STRAINED TURBULENCE AND LOW-DIFFUSIVITY TURBULENT MIXING USING HIGH PERFORMANCE COMPUTING

A Dissertation
Presented to
The Academic Faculty

By

Matthew P. Clay

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

August 2017

STRAINED TURBULENCE AND LOW-DIFFUSIVITY TURBULENT MIXING USING HIGH PERFORMANCE COMPUTING

Approved by:

Professor P. K. Yeung, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Professor Cyrus Aidun
School of Mechanical Engineering
Georgia Institute of Technology

Professor Devesh Ranjan
School of Mechanical Engineering
Georgia Institute of Technology

Professor Marilyn Smith
School of Aerospace Engineering
Georgia Institute of Technology

Professor Edmond Chow
School of Computational Science and Engineering
Georgia Institute of Technology

Date Approved: July 26, 2017

To my wife, Éva

ACKNOWLEDGMENTS

I am very grateful for all of the support I have received from many individuals throughout my time at Georgia Tech, without which completion of this thesis would have been impossible. First, I would like to thank my advisor Professor P. K. Yeung for taking me on as a student during very trying times. His constant support, encouragement, and eagerness to tackle new and challenging problems have been very inspiring. I would also like to thank Professors C. K. Aidun, D. Ranjan, M. J. Smith, and E. Chow for taking the time to be on my thesis committee, and for providing valuable feedback and suggestions which improved the quality of this work. Much of this work would not have been possible without the generous help and support of Professor T. Gotoh of Nagoya Institute of Technology, and it has been a privilege to collaborate with him. I am also indebted to Professors Z. Warhaft of Cornell University, Á. Gylfason of Reykjavik University, and S. Ayyalasomayajula of IIT Bhubaneswar for their encouragement and help with certain aspects of this work.

I greatly appreciate the generous financial support from the Science, Mathematics, and Research for Transformation (SMART) scholarship program and the National Science Foundation, which gave me the freedom to pursue very interesting problems for this thesis. I am truly thankful for the support of the research staff at AFRL/RW and their continued patience while I worked to finish my thesis research. I would also like to thank Dr. Benjamin Wilde, a Georgia Tech graduate from the combustion lab, for his encouragement and advice while I pursued the SMART scholarship. The simulations conducted for this thesis would not have been possible without the generous computer time allocations our research group received through the XSESE, NSF PRAC, and DoE INCITE programs. I would also like to thank the many expert HPC consultants at TACC, NCSA, and OLCF who helped me along the way. At NCSA, Dr. Jing Li and Dr. Gregory Bauer have always been willing to help, even at very

odd hours. Our recent work running on Titan would not have been possible without the generous help of Dr. Wayne Joubert, Dr. Oscar Hernandez, and Jeff Larkin.

During my time at Georgia Tech as both an undergraduate student and graduate student I have had the opportunity to work with many exceptionally talented individuals. It would be impossible to name everyone, but my interactions with them have been some of the most memorable of my life. I would like to thank Professor J. Seitzman and Dr. Yash Kochar for teaching me so much as an undergraduate student, both in the classroom and in the combustion lab. After completing my undergraduate degree, I was very fortunate to be given the opportunity to work on a project with Dr. David Scarborough and Brad Ochs in the combustion lab while pursing a master's degree. Following my work in the combustion lab, I worked in the computational combustion laboratory for a few years under Professor S. Menon. I will never forget the comradery of the research group, which included Dr. Andrew Smith, Dr. Kalyana Gottiparthi, Dr. Joseph Schulz, Dr. Timothy Gallagher, Dr. Balaji Muralidharan, Dr. Reetesh Ranjan, Dr. Michel Akiki, Timothy Dawson, Yusuke Nagaoka, and many others. During my time with Professor Yeung, I have had the privilege to work with some great lab mates, including Shine Zhai, Kiran Ravikumar, Jacob Sebastian, and Mer-Win Cheong. I would like to extend a special thanks to Dr. Dhawal Buaria, who continually fielded my questions and spearheaded the effort to get a code used for this work to run on GPUs.

