFTflds)
C      complex(8) cFFTvals(nFFTtotal,nFFTflds)
C-----------------------------------------------------------------------
      COMMON /COMPMYFFT/ cFFTvals
C-----------------------------------------------------------------------
      data nFFTdims /nFFTlx1,nFFTly1,nFFTlz1/
```

```
C**********************************************************************
C     ROUTINES FOR PERFORMING FFT's INSIDE NEK5000
C     REQUIRES FFTW (www.fftw.org), BUILT AND TESTED WITH v3.3.4
C
C     CODE WRITEN BY PHIL SAKIEVICH (psakievi@asu.edu)
C
C     UTILIZES DEFINITIONS IN fftw3.f by Matteo Frigo and Steven Johnson
C      (http://people.sc.fsu.edu/~jburkardt/f77_src/fftw3/fftw3.html)
C
C     INCLUDE FILE:  fftw3.f
C     LINK LIBS:    -lfftw3 -lw
C
C     IMPORTANT DOCUMENTATION:
C     http://www.fftw.org/fftw3_doc/Calling-FFTW-from-Legacy-Fortran.html#Calling-FF
TW-from-Legacy-Fortran
C**********************************************************************

C     OVERVIEW:
C     These subroutines are designed to allow one to set up a set of
C     points on a given processor get their values from somewhere in the
C     parallel envirnoment using intpts and then perform FFT's on them
C     locally using routines from fftw3.3.4.
C
C     It is the users responsibility to ensure that the:
C
C     1) The way the points are defined are compatible with the FFT they
C        intend to perform.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     SUBROUTINE MYFFT
      subroutine MyFFT()
      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'

      character*32 chFilename
      integer nFFTSetup  !variable to determine if setup has been called
      integer nFFToutstep
      data nFFTSetup,nFFToutstep /0,0/ !initialize value to zero
      save nFFTSetup,nFFToutstep     !save value between subsequent calls


      ! 1) Make sure a valid number of processors are present
      if(nFFTp2c.gt.np) then
         if(nid.eq.0)write(6,*),"ERROR FFT: FFTp2c> Total processors"
         call exitt()
      end if

      ! 2) Perform setup procedures
      if(nFFTSetup.eq.0) then
           call FFT_Define_Points()
          ! call FFT_Find_Points()
      end if
      !if(nFFTdmanual.ne.1.or.nFFTsetup.eq.0)then
      !     call FFT_Create_Plan()
      !endif
      nFFTSetup=1
      ! 3) Perform Interpolation
           call FFT_Find_Points()
          ! call FFT_Interp_Points()
      if(nFFTdmanual.ne.1.or.nFFTsetup.eq.0)then
           call FFT_Create_Plan()
      endif
      if(nid.lt.nFFTp2c)then
       write(chFilename,"(A4)")"test"
       call dwritevts(nid,nFFTdims,nFFTflds,rFFTpts,rFFTvals,chFilename)
      endif
      ! 3a) Convert velocity to cylindrical coordinates
           call FFT_Cart2Cyl_Vel()
      ! 4) Perform Transform
           call FFT_Transform()
      ! 5) Destroy Plan
          if(nFFTdmanual.ne.1)then
           call FFT_Destroy_Plan()
```

```
          end if
       ! 6) If desired write to file
       !     call FFT_ASCII_PRINT()
             call FFT_OUTPUT_WAVENUMBERS(nFFToutstep)
             call FFT_ENERGY_REPORT(nFFToutstep)
       nFFToutstep=nFFToutstep+1
       return
       end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     SUBROUTINE DEFINE POINTS
C     Users should use this to define the sampling points they want for
C     their FFT's. This example will be for points in a cylinder with
C     FFT in the theta direction
       subroutine FFT_Define_Points()
       include 'SIZE'
       include 'TOTAL'
       include 'MYFFT'
       common /myDomainRange/rMax,zMax,zMin,rRPnt,rRWgt,rZPnt,rZWgt
       real PI

       real rRPnt(nFFTGy),rRWgt(nFFTGy),rZPnt(nFFTGz),rZWgt(nFFTGz)
       integer i,j,k,ii,ntot
       integer iG,jG,kG
       real dR,dTheta,dZ,Rval,Tval,Zval,Xval,Yval
       real rMax,zMax,zMin

       ntot=lx1*ly1*lz1*lelv
       zMax=glmax(zm1,ntot)
       zMin=glmin(zm1,ntot)
       rMax=glmax(xm1,ntot)
       PI=4.0*atan(1.0)

       !Determine R locations
       call ZWGJD(rRPnt,rRWgt,nFFTGy,0.,0.)
       !Determin Z locations
       call ZWGJD(rZPnt,rZWgt,nFFTGz,0.,0.)
C       dR=xMax/(nFFTly1*nFFTbly-1)
       dTheta=2.0*PI/(nFFTlx1*nFFTblx)
C       dZ=1.0/(nFFTlz1*nFFTblZ-1)

       ! Good idea to zero out any thing that won't be using an FFT
       if(nid.ge.nFFTp2c) then
        do i=1,nFFTtotal
           rFFTpts(1,i)=0.0
           rFFTpts(2,i)=0.0
           if(if3d) rFFTpts(3,i)=0.0
        end do
       else !Initialize fields for processors of interest
        do k=1,nFFTlz1
           do j=1,nFFTly1
              do i=1,nFFTlx1
                 ii=i+(j-1)*nFFTlx1+(k-1)*nFFTlx1*nFFTly1
                 call FFT_L2G(i,j,k,iG,jG,kG,nid)

                 Rval=rMax*(0.5*rRPnt(jG)+0.5)
                 Tval=dTheta*(i-1)
                 Zval=zMin+(zMax-zMin)*(0.5*rZPnt(kG)+0.5)

                 rFFTpts(1,ii)=Rval*cos(Tval)
                 rFFTpts(2,ii)=Rval*sin(Tval)
                 rFFTpts(3,ii)=ZVal

              end do
           end do
        end do
       end if

       if(nid.eq.0) write(6,*) 'done::FFT points declared'

       return
       end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     SUBROUTINE FIND POINTS
```

```
C     Find the points that will be used for the FFT and interpolate them
C     to the array rFFTvals.  Note that rFFTvals is type real, and when
C     the FFT is performed the complex values will be held in cFFTvals.
C     DO NOT confuse rFFTvals and cFFTvals
      subroutine FFT_FIND_POINTS()
      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'
      common /scrcg/  pm1 (lx1,ly1,lz1,lelv) ! mapped pressure
      integer nxyz, nflds, iCalled
      integer ntot
      save iCalled
      data iCalled/0/

      nxyz=nx1*ny1*nz1
      ntot=nxyz*nelt

      nFFTitp_handle=0
      nFlds=0
      !Map pressure to grid1
      call prepost_map(0)

      !Perform setup on first call
      if(iCalled.eq.0)call intpts_setup(-1.0,nFFTitp_handle)

      ! pack working array
      if(ifvo) then
        call copy(rFFTwrk(1,1),vx,ntot)
        call copy(rFFTwrk(1,2),vy,ntot)
        if(if3d) call copy(rFFTwrk(1,3),vz,ntot)
        nflds = ndim
      endif

      if(ifpo) then
        nflds = nflds + 1
        call copy(rFFTwrk(1,nflds),pm1,ntot)
      endif

      if(ifto) then
        nflds = nflds + 1
        call copy(rFFTwrk(1,nflds),t,ntot)
      endif

      do i = 1,ldimt
         if(ifpsco(i)) then
           nflds = nflds + 1
           call copy(rFFTwrk(1,nflds),T(1,1,1,1,i+1),ntot)
         endif
      enddo