I would also like to thank my family and friends for their constant encouragement and patience while I worked at Georgia Tech for many years. Most importantly, I would like to thank my wife Éva for her unwavering support. She has endured much throughout the years, ranging from my absent-mindedness while thinking about some research task, to me coming home very late from the lab. I hope one day I can repay her what she has sacrificed in order for me to pursue this degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis presents a fundamental investigation of turbulent fluid flow using direct numerical simulation (DNS), a numerical approach in which exact governing equations are computed without modeling. The emphasis is on large-scale parallel computation and developing novel numerical methodologies which can either model experimental configurations or take into account extreme variations in the physical properties of the system. Simulations are conducted for turbulence and turbulent mixing under axisymmetric contraction, and turbulent mixing at high Schmidt number. The wealth of information available in DNS provides insight into how these flows evolve, and supplies data for many unresolved questions in turbulence theory.

The thesis begins with a study of turbulence and turbulent mixing under axisymmetric contraction, which is relevant to engineering flows through variable cross-section ducts. While axisymmetric contraction has been studied for many decades, the motivation to pursue it further came from the experimental work of Ayyalasomayajula & Warhaft (*J. Fluid Mech.*, vol. 566, 2006, pp. 273–307; AW henceforth), which showed that new behaviors emerge in the evolution of the one-dimensional (1-D) component velocity spectra at sufficiently high Reynolds number. To directly model the AW wind tunnel facility in the DNS, a spatially dependent strain rate profile was developed, and the numerical algorithm was extended to apply strain rates as functions of spatial location in a numerical wind tunnel, not time. Simulations of turbulence at sufficiently high Reynolds number subjected to strain and subsequent relaxation show very similar evolution of the spectra as reported by AW. Specifically, a "double-peak" emerges in the compensated transverse spectrum as a result of fast relaxation of the small scales. Simulations of turbulent mixing in the same numerical configuration are also conducted, motivated by the experiments of Gylfason & Warhaft (*J. Fluid Mech.*, vol. 628, 2009, pp. 339–356), which used the same wind

tunnel as AW. The numerical results show increasing agreement with rapid distortion theory as the strain rate is increased, and that scalars with transverse mean gradients relax following the strain much faster than scalars with streamwise mean gradients.

To simulate turbulent mixing of high Schmidt number (Sc) scalars, a numerical algorithm is developed to efficiently handle the increased resolution requirements of the so-called Batchelor scale of the scalar field, which is $\sqrt{Sc}$ time smaller than the Kolmogorov scale of the velocity field. To combat the high communication cost of Fourier pseudo-spectral (FPS) methods, a dual-grid dual-scheme numerical approach is adopted, which decouples the computation of the passive scalar from the computation of the velocity field (Gotoh *et al.*, *J. Comput. Phys.*, vol. 231, 2012, pp. 7398–7414). The velocity field is computed on a coarse grid that resolves the Kolmogorov scale using traditional FPS methods, whereas the scalar is computed on a fine grid that resolves the Batchelor scale with a combined compact finite difference (CCD) scheme (Mahesh, *J. Comput. Phys.*, vol. 145, 1998, pp. 332–358). The ideas of Gotoh *et al.* (2012) are extended to incorporate the physics of the passive scalar directly in the design of a parallel code intended for $Sc \gg 1$ simulations. Specifically, the disparate resolution requirements of the velocity and scalar fields are handled by an approach in which each field is computed separately in disjoint message passing communicators. Good scalability of the code is maintained by overlapping communication with computation as much as possible. In homogeneous computing environments, namely the XE6 partition of Blue Waters at the University of Illinois, Urbana-Champaign, substantial gains in scalability are obtained by dedicating certain OpenMP threads to perform communication, while others compute concurrently. The code is also ported to run on GPU-accelerated machines, specifically Titan at Oak Ridge National Laboratory, TN, achieving a speedup of 2.7X relative to the CPU-only code at the largest problem size of 8192^3. Here too, scalability is improved, this time by overlapping GPU computations with data movements and communication,

which is made possible through the latest directives added in OpenMP 4.5 — one of the most promising programming models on the path towards exascale in high performance computing.