      !find points
      if(icalled.eq.0)then
      call findpts(nFFTitp_handle,nFFTrcode,1,
     &                 nFFTproc,1,
     &                 nFFTelid,1,
     &                 rFFTrst,ndim,
     &                 rFFTdist,1,
     &                 rFFTpts(1,1),ndim,
     &                 rFFTpts(2,1),ndim,
     &                 rFFTpts(3,1),ndim,nFFTtotal)
       !check return codes
       do i=1,nFFTtotal
            ! check return code
            if(nFFTrcode(i).eq.1) then
              if (rFFTdist(i).gt.1e-12) then
                 nfail = nfail + 1
                 IF (NFAIL.LE.5) WRITE(6,'(a,1p4e15.7)')
     &      ' WARNING: point on boundary or outside the mesh xy[z]d^2:'
     &      ,(rFFTpts(k,i),k=1,ndim),rFFTdist(i)
              endif
            elseif(nFFTrcode(i).eq.2) then
              nfail = nfail + 1
              if (nfail.le.5) write(6,'(a,1p3e15.7)')
```

```
     &        ' WARNING: point not within mesh xy[z]: !',
     &        (rFFTpts(k,i),k=1,ndim)
           endif
         enddo
        icalled=1
       endif
       !Map pressure back to grid2
       !call prepost_map(1)

       if(nid.eq.0) write(6,*) 'done::FFT points found'

       !return
       !end
C-----------------------------------------------------------------------
C     SUBROUTINE FFT INTERP POINTS
C     This is the routine where the actual interpolation takes place
       !subroutine FFT_INTERP_POINTS()

       !include 'SIZE'
       !include 'TOTAL'
       !include 'MYFFT'

       !common /scrcg/  pm1 (lx1,ly1,lz1,lelv) ! mapped pressure

       !Map pressure to grid1
       !call prepost_map(0)

       !evaluate field at given points
       do ifld = 1,nFFTflds
          call findpts_eval(nFFTitp_handle,rFFTvals(ifld,1),nFFTflds,
     &                      nFFTrcode,1,
     &                      nFFTproc,1,
     &                      nFFTelid,1,
     &                      rFFTrst,ndim,nFFTtotal,
     &                      rFFTwrk(1,ifld))
          do j=1,nFFTtotal
             !cFFTvals(j,ifld)=DCMPLX(rFFTwrk(j,ifld),0.d0)
             cFFTvals(j,ifld)=DCMPLX(rFFTvals(ifld,j),0.d0)
          end do
       enddo

       !Map pressure back to grid2
       call prepost_map(1)
       if(nid.eq.0) write(6,*) 'done::FFT points interpolation'

       return
       end
C-----------------------------------------------------------------------
C     SUBROUTINE FFT CREATE PLAN
C     FFTW requires plans for performing FFT's to be generated. These
C     must also be destroyed later.  Documenation can be found in the
C     fftw resources online.
       subroutine FFT_CREATE_PLAN()

       include 'SIZE'
       include 'TOTAL'
       include 'MYFFT'

       ! I will use one of the most general plans fftw_plan_many_dft_r2c
       ! it requires the following parameters:

       integer rank,n(nFFTorder),howmany, idist, odist, istride, ostride,
     $  inembed(nFFTorder), onembed(nFFTorder), swap

       !This shares the memory for n and the embedding.  My routine
       !does not function with embedded FFT for padding etc. If you need
       !embedding you will need to modify this routine
       equivalence (n,inembed)
       equivalence (n,onembed)

       !Set up FFT plan parameters
       rank=nFFTorder !order of FFT 1d, 2d, 3d
```

```
      ! 1-D FFT parameters----------------------------------------
      if(rank.eq.1)then
        if(bFFTd2t(1))then
           n(1)=nFFTlx1 !size of FFT
           istride=1  !distance between points in memory
           idist=nFFTlx1 !dist between first elements of different FFTs
           howmany=nFFTtotal/nFFTlx1*nFFTflds !total num FFTs
         else if(bFFTd2t(2)) then
           n(1)=nFFTly1
           istride=nFFTlx1
           idist=1
           howmany=nFFTtotal/nFFTly1*nFFTflds
         else if(bFFTd2t(3)) then
           n(1)=nFFTlz1
           istride=nFFTlx1*nFFTly1
           idist=nFFTtotal
           howmany=nFFTtotal/nFFTlz1*nFFTflds
         else
           go to 100
         endif
      else
        go to 100
      end if

      ostride=istride
      odist=idist
      if(nid.eq.0)then
        write(6,*)"FFT rank:",rank
        write(6,*)"FFT n:",n
        write(6,*)"FFT howmany:",howmany
        write(6,*)"FFT inembed:",inembed
        write(6,*)"FFT istride:",istride
        write(6,*)"FFT idist:",idist
        write(6,*)"FFT onembed:",onembed
        write(6,*)"FFT ostride:",ostride
        write(6,*)"FFT odist:",odist
       ! write(6,*)"FFT rVals:",rFFTvals
      endif
      call dfftw_plan_many_dft(nFFTplan,rank,n,howmany,cFFTvals,
     $                         inembed,istride,idist,cFFTvals,
     $                         onembed,ostride,odist,
     $                         FFTW_FORWARD,FFTW_ESTIMATE)

      if(nid.eq.0) write(6,*) 'done::FFT plan creation'
      return
  100 if(nid.eq.0) write(6,*) 'ERROR:: unsupported bFFTd2t entry'
      call exitt()
      return
      end
C-----------------------------------------------------------------------
C     SUBROUTINE PERFORM FFT
      subroutine FFT_TRANSFORM()

      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'

      call dfftw_execute_dft(nFFTplan,cFFTvals,cFFTvals)
      if(nid.eq.0) write(6,*) 'done::FFT transform'

      return
      end
C-----------------------------------------------------------------------
C     SUBROUTINE DESTROY FFT PLAN
      subroutine FFT_DESTROY_PLAN()

      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'

      call dfftw_destroy_plan(nFFTplan)
      if(nid.eq.0) write(6,*) 'done::FFT plan destroyed'
      return
```

```
      end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     SUBROUTINE PRINT FFT DATA TO TEXT FILE
      subroutine FFT_ASCII_PRINT()

      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'

      integer nFileNum,nFileErr
      character*32 filename
      data nFileNum /1/
      save nFileNum

      !Each processor will write to the same file one at a time
      do i=1,nFFTp2c
       !if i=my rank then write data
       if(nid.eq.i-1) then
         write(filename,"('myFFT',I0,'.dat')")nFileNum

         !if rank=0 create new file, else open old file
         if(i.eq.1)then
           open(unit=10,file=filename,iostat=nFileErr,status='REPLACE')
         else
           open(unit=10,file=filename,iostat=nFileErr,status='OLD',
     $       access='APPEND')
         end if

         do j=1,nFFTtotal
           write(10,*) nid,GetAngle(rFFTpts(1,j),rFFTpts(2,j)),
     $                        (real(cFFTvals(j,ii)),ii=5,5)
         end do

         close(unit=10)
       endif
       call nekgsync()
      end do
      if(nid.eq.0) write(6,*) 'done::FFT results printed to file'
      return
      end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     subroutine FFT_OFFSET
C     Determines local to global array offset based on lexigraphical
C     ordering i.e. x+y*nX+z*nX*nY
C     Must use even number of processors divisible for 1d
      subroutine FFT_OFFSET(nMyR,nXoff,nYoff,nZoff)
      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'
      !INPUT
      integer nMyR
      !OUTPUT
      integer nXoff,nYoff,nZoff

      nXoff=0
      nYoff=0
      nZoff=0

      if(nFFTblX.gt.1) nXoff=mod(nMyR,nFFTbLX)
      if(nFFTblY.gt.1.and.nFFTblX.gt.1) then
          nYoff=nMyR/nFFTblX
      else
          if(nFFTblY.gt.1) nYoff=mod(nMyR,nFFTblY)
      endif
      if(nFFTblz.gt.1) nZoff=nMyR/(nFFTblX*nFFTblY)

      end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     SUBROUTINE FFT_OUTPUT_WAVENUMBERS()
C     This subroutine is to output the data for each wave number in a
C     seperate file.  Data is collected onto rank0 and written there
      subroutine FFT_OUTPUT_WAVENUMBERS(nFFToutstep)
      include 'SIZE'
```