The newly developed turbulent mixing code is used to generate a DNS database for high Schmidt number passive scalar mixing under the presence of a uniform mean scalar gradient in forced isotropic turbulence at Taylor-scale Reynolds number approximately 140. Scalars with Schmidt numbers ranging from 4 to 512 (the highest of which is comparable to salinity mixing in the ocean) are simulated on Blue Waters and Titan using grid resolutions 1024^3 to 8192^3. Results at moderate Schmidt numbers agree well with previous work at more modest problem sizes. The current simulations greatly extend the maximum Schmidt number compared to prior work at this Reynolds number, and strongly suggest an approach toward isotropy and saturation of intermittency in the scalar field with increasing Schmidt number. Isotropy as a function of scale size is analyzed with the skewness structure function, with an interesting result being the development of a local minimum at 20–30 Batchelor scales as Schmidt number is increased. The shape of the scalar spectrum suggests the emergence of Batchelor scaling with increasing Schmidt number, and shows an exponential decrease in the far-diffusive range. Scalar gradient evolution is analyzed in Fourier space after deriving the governing equation for the scalar gradient covariance spectrum. Results show the nonlinear amplification of scalar gradients by the fluctuating strain rates dominating the scalar gradient evolution over a wide range of scales, and that the spectral budget only has a weak dependence on Schmidt number.

CHAPTER I

INTRODUCTION

Turbulence is the most common state of fluid motion in natural and engineering flows, and is characterized by disorderly fluctuations in three-dimensional (3-D) space and time spanning a wide range of scales. The multiscale nature of turbulence enables turbulent flows to provide very rapid mixing compared to their laminar counterparts: in a turbulent flow, large scale inhomogeneities in material properties, e.g., momentum or the concentration of a transported substance, are gradually broken down into smaller and smaller scales until molecular diffusion smears them out. Such rapid mixing can be useful, for example, in engineering applications like jet engine combustors which require fuel and oxidizer to be mixed at the molecular level before chemical reactions can proceed (Peters, 2000). Turbulent flows are also unsteady, and their sensitivity to changes in their initial and boundary conditions due to strong nonlinear interactions makes a deterministic approach to studying turbulence intractable (Pope, 2000). Instead, a statistical approach in useful in studying many different aspects of turbulent flows, in both laboratory experiments and numerical computations.

In this thesis direct numerical simulation (DNS) is used to compute the evolution of turbulent flows that are of fundamental and practical interest according to exact equations of motion. Since the pioneering simulations of Orszag & Patterson (1972) using 32^3 grid points, DNS has served as a powerful tool for physical understanding and theory development in turbulence research (Moin & Mahesh, 1998; Ishihara *et al.*, 2009). The major challenge when using DNS is that the resolution requirements, i.e., the number of grid points, increase very rapidly with the Reynolds number (Re) (Yakhot & Sreenivasan, 2005), and many flows of interest in engineering and nature occur at high Re. The computation of turbulence using DNS is a recognized grand

challenge in high-performance computing (HPC) (Yokokawa *et al.*, 2002), and recent simulations using multi-petaflop supercomputers have reached problems sizes as large as $\mathcal{O}(1)$ trillion grid points (Yeung *et al.*, 2015; Ishihara *et al.*, 2016), with high Re and good small-scale resolution being the major driving requirements for the numerical configurations. Such simulations provide deep insight into fundamental aspects of turbulence, including the shape of the energy spectrum at high Re (Donzis & Sreenivasan, 2010) and small-scale intermittency (Sreenivasan & Antonia, 1997), i.e., the extreme fluctuations that occur in high Re turbulent flows.

Because of its large computational cost, DNS is typically used for canonical flow configurations where simple boundary conditions permit the use of highly accurate numerical methods. A challenge when using DNS to understand more complex flows is to develop numerical methodologies that can take into account all of the relevant features of the flows of interest, despite all of the simplifications that typically accompany a DNS. For example, to understand the evolution of isotropic turbulence subjected to mean strain in a variable cross-section wind tunnel, one can attempt to model the effects of the experimental strain rates directly in the DNS. For other types of flows, it becomes challenging to properly incorporate the physical parameters controlling the system of interest. Such difficulties arise when using DNS to study the turbulent mixing of a substance of low molecular diffusivity, where the resolution requirements for the substance are much stricter than those for the turbulent velocity field (Gotoh & Yeung, 2013). When attempting DNS of such flows, it is conceivable that the best understanding and most efficient computation would be obtained when using numerical approaches which are designed for the target problem of interest.

In many engineering applications a turbulent flow is subjected to a change in cross-section of the carrying device, e.g., flows through nozzles and diffusers, where the effects of deformation by irrotational mean strain are of great interest. The mean strain rates give rise to anisotropy, which weakens when the strain is removed. As-

suming incompressibility, three principal sub-classes of irrotational mean strain can be identified, namely: (a) axisymmetric contraction (one extensional direction and two equally compressive directions), (b) axisymmetric expansion (two equally extensional directions and one compressive direction) and (c) plane strain (one extensional direction and one compressive direction). A number of experimental (Gence & Mathieu, 1979; Liu *et al.*, 1999; Choi & Lumley, 2001; Ayyalasomayajula & Warhaft, 2006; Brown *et al.*, 2006) and numerical (Lee & Reynolds, 1985; Zusi & Perot, 2013, 2014) studies covering one or more of these sub-classes are known. All three sub-classes are distinct and important; however, axisymmetric contraction and expansion (as well as relaxation therefrom) are more closely related to flows through conduits of variable cross-section in engineering devices, and to converging and diverging sections in laboratory wind tunnel facilities. A number of early experimental (Uberoi, 1956; Mills & Corrsin, 1959; Reynolds & Tucker, 1975; Warhaft, 1980) and theoretical (Batchelor & Proudman, 1954) studies have focused on the response of isotropic turbulence subjected to axisymmetric contraction. Velocity fluctuations are (as expected from the Reynolds stress transport equations) suppressed in the extensional direction but amplified in the compressive directions. The small scales also depart from isotropy when the strain rate is large (Uberoi, 1956). However, in these early studies Re was usually limited, and no attempt was made to investigate the Re dependence of the flow. Further, while the study of turbulent mixing of scalar quantities for this important class of flow is very interesting from an engineering perspective, it has only received limited attention (Warhaft, 1980; Gylfason & Warhaft, 2009).

The first major objective of this thesis is to present a computational investigation of turbulence and turbulent mixing under axisymmetric contraction and subsequent relaxation, with a special interest in conditions corresponding to experiments. For the velocity field, close comparisons are made with the experiments of Ayyalasomayajula & Warhaft (2006) (AW henceforth) in which grid-generated turbulence was passed

through an axisymmetric contraction of area ratio 4:1. Using both passive and active grids, the experiments of AW covered a range of Reynolds numbers (40 to 470, based on the Taylor scale). A significant result was that a qualitative change in the form of the transverse one-dimensional (1-D) spectra occurred during relaxation (referred to as a "double peak") only if the Re was sufficiently high. Increasing departures from rapid distortion theory (RDT) (Savill, 1987) were also observed as Re was increased. The simulations of scalar mixing under axisymmetric contraction follow the experiments of Gylfason & Warhaft (2009) (GW henceforth), which used the same wind tunnel as AW to study the evolution of mild temperature fluctuations generated under the presence of a uniform transverse mean temperature gradient. Here the focus is on the evolution of scalar derivative statistics which form the scalar dissipation rate, a fundamental quantity of interest in many applications (Bilger, 2004).

Although strained turbulence becomes less anisotropic upon the removal of strain (Sarkar & Speziale, 1990; Choi & Lumley, 2001), a full return to isotropy is not guaranteed. For example, the simulations of Chasnov (1995) and Davidson *et al.* (2012) showed that for anisotropic Saffman turbulence (Saffman, 1967; Krogstad & Davidson, 2010), the large scales do not return to isotropy. Changes in the form of the spectra measured by AW also imply that different scales respond differently to both the application and removal of strain. While a considerable body of work is known for the return-to-isotropy problem in Reynolds stress closures (Lumley & Newman, 1977; Speziale, 1991), at a more detailed level, the nature of anisotropy development at different scale sizes, which is important for subgrid-scale modeling (Liu *et al.*, 1999), is still not well understood (Sagaut & Cambon, 2008). To understand the mechanisms involved it is necessary to study spectral transfer resulting from nonlinear interactions and pressure-strain correlations between different scale sizes and velocity components (in the extensional versus compressive directions). Direct numerical simulations using Fourier pseudo-spectral methods are well suited to provide the detailed information

necessary for this purpose.