```
      include 'TOTAL'
      include 'MYFFT'

      integer,parameter::nInFFT=nFFTlx1
      integer,parameter::nInSlice=nFFTly1*nFFTbly*nFFTlz1*nFFTblZ

      !Define Size of array for wave number
      real dataOutReal(nFFTflds,nInSlice)
      real dataOutComp(nFFTflds,nInSlice)
      real dataWork(nFFTflds,nInSlice)
      !TODO determine parameters for size
      real dataOutPts(3,nInSlice)
      real dataWrkPts(3,nInSlice)
      real theta
      !Define number of wave numbers to output
      integer nMyWave,nFFToutstep
      integer iG,jG,kG
      integer,dimension(3):: nDimension
      character*32 chFileName

      data nDimension /nFFTGy,1,nFFTGz/
      !Loop over the wave numbers
      do i=1,nInFFT/2+1
        !Zero out workign arrays
        call rzero(dataOutReal,nInSlice*nFFTflds)
        call rzero(dataOutComp,nInSlice*nFFTflds)
        call rzero(dataWork,nInSlice*nFFTflds)
        call rzero(dataOutPts,nInSlice*3)
        call rzero(dataWrkPts,nInSlice*3)
        !populate local spot in the array
        if(nid.lt.nFFTp2c)then
        do k=1,nFFTlz1
          do j=1,nFFTly1
            call FFT_L2G(i,j,k,iG,jG,kG,nid)
            !kg=(k)+mod(nid,nFFTblY)*nFFTly1
            !jg=(j)+(nid/nFFTblY)*nFFTlz1
            iii=(i)+(j-1)*nFFTlx1+(k-1)*nFFTlx1*nFFTly1
            ii=(jg)+(kg-1)*nFFTly1*nFFTbly
         !   if(nid.eq.1) write(6,*) i,j,k,ii,iii,"INDEX"
            if(i.lt.nInFFT/2)then
               nMyWave=i-1
            else
               nMyWave=nInFFT-i
            endif
           ! theta=GetAngle(rFFTpts(1,iii),rFFTpts(2,iii))
            !Convert to cylindrical coordinates
            dataOutPts(1,ii)=sqrt(rFFTpts(1,iii)**2+rFFTpts(2,iii)**2)
            dataOutPts(2,ii)=0.d0
            dataOutPts(3,ii)=rFFTpts(3,iii)

            do jj=1,nFFTflds
               dataOutReal(jj,ii)=real(cFFTvals(iii,jj))/dble(nInFFT)
               dataOutComp(jj,ii)=dimag(cFFTvals(iii,jj))/dble(nInFFT)
            end do

          enddo
        enddo
        endif
        !gather procedure
        call gop(dataOutPts,dataWrkPts,'+  ',3*nInSlice)
        call gop(dataOutReal,dataWork,'+  ',nInSlice*nFFTflds)
        call gop(dataOutComp,dataWork,'+  ',nInSlice*nFFTflds)
        !write data to file on rank0
         write(chFileName,"('./Snaps/',I0,'/SymS1_',I0,'_TSTEP')")
     $                    i-1,nMyWave
        if(nid.eq.0) call dwritevtsc(nFFToutstep,nDimension,nFFTflds,
     $     dataOutPts,
     $                     dataOutReal,dataOutComp,chFileName)
       end do
       return
       end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine FFT_Cart2Cyl_Vel()
```

```
      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'

      real locTheta,velX,velY

      do i=1,nFFTtotal
         velX=cFFTvals(i,1)
         velY=cFFTvals(i,2)
             locTheta=getangle(rFFTpts(1,i),rFFTpts(2,i))
         cFFTvals(i,1)=velX*cos(locTheta)+velY*sin(locTheta)
         cFFTvals(i,2)=-velX*sin(locTheta)+velY*cos(locTheta)
      end do

      return
      end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function GetAngle(x,y)
      real x,y
         if(x.gt.0.0)then
            GetAngle=atan(y/x)
         else
            if(x.lt.0.0)then
              GetAngle=atan(y/x)+4.0*atan(1.0)
            else
               if(y.ge.0.0)then
                  GetAngle=2.0*atan(1.0)
               else
                  GetAngle=-2.0*atan(1.0)
               end if
            endif
         endif
      return
      end
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine FFT_ENERGY_REPORT(nFFToutstep)
C     OUTPUT THE ENERGY FOR EACH WAVE NUMBER INEGRATED OVER THE R-Z
C     PLANE USEING NUMERICAL INTEGRATION
C      ---> \int_0^R\int_0^H A(r,z)*A(r,z) r dr dz (*is complex conj)
      include 'SIZE'
      include 'TOTAL'
      include 'MYFFT'
      common /myDomainRange/rMax,zMax,zMin,rRPnt,rRWgt,rZPnt,rZWgt
      real rRPnt(nFFTGy),rRWgt(nFFTGy),rZPnt(nFFTGz),rZWgt(nFFTGz)
      real EnergyWave(nFFTflds,nFFTlx1),EnergyWork(nFFTflds,nFFTlx1)
      real radius
      integer nFFToutstep,ii,iii,iG,jG,kG
      character*80 fileName
      call rzero(EnergyWave,nFFTflds*nFFTlx1)
      do ii=1,nFFTlz1
      do j=1,nFFTly1
        do i=1,nFFTlx1
           do k=1,nFFTflds
              call FFT_L2G(i,j,ii,iG,jG,kG,nid)
              iii=(j-1)+(ii-1)*nFFTly1
              radius=sqrt(rFFTpts(1,i+iii*nFFTlx1)**2+
     $             rFFTpts(2,i+iii*nFFTlx1)**2)

C             Ek=sum_z sum_r phi(r,z)*CC[phi(r,z)] *wr*R/2*wz*H/2
C             Divide by nFFTlx1 to normalize the Fourier Coefficient
              EnergyWave(k,i)=EnergyWave(k,i)+
     $         real(dconjg(cFFTvals(i+iii*nFFTlx1,k)/dble(nFFTlx1))
     $               *(cFFTvals(i+iii*nFFTlx1,k)/dble(nFFTlx1)))
     $               *radius*rRWgt(jG)*rZWgt(kG)
     $               *0.25*rMax*(zMax-zMin)
           end do
        end do
      end do
      end do
      call gop(EnergyWave,EnergyWork,'+  ',nFFTflds*nFFTlx1)

      write(fileName,"('./Snaps/EnergyReport_',I0,'.dat')")nFFToutstep
      if(nid.eq.0)then
```