It is well known that, provided the strain rate is uniform in space, turbulence under irrotational mean strain is homogeneous and can be simulated on a solution domain that deforms with the mean flow (Rogallo, 1981). A number of such simulations, with strain rates held constant in time, have been helpful in examining the structure of Reynolds-stress anisotropy (Lee & Reynolds, 1985), assessing turbulence models (Zusi & Perot, 2013, 2014), and understanding inertial and fluid particle statistics (Lee *et al.*, 2015). However as in the case of Gualtieri & Meneveau (2010) (in conjunction with Chen *et al.* (2006)), a time-dependent strain rate in the DNS is necessary to facilitate comparisons with experiment, where typically the turbulence is measured as it evolves in space along the centerline of a wind tunnel, rather than in time.

In this thesis a smoothly-varying time-dependent strain rate is developed to closely mimic the laboratory conditions of AW and GW. A series of numerical simulations in a deforming periodic domain have been performed to study the effect of the Reynolds number and possible numerical or sampling limitations of the results. To capture the natural behaviors of both large- and small-scale motions faithfully, simulation parameters are chosen to ensure that, at all times, the large scales are sufficiently well sampled along the shortest side of the solution domain, while the small scales are sufficiently well resolved along the direction of coarsest grid spacing. A pre-simulation is first carried out in order to produce a state of fully developed isotropic turbulent flow before strain is applied. The highest grid resolution for simulations focusing on the velocity field is 4096^3, and up to 2048^3 grids have been used to date for the turbulent mixing studies. During the application of strain the small scales of both the velocity and scalar fields become strongly anisotropic, but in contrast to the large scales they return to isotropy quickly when strain is removed. For the case of highest pre-strain Reynolds number (95 based on the Taylor scale), there is clear evidence of the characteristic changes in the spectral shapes for the velocity components that

AW reported in their higher Reynolds number experiments. Detailed analyses of the evolution of axisymmetric spectra (Mininni *et al.*, 2012) in the simulations show that this feature is the result of a strong contrast in rates of return to isotropy between the large scales and the small scales, with this contrast being stronger at higher Reynolds number. Similarly, the evolution of the scalar spectrum during the application of strain is in agreement with rapid distortion theory, and during relaxation the change in shape of the scalar spectrum agrees well with the results of GW.

As discussed previously, a key attribute of turbulent flows is their ability to provide efficient mixing. Often, the substance or property being mixed is of such a low concentration that it does not affect the fluid motion — such substances or properties are called passive scalars, examples of which include dye in water and small temperature fluctuations in air (Warhaft, 2000). Aside from the Reynolds number, an important non-dimensional parameter in passive scalar mixing is the Schmidt number $Sc = \nu/D$, where ν is the kinematic viscosity of the fluid and D is the molecular diffusivity of the scalar. The Schmidt number can vary widely depending on the application: mixing in liquid metals occurs at $Sc = \mathcal{O}(0.01)$, typical gas-phase mixing occurs at $Sc = \mathcal{O}(1)$, and dyes mixing in liquids occurs at $Sc = \mathcal{O}(1000)$ (Gotoh & Yeung, 2013). Mixing at $Sc \gg 1$ is challenging for both experiments and computations due to the distinctly different nature of the scalar field when compared to $Sc \lesssim 1$ mixing. When $Sc \gg 1$, the smallest scale in the scalar field is the Batchelor scale $\eta_B = \eta Sc^{-1/2}$, which is $\sqrt{Sc}$ times smaller than the Kolmogorov scale η (a measure of the smallest length scales in the velocity field). The stringent resolution requirements of the Batchelor scales in $Sc \gg 1$ mixing are difficult to meet in both DNS and experiment. Often, for a fixed resolution (i.e., a given grid size in numerical simulations), one is forced to limit either the Reynolds number or Schmidt number in order to maintain adequate resolution of both the velocity and scalar fields (Donzis & Yeung, 2010).