```
        open(unit=10,file=fileName,status='REPLACE')
        do i =1,nFFTlx1/2
          write(10,*)i-1,(EnergyWave(k,i),k=1,nFFTflds)
        end do
        do i =nFFTlx1/2+1,nFFTlx1
          write(10,*)i-(nFFTlx1+1),(EnergyWave(k,i),k=1,nFFTflds)
        end do
        close(unit=10)
      end if
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine FFT_L2G(iL,jL,kL,iG,jG,kG,me)
C     MAP LOCAL INDEX IN GRID TO GLOBAL INDEX
C     I-Innermost loop corresponding to nFFTlx1
C     J-Middle loop corresponding to nFFTly1
C     K-Outermost loop corresponding to nFFTlz1
C     me-mpi rank
      include 'SIZE'
      include 'MYFFT'
      integer iL,jL,kL,iG,jG,kG,me
c      write(6,*)'INSIDE FFT_L2G',nFFTbly,me
c      write(6,*) iL,jL,kL,iG,jG,kG,me
      kG=kL+(me/nFFTblY)*nFFTlz1
      jG=jL+mod(me,nFFTblY)*nFFTly1
      iG=iL
c      write(6,*) iL,jL,kL,iG,jG,kG,me
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function CGL_POINT(N,I)
C     Chebyshev-Gauss-Lobatto quadrature point
      !N is total number of points (N-1 polynomial)
      !I is current point in series from 1:N
      integer N,I
      CGL_POINT=-cos(4.0*atan(1.0)*dble(I-1)/dble(N-1))
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function CGL_WEIGHT(N,I)
C     Chebyshev-Gauss-Lobatto quadrature weight
C     i from 1:N
      integer N,I
      if(I.eq.1.or.I.eq.N)then
        CGL_WEIGHT=4.0*atan(1.0)*0.5/dble(N-1)
      else
        CGL_WEIGHT=4.0*atan(1.0)/dble(N-1)
      end if
      CGL_WEIGHT=CGL_WEIGHT*sqrt(1.0-CGL_POINT(N,I)**2)
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function CG_POINT(N,I)
C     Chebyshev-Gauss quadrature point
C     i from 1:N
      integer N,I
      PI=4.0*atan(1.0)
      CG_POINT=-cos((2.0*i-1)*PI/(2.0*dble(N)))
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function CG_WEIGHT(N,I)
      integer N,I
      CG_WEIGHT=4.0*atan(1.0)/dble(N)*sqrt(1.0-CG_POINT(N,I)**2)
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function QuadPnt(N,I)
      integer N,I
      QuadPnt=CG_POINT(N,I)
      !QuadPnt=CGL_POINT(N,I)
      return
      end
```

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      real function QuadWgt(N,I)
      integer N,I
      QuadWgt=CG_Weight(N)
      !QuadWgt=CGL_Weight(N,I)
      return
      end
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <iostream>
#include <string>
#include <sstream>
#include "vtkVersion.h"
#include "vtkDoubleArray.h"
#include "vtkFloatArray.h"
#include "vtkPointData.h"
#include "vtkPoints.h"
#include "vtkXMLStructuredGridWriter.h"
#include "vtkXMLPStructuredGridWriter.h"
#include "vtkStructuredGrid.h"
#include "vtkSmartPointer.h"
#include "psVtkOutput.h"

void cpphello_()
{
  std::cout<<"hello world from cpp \n";
}

void dwritevtsc_(int* nPartition, int* nSize, int* nFlds, double* dPnts, double* dRF
lds,double* dCFlds, char* chFileName, int nLenFN)
{
  int nTotal=(nSize[0]*nSize[1]*nSize[2]);
  //Assign filename to string, trim white spaces and then add .vts
  chFileName[nLenFN--]='\0'; //null terminate string
        std::string stFileName,stPFileName;
  std::stringstream ss;
  for(int i=0;i<nLenFN;i++)
  {
    if(chFileName[i]!=' ')
      stFileName+=chFileName[i];
    else
      break;
  }
  stPFileName=stFileName;
  ss<<*nPartition;
  stFileName+="_";
  stFileName+=ss.str();
  stFileName+=".vts";
  //std::cout<<"nTotal: "<<nTotal<<", "<<*nFlds<<", "<<*nPartition<<" "+stFileName<<
"\n";
  //std::cout<<"nSize: "<<nSize[0]<<", "<<nSize[1]<<", "<<nSize[2]<<" "+stFileName<<
"\n";

  //Step 1: setup grid, points and variable objects
  vtkSmartPointer<vtkStructuredGrid> sGrid =
    vtkSmartPointer<vtkStructuredGrid>::New();
  vtkSmartPointer<vtkPoints> points =
    vtkSmartPointer<vtkPoints>::New();
  vtkSmartPointer<vtkDoubleArray> R_velocity =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> R_pressure =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> R_temperature =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> C_velocity =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> C_pressure =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> C_temperature =
    vtkSmartPointer<vtkDoubleArray>::New();

  //Step 2: Set Grid dimensions and variable tuples
  sGrid->SetDimensions(nSize[0],nSize[1],nSize[2]);
  points->Allocate(nTotal);
  R_velocity->SetNumberOfComponents(3);
  R_pressure->SetNumberOfComponents(1);
  R_temperature->SetNumberOfComponents(1);
  R_velocity->SetNumberOfTuples(nTotal);
  R_pressure->SetNumberOfTuples(nTotal);
  R_temperature->SetNumberOfTuples(nTotal);
  R_velocity->SetName("R_velocity");
  R_pressure->SetName("R_pressure");
```

```
  R_temperature->SetName("R_temperature");
  C_velocity->SetNumberOfComponents(3);
  C_pressure->SetNumberOfComponents(1);
  C_temperature->SetNumberOfComponents(1);
  C_velocity->SetNumberOfTuples(nTotal);
  C_pressure->SetNumberOfTuples(nTotal);
  C_temperature->SetNumberOfTuples(nTotal);
  C_velocity->SetName("C_velocity");
  C_pressure->SetName("C_pressure");
  C_temperature->SetName("C_temperature");

  //Step 3: Copy data into vtk Objects
  int j=0;
  for( int i=0;i<nTotal*3;i+=3)
  {
  //  std::cout<<"i="<<i<<"\n";
    points->InsertPoint(j,&dPnts[i]);
    j++;
  }

  //std::cout<<"Points allocated succesfully\n";
  j=0;
  for(int i=0;i<nTotal*(*nFlds);i+=(*nFlds))
  {
  //  std::cout<<"i="<<i<<"\n";
    R_velocity->InsertTuple(j,&dRFlds[i]);
    R_pressure->InsertTuple(j,&dRFlds[i+3]);
    R_temperature->InsertTuple(j,&dRFlds[i+4]);
    C_velocity->InsertTuple(j,&dCFlds[i]);
    C_pressure->InsertTuple(j,&dCFlds[i+3]);
    C_temperature->InsertTuple(j,&dCFlds[i+4]);
    j++;
  }
  //std::cout<<"Fields allocated succesfully\n";

  //Step 4: Assign values to grid
  sGrid->SetPoints(points);
  sGrid->GetPointData()->AddArray(C_velocity);
  sGrid->GetPointData()->AddArray(C_pressure);
  sGrid->GetPointData()->AddArray(C_temperature);
  sGrid->GetPointData()->SetVectors(R_velocity);
  sGrid->GetPointData()->AddArray(R_pressure);
  sGrid->GetPointData()->SetScalars(R_temperature);
  //std::cout<<"Grid allocated succesfully\n";

  //Step 5: Setup writer object and write file
    vtkSmartPointer<vtkXMLStructuredGridWriter> writer =
      vtkSmartPointer<vtkXMLStructuredGridWriter>::New();
    writer->SetFileName(&stFileName[0]);
    writer->SetInputData(sGrid);
    writer->Write();
  //Step 6: If partition zero write pvts vile
  //if(*nPartition==0)
  /*{
    stPFileName+=".pvts";
    vtkSmartPointer<vtkXMLPStructuredGridWriter> pwriter =
      vtkSmartPointer<vtkXMLPStructuredGridWriter>::New();
    pwriter->SetFileName(&stPFileName[0]);
    pwriter->SetInputData(sGrid);
    pwriter->SetNumberOfPieces(8);
    //pwriter->SetUpdateExtent(ext);
    pwriter->Write();
  }*/
}
void dwritevts_(int* nPartition, int* nSize, int* nFlds, double* dPnts, double* dFld
s, char* chFileName, int nLenFN)
{
  int nTotal=(nSize[0]*nSize[1]*nSize[2]);
  //Assign filename to string, trim white spaces and then add .vts
  chFileName[nLenFN--]='\0'; //null terminate string
        std::string stFileName,stPFileName;
  std::stringstream ss;
  for(int i=0;i<nLenFN;i++)
```

```
  {
    if(chFileName[i]!=' ')
      stFileName+=chFileName[i];
    else
      break;
  }
  stPFileName=stFileName;
  ss<<*nPartition;
  stFileName+="_";
  stFileName+=ss.str();
  stFileName+=".vts";
  //std::cout<<"nTotal: "<<nTotal<<", "<<*nFlds<<", "<<*nPartition<<" "+stFileName<<
"\n";
  //std::cout<<"nSize: "<<nSize[0]<<", "<<nSize[1]<<", "<<nSize[2]<<" "+stFileName<<
"\n";

  //Step 1: setup grid, points and variable objects
  vtkSmartPointer<vtkStructuredGrid> sGrid =
    vtkSmartPointer<vtkStructuredGrid>::New();
  vtkSmartPointer<vtkPoints> points =
    vtkSmartPointer<vtkPoints>::New();
  vtkSmartPointer<vtkDoubleArray> velocity =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> pressure =
    vtkSmartPointer<vtkDoubleArray>::New();
  vtkSmartPointer<vtkDoubleArray> temperature =
    vtkSmartPointer<vtkDoubleArray>::New();

  //Step 2: Set Grid dimensions and variable tuples
  sGrid->SetDimensions(nSize[0],nSize[1],nSize[2]);
  points->Allocate(nTotal);
  velocity->SetNumberOfComponents(3);
  pressure->SetNumberOfComponents(1);
  temperature->SetNumberOfComponents(1);
  velocity->SetNumberOfTuples(nTotal);
  pressure->SetNumberOfTuples(nTotal);
  temperature->SetNumberOfTuples(nTotal);
  velocity->SetName("velocity");
  pressure->SetName("pressure");
  temperature->SetName("temperature");

  //Step 3: Copy data into vtk Objects
  int j=0;
  for( int i=0;i<nTotal*3;i+=3)
  {
  //  std::cout<<"i="<<i<<"\n";
    points->InsertPoint(j,&dPnts[i]);
    j++;
  }

  //std::cout<<"Points allocated succesfully\n";
  j=0;
  for(int i=0;i<nTotal*(*nFlds);i+=(*nFlds))
  {
  //  std::cout<<"i="<<i<<"\n";
    velocity->InsertTuple(j,&dFlds[i]);
    pressure->InsertTuple(j,&dFlds[i+3]);
    temperature->InsertTuple(j,&dFlds[i+4]);
    j++;
  }
  //std::cout<<"Fields allocated succesfully\n";

  //Step 4: Assign values to grid
  sGrid->SetPoints(points);
  sGrid->GetPointData()->SetVectors(velocity);
  sGrid->GetPointData()->AddArray(pressure);
  sGrid->GetPointData()->SetScalars(temperature);

  //Step 5: Setup writer object and write file
    vtkSmartPointer<vtkXMLStructuredGridWriter> writer =
      vtkSmartPointer<vtkXMLStructuredGridWriter>::New();