The dramatic increase in the computational cost of $Sc \gg 1$ simulations due to the

strict resolution requirements of the Batchelor scales also forces one to consider how to efficiently carry out the computations. Traditionally, DNS of turbulent mixing in homogeneous isotropic turbulence in a periodic domain employ Fourier pseudo-spectral (FPS) methods, and compute the velocity field and scalar field on the same grid. At high Schmidt number, this implies that the velocity field is computed at very high resolution — much higher than might be required to obtain accurate results. Furthermore, pseudo-spectral methods implemented in multiple dimensions require expensive memory transpositions, which reduce parallel efficiency at large problem sizes. To improve efficiency, one may ask: (i) given that the scalar equation is local in space and does not require the solution of a nonlocal (Poisson) equation like the velocity field, what is the best method to accurately and efficiently compute the scalar field; (ii) can the separation of scales between the scalar and velocity fields at high Sc be taken advantage of to save resources by computing the velocity field on a coarser grid than is required for the scalar; and (iii) can the one-way coupling between the velocity and passive scalar be exploited in the design of a parallel code that computes their combined evolution?

The second major objective of this thesis is to develop a parallel algorithm that meets the considerations above for problems of at least 8192^3 (0.5 trillion) grid points for the scalar field, which is comparable to recent large-scale DNS of a turbulent velocity field (Yeung *et al.*, 2015; Ishihara *et al.*, 2016). For consideration (i), note that the advection-diffusion equation for the passive scalar is local in space, and therefore does not require the same communication-intensive global approaches used for the computation of the incompressible velocity field, e.g., FPS (Mininni, 2011). An attractive alternative is to use compact finite difference schemes, which can achieve high accuracy with significantly reduced communication requirements (Mahesh, 1998), and can be implemented in a manner that does not require memory transposes (Nihei & Ishii, 2003). Since compact finite difference schemes only require thin ghost lay-

ers between adjacent sub-domains (Lele, 1992), the volume of message traffic is low. For these reasons, and because of the need to obtain first and second derivatives for the advection-diffusion equation, the eighth-order combined compact finite difference (CCD) scheme of Mahesh (1998) is used. The scheme computes both first and second derivatives and was used by Gotoh *et al.* (Gotoh *et al.*, 2012) for the scalar field. Consideration (ii) is addressed by adopting a dual-resolution approach (Gotoh *et al.*, 2012; Brethouwer *et al.*, 2003), where the scalar field is computed on a fine grid of size N_θ^3 that resolves the Batchelor scale, while the velocity field is computed on a coarse grid of size N_v^3 that resolves the Kolmogorov scale. When the velocity field is needed on the fine grid it is interpolated from the coarse grid values, which is feasible because the velocity field is very smooth with respect to the fine scalar grid.

For consideration (iii) noted previously, the design of an efficient parallel interface that provides the required linkage between the velocity and scalar fields computed on different grids and with different numerical methods is nontrivial, especially at petascale problem sizes. Since multi-cored processors are in widespread use, the codes employ a hybrid approach in which communication is handled by the Message Passing Interface (MPI) among so-called MPI processes, each being capable of spawning a number of shared-memory OpenMP threads. In production simulations N_θ/N_v is typically 4 or 8, which means the velocity and scalar field problem sizes differ by a ratio as large as $4^3 = 64$ or $8^3 = 512$. This implies that the overall computational requirements are driven by what is needed for the scalar field on the finer grid. Since the code essentially solves two sets of equations simultaneously it is convenient to separate the global communicator (`MPI_COMM_WORLD`) into two distinct communicators of size M_v and M_θ, which are easily matched to each problem size, i.e., $M_\theta/M_v \propto N_\theta^3/N_v^3$. The velocity and scalar fields are computed in their respective disjoint communicators, while the one-way transfer of velocity information from the coarser grid to the finer grid occurs through the global communicator. Because this transfer is one way it can

be handled by non-blocking point-to-point communication and overlapped with the more time-consuming operations for the scalar.

When implementing codes of this nature, it is important to note trends or changes in the HPC landscape. In particular, one recent trend is that computing architectures are becoming heterogeneous, where a typical host central processing unit (CPU) is paired with an accelerator device (often a graphical processing unit, or GPU) to increase computational throughput. The incorporation of accelerators in large machines is economical due to their low power consumption per floating point operation (Keckler *et al.*, 2011), but their increased complexity places a burden on the application programmer. For example, in current petascale heterogeneous machines like Titan at Oak Ridge National Laboratory (ORNL), TN, the host and device memory spaces are distinct, which implies that any data required for computations on the accelerator must be copied to the device, before the results can be computed and copied back to the host. Because the memory bandwidth between the host and device is often limited (Fujii *et al.*, 2013), one must minimize data movement in the application to obtain good performance. One must also take care that the computational work offloaded to the device can benefit from the massive multithreading typically provided by accelerators. The algorithm developed in this thesis for high Sc turbulent mixing is a good candidate for acceleration, primarily due to its moderate communication cost, and its loops which can take advantage of massive multithreading.

A specific coding objective of this thesis is to port and optimize the CPU-based code for high Sc turbulent mixing to the Cray XK7 GPU architecture, primarily on Titan, the 27 petaflops supercomputer at ORNL. To avoid extensive code rewrites, a directive-based approach is used, which considerably simplifies programming for heterogeneous systems. Two major options for directive-based programming are OpenACC and OpenMP, which both allow the user to specify how the workload is shared between the host and device by inserting directives (i.e., constructs, or prag-

mas) to a pre-existing code. While programming with OpenACC or OpenMP occurs at a higher level than hardware-specific languages like CUDA or other accelerator-programming languages like OpenCL, there are still many opportunities to obtain significant speedups. Specifically, both OpenACC and OpenMP allow the programmer to control data locality and therefore minimize data movement. Also, when the algorithm permits, communication and computation can be overlapped if the host can interact asynchronously with the device. While OpenACC has had asynchronous capabilities since its inception, the latest OpenMP 4.5 standard has added the essential task and asynchronous clauses (i.e., `DEPEND` and `NOWAIT`) to the device (i.e., `TARGET`) constructs, which make asynchronous algorithms possible. In the Fortran code developed in this thesis, OpenMP is the primary programming model used for acceleration. In particular, the latest OpenMP 4.5 features for asynchronous operations supported by the Cray Compiler Environment 8.5.7 are used extensively. The code achieves a speedup of 2.7X compared to CPU-only execution at the largest problem size, which requires 8192 XK7 nodes on Titan for the scalar. The techniques adopted — especially the use of OpenMP 4.5 on thousands of nodes — are quite new, and current trends in the HPC landscape suggest OpenMP will be increasingly well supported on GPU-dominated platforms of theoretical peak in the order of several hundreds of petaflops to arrive in the near future, and likely exascale architectures.

Because $Sc \gg 1$ mixing is so challenging for both experiment and computation, there are many fundamental questions regarding the scalar field that have not been adequately resolved. Perhaps one of the most important of these questions concerns the isotropy (or universality) of the passive scalar field. Beginning with Kolmogorov (1941), a key hypothesis driving turbulence research is that at high Re the small-scale structure of the turbulent velocity field is isotropic, i.e., universal, and is independent of the large-scale flow. Obukhov (1949) and Corrsin (1951) extended Kolmogorov's hypotheses for the velocity field to the scalar field; however, many experimental and

numerical efforts have found that the scalar field does not exhibit local isotropy, even at very high Re. The violation of local isotropy is often reported in the measurement of the scalar derivative skewness, which should be zero if local isotropy holds, but is found to be $\mathcal{O}(1)$ for mild temperature fluctuations in air (Sreenivasan, 1991) — a scalar with $Sc = \mathcal{O}(1)$. Previous simulations (Yeung *et al.*, 2004) have suggested an approach to local isotropy with increasing Sc, but have been limited to relatively low Re. Simulations at higher Re (Donzis & Yeung, 2010), on the other hand, have been limited to moderate values of Sc, such that the separation of scales between the velocity and scalar fields is limited.