    writer->SetFileName(&stFileName[0]);
    writer->SetInputData(sGrid);
```

```
    writer->Write();
  //Step 6: If partition zero write pvts vile
  //if(*nPartition==0)
  /*{
    stPFileName+=".pvts";
    vtkSmartPointer<vtkXMLPStructuredGridWriter> pwriter =
      vtkSmartPointer<vtkXMLPStructuredGridWriter>::New();
    pwriter->SetFileName(&stPFileName[0]);
    pwriter->SetInputData(sGrid);
    pwriter->SetNumberOfPieces(8);
    //pwriter->SetUpdateExtent(ext);
    pwriter->Write();
  }*/
}
```

```
#ifndef PSVTKOUTPUT_
#define PSVTKOUTPUT_

extern "C" {
void cpphello_();
void dwritevtsc_(int* nPartition, int* nSize, int* nFlds, double* dPnts, double* dRF
lds,double* dCFlds,char* chFileName, int nLenFN);
void dwritevts_(int* nPartition, int* nSize, int* nFlds, double* dPnts, double* dFld
s,char* chFileName, int nLenFN);
}

#endif
```

```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 29 11:16:59 2016

@author: psakievich
"""
import modred as mr
import numpy as np
import Quadratures
#VTK RELATED STUFFS
from vtk import vtkXMLStructuredGridReader,vtkXMLStructuredGridWriter, \
    vtkStructuredGrid
from vtk.numpy_interface import dataset_adapter as dsa
'''
Vector class
This class operates on the flow field variables
as a single, flattened vector.  The vector
interfaces with the VTK structured grid.
Scalar*Vector, Vector+Vector and
(Vector,Vector).  The actual variables that
are used in the in the inner product are defined
by the variables __MyRealData and __MyImagData.
'''
class MrVtkVector(mr.Vector):
    #use to define which datasets for inner product
    __MyRealData=[3,5]
    __MyImagData=[0,2]
    def __init__(self,vtkStrGrid):
        self.data=vtkStrGrid
    def __add__(self, other):
        """Return an object that is self+other for all fields
        """
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_me=dsa.WrapDataObject(self.data)
        math_data=dsa.WrapDataObject(new_data)
        math_other=dsa.WrapDataObject(other.data)
        numFlds=len(math_me.PointData.keys())
        for i in range(numFlds):
            math_data.PointData[i][:]= \
                math_me.PointData[i][:]+ \
                math_other.PointData[i][:]

        return MrVtkVector(new_data)

    def __mul__(self,scalar):
        """Return an object that is self*scalar for all fields
        """
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_data=dsa.WrapDataObject(new_data)
        math_me=dsa.WrapDataObject(self.data)
        numFlds=len(math_me.PointData.keys())
        numReal=int(numFlds/2)
        for i in range(numReal):
            math_data.PointData[i+numReal][:]= \
                math_me.PointData[i+numReal][:]*np.real(scalar)- \
                math_me.PointData[i][:]*np.imag(scalar)
            math_data.PointData[i][:]= \
                math_me.PointData[i][:]*np.real(scalar)+ \
                math_me.PointData[i+numReal][:]*np.imag(scalar)
        return MrVtkVector(new_data)

    def inner_product(self,other):
        weighted_me=self.weighted_copy()
        math_me=dsa.WrapDataObject(weighted_me.data)
        math_other=dsa.WrapDataObject(other.data)
        IP=0.0
        for i in range(len(self.__MyImagData)):
            IP=IP+np.vdot(math_me.PointData[self.__MyRealData[i]][:]+ \
                1j*math_me.PointData[self.__MyImagData[i]][:], \
                math_other.PointData[self.__MyRealData[i]][:]+ \
                1j*math_other.PointData[self.__MyImagData[i]][:])
```

```
        return IP

    def complex_conjugate(self):
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_data=dsa.WrapDataObject(new_data)
        math_me=dsa.WrapDataObject(self.data)
        numFlds=len(math_me.PointData.keys())
        for i in range(numFlds/2):
            math_data.PointData[i][:]*=-1.0
        return MrVtkVector(new_data)

    def integrated_values(self):
        weighted_me=self.weighted_copy()
        math_me=dsa.WrapDataObject(weighted_me.data)
        numFlds=len(math_me.PointData.keys())
        k=0
        for i in range(numFlds):
           if(len(math_me.PointData[i].shape)>1):
               k=k+math_me.PointData[i].shape[1]
           else:
               k=k+1
        result=np.empty(k)
        j=0
        for i in range(numFlds):
            if(len(math_me.PointData[i].shape)>1):
                for k in range(math_me.PointData[i].shape[1]):
                    result[j]=np.sum(math_me.PointData[i][:,k])
                    j=j+1
            else:
                result[j]=np.sum(math_me.PointData[i][:])
                j=j+1
        return result

    def get_rc_lists(self):
        return (self.__MyRealData,self.__MyImagData)

    def weight_matrix(self,QD=Quadratures.GaussLegendre()):
        '''
        Weighting matrix for the numerical integration. Different
        Quadratures can be specified
        '''
        dims=self.data.GetDimensions()
        bounds=self.data.GetPoints().GetBounds()
        B=np.array([bounds[1]-bounds[0],bounds[3]-bounds[2],bounds[5]-bounds[4]])
        wz=QD.Weights(dims[2])
        wr=QD.Weights(dims[0])
        weights=np.outer(wz,wr) #r is fastest varying in dataset
        weights=np.reshape(weights,dims[0]*dims[2])
        math_me=dsa.WrapDataObject(self.data)
        weights=weights*math_me.Points[:,0] #multiply by R
        weights=weights*0.25*B[0]*B[2] #multiply by jacobian
        return weights

    def weighted_copy(self):
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_new=dsa.WrapDataObject(new_data)
        w=self.weight_matrix()
        nFields=len(math_new.PointData.keys())
        for i in range(nFields):
            if(len(math_new.PointData[i].shape)>1):
                for j in range(math_new.PointData[i].shape[1]):
                    math_new.PointData[i][:,j]=math_new.PointData[i][:,j]*w
            else:
                math_new.PointData[i][:]=math_new.PointData[i][:]*w
        return MrVtkVector(new_data)
'''
Vector handle
'''
class MrVtkVecHandle(mr.VecHandle):
    def __init__(self, vec_path, base_handle=None, scale=None):
        mr.VecHandle.__init__(self,base_handle,scale)
```

```
        self.vec_path=vec_path

    def _get(self):
        reader=vtkXMLStructuredGridReader()
        reader.SetFileName(self.vec_path)
        reader.Update()
        return(MrVtkVector(reader.GetOutput()))

    def _put(self,vec):
        writer=vtkXMLStructuredGridWriter()
        writer.SetInputData(vec.data)
        writer.SetFileName(self.vec_path)
        writer.Write()
class MrVtkVecHandleCreateFluctuation(mr.VecHandle):
    def __init__(self, vec_path_inst, vec_path_mean,base_handle=None, scale=None):
        mr.VecHandle.__init__(self,base_handle,scale)
        self.vec_path=vec_path_inst
        self.vec_path_mean=vec_path_mean

    def _get(self):
        reader=vtkXMLStructuredGridReader()
        reader.SetFileName(self.vec_path)
        reader.Update()
        hMean=MrVtkVecHandle(self.vec_path_mean)
        return(MrVtkVector(reader.GetOutput())+ \
               -1.0*hMean.get())

    def _put(self,vec):
        writer=vtkXMLStructuredGridWriter()
        writer.SetInputData(vec.data)
        writer.SetFileName(self.vec_path)
        writer.Write()

class MrVtkVecHandleOperateOnFluctuation(mr.VecHandle):
    def __init__(self, vec_path_inst, vec_path_mean,base_handle=None, scale=None):
        mr.VecHandle.__init__(self,base_handle,scale)
        self.vec_path=vec_path_inst
        self.vec_path_mean=vec_path_mean

    def _get(self):
        reader=vtkXMLStructuredGridReader()
        reader.SetFileName(self.vec_path)
        reader.Update()
        hMean=MrVtkVecHandle(self.vec_path_mean)
        return(MrVtkVector(reader.GetOutput())+ \
               -1.0*hMean.get())

    def _put(self,vec):
        hMean=MrVtkVecHandle(self.vec_path_mean)
        vec+=hMean.get()
        writer=vtkXMLStructuredGridWriter()
        writer.SetInputData(vec.data)
        writer.SetFileName(self.vec_path)
        writer.Write()
'''
 Namespace functions
'''
def inner_product(v1,v2):
    return v1.inner_product(v2)

def point_product(v1,v2):
    new_data=vtkStructuredGrid()
    new_data.DeepCopy(v1.data)
    math_new=dsa.WrapDataObject(new_data)
    math_v1=dsa.WrapDataObject(v1.data)
    math_v2=dsa.WrapDataObject(v2.data)
    nFields=len(math_new.PointData.keys())
    offset=nFields//2
    for i in range(nFields//2):
        math_new.PointData[i+offset][:]= \
            math_v1.PointData[i+offset][:]*math_v2.PointData[i+offset][:]- \
            math_v1.PointData[i][:]*math_v2.PointData[i][:]
        math_new.PointData[i][:]= \
```

```
            math_v1.PointData[i][:]*math_v2.PointData[i+offset][:]+ \
            math_v1.PointData[i+offset][:]*math_v2.PointData[i][:]
    return MrVtkVector(new_data)

def point_division(v1,v2):
    new_data=point_product(v1,v2.complex_conjugate())
    divisor=point_product(v2,v2.complex_conjugate())
    math_new=dsa.WrapDataObject(new_data.data)
    math_div=dsa.WrapDataObject(divisor.data)
    nFields=len(math_new.PointData.keys())
    offset=nFields//2
    for i in range(offset):
        math_new.PointData[i+offset][:]= \
            math_new.PointData[i+offset][:]/math_div.PointData[i+offset][:]
        math_new.PointData[i][:]= \
            math_new.PointData[i][:]/math_div.PointData[i+offset][:]
    return new_data
```

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov  9 13:30:09 2016
Class for interacting with vtk files where all data is R space
@author: psakievi
"""
import modred as mr
import numpy as np
from vtk import vtkStructuredGrid, vtkXMLStructuredGridReader, \
    vtkXMLStructuredGridWriter
from vtk.numpy_interface import dataset_adapter as dsa
#import MrImaginaryVtk as miv

class MrVtkVector(mr.Vector):
    def __init__(self,vtkStrGrid):
        self.data=vtkStrGrid
    def __mul__(self,scalar):
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_me=dsa.WrapDataObject(self.data)
        math_new=dsa.WrapDataObject(new_data)
        numFlds=len(math_me.PointData.keys())
        for i in range(numFlds):
            math_new.PointData[i][:]=math_me.PointData[i][:]*scalar
        return MrVtkVector(new_data)
    def __add__(self, other):
        """Return an object that is self+other for all fields
        """
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_me=dsa.WrapDataObject(self.data)
        math_data=dsa.WrapDataObject(new_data)
        math_other=dsa.WrapDataObject(other.data)
        numFlds=len(math_me.PointData.keys())
        for i in range(numFlds):
            math_data.PointData[i][:]= \
                math_me.PointData[i][:]+ \
                math_other.PointData[i][:]

        return MrVtkVector(new_data)
    def inner_product(self,other):
        weighted_me=self.weighted_copy()
        math_me=dsa.WrapDataObject(weighted_me.data)
        math_other=dsa.WrapDataObject(other.data)
        numFlds=len(math_me.PointData.keys())
        IP=0.0
        for i in range(numFlds):
            IP=IP+np.vdot(np.transpose(math_me.PointData[i][:]), \
                math_other.PointData[i][:])
        return IP
    def power(self,power):
        new_data=vtkStructuredGrid()
        new_data.DeepCopy(self.data)
        math_me=dsa.WrapDataObject(self.data)
        math_new=dsa.WrapDataObject(new_data)
        numFlds=len(math_me.PointData.keys())
        for i in range(numFlds):
            math_new.PointData[i][:]=math_me.PointData[i][:]**power
        return MrVtkVector(new_data)
class MrVtkVecHandle(mr.VecHandle):
    def __init__(self, vec_path, base_handle=None, scale=None):
        mr.VecHandle.__init__(self,base_handle,scale)
        self.vec_path=vec_path

    def _get(self):
        reader=vtkXMLStructuredGridReader()
        reader.SetFileName(self.vec_path)
        reader.Update()
        return(MrVtkVector(reader.GetOutput()))

    def _put(self,vec):
        writer=vtkXMLStructuredGridWriter()
        writer.SetInputData(vec.data)
```

```
        writer.SetFileName(self.vec_path)
        writer.Write()

def point_product(v1,v2):
    new_data=vtkStructuredGrid()
    new_data.DeepCopy(v1.data)
    mv1=dsa.WrapDataObject(v1.data)
    mv2=dsa.WrapDataObject(v2.data)
    mvN=dsa.WrapDataObject(new_data)
    numFlds=len(mv1.PointData.keys())
    for i in range(numFlds):
        mvN.PointData[i][:]=mv1.PointData[i][:]*mv2.PointData[i][:]
    return MrVtkVector(new_data)
def point_division(v1,v2):
    new_data=vtkStructuredGrid()
    new_data.DeepCopy(v1.data)
    mv1=dsa.WrapDataObject(v1.data)
    mv2=dsa.WrapDataObject(v2.data)
    mvN=dsa.WrapDataObject(new_data)
    numFlds=len(mv1.PointData.keys())
    for i in range(numFlds):
        mvN.PointData[i][:]=mv1.PointData[i][:]/mv2.PointData[i][:]
    return MrVtkVector(new_data)
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Nov 11 19:23:32 2016

Quadratures for numerical integration

@author: psakievich
"""
import numpy as np
import scipy.special as ss

class GaussLobattoChebyshev():
    def Point(self,n,i):
        return -np.cos(np.pi*i/(n-1.0))
    def Weight(self,n,i):
        n1=int(n)
        i1=int(i)
        if (i1==0 or i1==n1-1):
            return np.pi/(n-1)*0.5*np.sqrt(1.0-self.Point(n,i)**2)
        else:
            return np.pi/(n-1)*np.sqrt(1.0-self.Point(n,i)**2)
    def Points(self,n):
        a=np.empty(int(n))
        for i in range(int(n)):
            a[i]=self.Point(n,i)
        return a
    def Weights(self,n):
        a=np.empty(int(n))
        for i in range(int(n)):
            a[i]=self.Weight(n,i)
        return a
class GaussChebyshev():
    def Point(self,n,i):
        return -np.cos((2.0*float(i)+1.0)*np.pi/(2.0*float(n)))
    def Weight(self,n,i):
        p=self.Point(int(n),i)
        w=np.pi/float(n)*np.sqrt(1.0-p**2)
        return w
    def Points(self,n):
        a=np.empty(int(n))
        for i in range(int(n)):
            a[i]=self.Point(n,i)
        return a
    def Weights(self,n):
        w=np.empty(int(n))
        for i in range(int(n)):
            w[i]=self.Weight(n,i)
        return w
class GaussLegendre():
    def Points(self,n):
        p,w=ss.p_roots(n)
        return p
    def Weights(self,n):
        p,w=ss.p_roots(n)
        return w
    def Point(self,n,i):
        return self.Points(n)[i]
    def Weight(self,n,i):
        return self.Weights(n)[i]
class GaussLobattoLegendre():
    '''
    pg. 61 Karniadakis and Sherwin "Spectral/hp Element Methods for
    Computational Fluid Dynamics" Secion Edition
    '''
    def Points(self,n):
        p=np.empty(n)
        w=np.empty(n)
        p[1:n-1],w[1:n-1]=ss.j_roots(n-2,1.0,1.0)
        p[0]=-1.0
        p[n-1]=1.0
        return p
    def Weights(self,n):
```

```
        w=np.empty(n)
        p=self.Points(n)
        for i in range(n):
            w[i]=2.0/(n*(n-1)*ss.eval_legendre(n-1,p[i])**2)
        return w
    def Point(self,n,i):
        return self.Points(n)[i]
    def Weight(self,n,i):
        return self.Weights(n)[i]

def Converge(QuadratureClass,start=5,stop=50,inc=5):
    error=[]
    pnts=[]
    for i in range(start,stop,inc):
        pnts.append(i)
        exact=2.0/3.0 #x^2
        test=np.sum(QuadratureClass.Points(i)**2*QuadratureClass.Weights(i))
        error.append(abs(exact-test))
    return pnts,error
```

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 20 10:21:45 2016

@author: psakievich
"""

from vtk import vtkStructuredGrid, \
vtkXMLStructuredGridReader, \
vtkXMLStructuredGridWriter, \
vtkPoints, \
vtkDoubleArray
import numpy as np
from vtk.numpy_interface import dataset_adapter as dsa
'''
Routine for documenting individual modes
Transform Fourier coefficients back to real
space. One period of the mode is documented
over a user specified angle
'''
def FourierToRealDoc(fileName,iFFTsize,outputFile,ang,modeNumber=1):
    reader=vtkXMLStructuredGridReader()
    #load the grid into memory
    reader=vtkXMLStructuredGridReader()
    reader.SetFileName(fileName)
    reader.Update()
    fourierGrid=(dsa.WrapDataObject(reader.GetOutput()))
    #clean up
    del reader
    #print(modeNumber)
    #setup mesh dimensions
    nR=fourierGrid.GetDimensions()[0]
    nTheta=iFFTsize+1
    nZ=fourierGrid.GetDimensions()[2]
    nTotal=nR*nZ*nTheta
    keys=fourierGrid.PointData.keys()
    theta=np.linspace(0,ang,nTheta)
    #Create 3D grid
    grid3d=vtkStructuredGrid()
    grid3d.SetDimensions(nTheta,nR,nZ)

    points=vtkPoints()
    points.Allocate(nTotal)
    for k in range(nZ):
        for j in range(nR):
            for i in range(nTheta):
                x=fourierGrid.Points[j+k*nR,0]*np.cos(theta[i])
                y=fourierGrid.Points[j+k*nR,0]*np.sin(theta[i])
                z=fourierGrid.Points[j+k*nR,2]
                #print(x,y,z,theta[i],sGrid.Points[j+k*nR,1],sGrid.Points[j+k*nR,2])
                points.InsertNextPoint(x,y,z)
    #Set Up 3D grid
    grid3d.SetPoints(points)
    vel3d=vtkDoubleArray()
    tem3d=vtkDoubleArray()
    pre3d=vtkDoubleArray()

    vel3d.SetName('velocity')
    tem3d.SetName('temperature')
    pre3d.SetName('pressure')

    vel3d.SetNumberOfComponents(3)
    tem3d.SetNumberOfComponents(1)
    pre3d.SetNumberOfComponents(1)

    vel3d.SetNumberOfTuples(nTotal)
    tem3d.SetNumberOfTuples(nTotal)
    pre3d.SetNumberOfTuples(nTotal)
    #Set up iffts and populate 3d Grid
    for k in range(nZ):
        for i in range(nR):
            temp=np.zeros(iFFTsize/2+1,dtype=complex)
            v=np.array([temp.copy(),temp.copy(),temp.copy()])
```

```
            press=temp.copy()
            #set up vectors for ifft

            for ii in range(3):
                v[ii,modeNumber]=complex( \
                    fourierGrid.PointData[keys[3]][i+k*nR,ii], \
                    fourierGrid.PointData[keys[0]][i+k*nR,ii])
            press[modeNumber]=complex( \
                fourierGrid.PointData[keys[4]][i+k*nR], \
                fourierGrid.PointData[keys[1]][i+k*nR])
            temp[modeNumber]=complex( \
                fourierGrid.PointData[keys[5]][i+k*nR], \
                fourierGrid.PointData[keys[2]][i+k*nR])
            #scale by grid size
            #print(i,k,temp)
            v=v*iFFTsize
            press=press*iFFTsize
            temp=temp*iFFTsize
            temp1=np.zeros(nTheta)
            press1=np.zeros(nTheta)
            v1=np.array([np.zeros(nTheta),np.zeros(nTheta),np.zeros(nTheta)])
            #conduct iFFT's
            for ii in range(3):
                v1[ii,0:nTheta-1]=np.fft.irfft(v[ii,:])
                v1[ii,nTheta-1]=v1[ii,0]
            press1[0:nTheta-1]=np.fft.irfft(press)
            temp1[0:nTheta-1]=np.fft.irfft(temp)
            press1[nTheta-1]=press1[0]
            temp1[nTheta-1]=temp1[0]
            #Translate data to cartesiant coordinates for visualization purposes
            for jj in range(nTheta):
                index=jj+i*nTheta+k*nR*nTheta
                vx=v1[0,jj]*np.cos(theta[jj])-v1[1,jj]*np.sin(theta[jj])
                vy=v1[0,jj]*np.sin(theta[jj])+v1[1,jj]*np.cos(theta[jj])
                vel3d.SetTuple3(index,vx,vy,v1[2,jj])
                tem3d.SetTuple1(index,temp1[jj])
                pre3d.SetTuple1(index,press1[jj])
    grid3d.GetPointData().SetVectors(vel3d)
    grid3d.GetPointData().AddArray(pre3d)
    grid3d.GetPointData().SetScalars(tem3d)

    writer=vtkXMLStructuredGridWriter()
    writer.SetFileName(outputFile)
    writer.SetInputData(grid3d)
    writer.Write()
'''
Transform a list of Fourier modes back to real space.
Multiple input files are specifie through fileNames
and output as one file outputFile.

fileNames must be a list i.e. []
iFFTsize must be 2 times larger than the largest
wave number.
'''
def FourierToReal(fileNames,iFFTsize,outputFile):
    numModes=len(fileNames)
    fourierGrids=[]
    modeNumber=[]
    reader=vtkXMLStructuredGridReader()
    #load each of the grids into memory
    for i in range(numModes):
        reader=vtkXMLStructuredGridReader()
        reader.SetFileName(fileNames[i])
        reader.Update()
        fourierGrids.append(dsa.WrapDataObject(reader.GetOutput()))
        tempVar=fileNames[i].split('_')
        modeNumber.append(int(tempVar[1]))
        #clean up
    del reader, tempVar
    #print(modeNumber)
    #setup mesh dimensions
    nR=fourierGrids[0].GetDimensions()[0]
    nTheta=iFFTsize+1
```

```
    nZ=fourierGrids[0].GetDimensions()[2]
    nTotal=nR*nZ*nTheta
    keys=fourierGrids[0].PointData.keys()
    theta=np.linspace(0,2*np.pi,nTheta)
    #Create 3D grid
    grid3d=vtkStructuredGrid()
    grid3d.SetDimensions(nTheta,nR,nZ)

    points=vtkPoints()
    points.Allocate(nTotal)
    for k in range(nZ):
        for j in range(nR):
            for i in range(nTheta):
                x=fourierGrids[0].Points[j+k*nR,0]*np.cos(theta[i])
                y=fourierGrids[0].Points[j+k*nR,0]*np.sin(theta[i])
                z=fourierGrids[0].Points[j+k*nR,2]
                #print(x,y,z,theta[i],sGrid.Points[j+k*nR,1],sGrid.Points[j+k*nR,2])
                points.InsertNextPoint(x,y,z)
    #Set Up 3D grid
    grid3d.SetPoints(points)
    vel3d=vtkDoubleArray()
    tem3d=vtkDoubleArray()
    pre3d=vtkDoubleArray()

    vel3d.SetName('velocity')
    tem3d.SetName('temperature')
    pre3d.SetName('pressure')

    vel3d.SetNumberOfComponents(3)
    tem3d.SetNumberOfComponents(1)
    pre3d.SetNumberOfComponents(1)

    vel3d.SetNumberOfTuples(nTotal)
    tem3d.SetNumberOfTuples(nTotal)
    pre3d.SetNumberOfTuples(nTotal)
    #Set up iffts and populate 3d Grid
    for k in range(nZ):
        for i in range(nR):
            temp=np.zeros(iFFTsize/2+1,dtype=complex)
            v=np.array([temp.copy(),temp.copy(),temp.copy()])
            press=temp.copy()
            #set up vectors for ifft
            for kk in range(numModes):
                for ii in range(3):
                    v[ii,modeNumber[kk]]=complex( \
                        fourierGrids[kk].PointData[keys[3]][i+k*nR,ii], \
                        fourierGrids[kk].PointData[keys[0]][i+k*nR,ii])
                press[modeNumber[kk]]=complex( \
                    fourierGrids[kk].PointData[keys[4]][i+k*nR], \
                    fourierGrids[kk].PointData[keys[1]][i+k*nR])
                temp[modeNumber[kk]]=complex( \
                    fourierGrids[kk].PointData[keys[5]][i+k*nR], \
                    fourierGrids[kk].PointData[keys[2]][i+k*nR])
            #scale by grid size
            #print(i,k,temp)
            v=v*iFFTsize
            press=press*iFFTsize
            temp=temp*iFFTsize
            temp1=np.zeros(nTheta)
            press1=np.zeros(nTheta)
            v1=np.array([np.zeros(nTheta),np.zeros(nTheta),np.zeros(nTheta)])
            #conduct iFFT's
            for ii in range(3):
                v1[ii,0:nTheta-1]=np.fft.irfft(v[ii,:])
                v1[ii,nTheta-1]=v1[ii,0]
            press1[0:nTheta-1]=np.fft.irfft(press)
            temp1[0:nTheta-1]=np.fft.irfft(temp)
            press1[nTheta-1]=press1[0]
            temp1[nTheta-1]=temp1[0]
            #Translate data to cartesiant coordinates for visualization purposes
            for jj in range(nTheta):
                index=jj+i*nTheta+k*nR*nTheta
                vx=v1[0,jj]*np.cos(theta[jj])-v1[1,jj]*np.sin(theta[jj])
```

```
                vy=v1[0,jj]*np.sin(theta[jj])+v1[1,jj]*np.cos(theta[jj])
                vel3d.SetTuple3(index,vx,vy,v1[2,jj])
                tem3d.SetTuple1(index,temp1[jj])
                pre3d.SetTuple1(index,press1[jj])
    grid3d.GetPointData().SetVectors(vel3d)
    grid3d.GetPointData().AddArray(pre3d)
    grid3d.GetPointData().SetScalars(tem3d)

    writer=vtkXMLStructuredGridWriter()
    writer.SetFileName(outputFile)
    writer.SetInputData(grid3d)
    writer.Write()
```

APPENDIX C

CONSENT TO USE PUBLISHED MATERIAL

I, Philip John Sakievich, declare that consent as been obtained from co-authors Dr. Yulia Peet and Dr. Ronald Adrian to use the published work in chapter 3 of this document.

Chapter 3 was published in the International Journal of Heat and Fluid Flow in 2016 with the title *Large-scale thermal motions of turbulent Rayleigh-Bénard convection in a wide aspect-ratio cylindrical domain.*

Additional permission has been obtained from Dr. Ronald Adrian to reuse an image from his 1986 paper *Turbulent thermal convection in wide horizontal fluid layers* in the journal Experiments in Fluids.