Also of interest at high Re and high Sc are the scaling properties of the scalar field. In spectral space the primary target is the shape of the scalar spectrum, which presents the scale-dependent contribution to the scalar variance. At high Sc, the presence of sub-Kolmogorov fluctuations in the scalar field — the so-called Batchelor scales described previously — implies that there is significant spectral content in the scalar spectrum beyond the dissipation range for the velocity field. The range of scales beyond the viscous cutoff for the velocity field, but before significant dissipation occurs in the scalar field, is referred to as the viscous-convective range, for which both Batchelor (1959) and Kraichnan (1974) theorized a k^{-1} scaling (referred to as Batchelor scaling). Experimental support for Batchelor scaling is elusive (Miller & Dimotakis, 1996; Warhaft, 2000), but computations have provided increasing support for such scaling (Donzis *et al.*, 2010; Gotoh *et al.*, 2014). While Batchelor scaling pertains to the spectral representation of the scalar fluctuations, there are many scaling properties in physical space that are also of interest. Here an exact result is attributed to Yaglom (1949) for the mixed third-order structure function between the velocity and scalar field. High Reynolds number data supports Yaglom's relation in the inertial-convective range of scales well (Yeung & Donzis, 2005), but theory suggests that the relation should continue to hold in the viscous-convective range

which develops at high Sc (Gotoh & Yeung, 2013). Previous DNS at lower Re and moderate Sc support such scaling (Yeung *et al.*, 2002; Iyer & Yeung, 2014), which should be investigated further at higher Re and higher Sc.

Following the development of numerical algorithms capable of simulating high Re and high Sc mixing efficiently, simulations are conducted to investigate the aforementioned (and more) science questions pertaining to the passive scalar field mixed in a turbulent flow. The third major objective of this thesis is therefore to systematically investigate the dependence of scalar statistics on the Schmidt number. To do this a DNS database at a fixed Reynolds number ($R_\lambda \approx 140$, with λ being the Taylor scale) is generated covering a range of Schmidt numbers. The Schmidt number is increased from $Sc = 4$ to $Sc = 512$, with the highest Schmidt number being comparable to salinity mixing in the ocean. The resolutions required for these simulations vary from 1024^3 at the lower Schmidt numbers to 8192^3 for $Sc = 512$. Good resolution is maintained in the simulations to study small-scale statistics, and in many instances a given Schmidt number is run with different numerical configurations (e.g., different grid resolutions or different restrictions on the time step) to assess the impact of the numerics on the results. The simulations were carried out on the petascale machines Blue Waters and Titan, under PRAC and INCITE allocations, respectively.

In summary, the major objectives of this thesis are as follows:

1. **To simulate turbulence and turbulent mixing under irrotational axisymmetric contraction and subsequent relaxation.**

   Simulations of turbulence subjected to axisymmetric contraction will be conducted, using a methodology to model the spatially-varying strain rate from the AW and GW wind tunnel experiments directly in the time-evolving DNS. Analysis will focus on the evolution of the 1-D component velocity spectra to see if DNS can reproduce and explain the AW results. Passive scalar mixing will also be simulated in the same configurations, focusing on the mixing of scalar fluctuations generated

by mean scalar gradients in the streamwise and transverse directions. Comparisons to the results of GW will be made.

2. **Develop efficient algorithms for DNS of high Re and high Sc mixing.**

   To simulate turbulent mixing at high Re and high Sc, the computational methods developed by Gotoh *et al.* (2012) will be extended to scale to a larger number of processors. Specifically, the processors solving for the velocity field and scalar field will be decoupled, which will enable simulations for arbitrary $Sc \gg 1$. The objective is to develop an approach that scales to large process counts ($\mathcal{O}(10^5)$ cores) in both homogeneous and heterogeneous computing environments.

3. **Study passive scalar mixing in isotropic turbulence with $Sc \gg 1$ and Re sufficiently high for a narrow inertial range.**

   Using the new computational algorithms, simulations will be conducted to study turbulent mixing at a Reynolds number sufficiently high to support a narrow inertial range ($R_\lambda \approx 140$) and high Schmidt number (up to $Sc = 512$), while maintaining good small-scale resolution. Simulations will be conducted to address many open questions about passive scalar mixing, including the status of local isotropy and the form of the scalar spectrum.

CHAPTER II

TURBULENCE UNDER AXISYMMETRIC CONTRACTION

The evolution of a turbulent velocity field subjected to irrotational mean strain in the form of axisymmetric contraction is a canonical representation of flows in many engineering devices, e.g., flows through nozzles and diffusers. While this configuration has been studied extensively in both experimental (Choi & Lumley, 2001) and numerical (Lee & Reynolds, 1985) settings, the flows were typically of low Reynolds number. The experiments of Ayyalasomayajula & Warhaft (2006) showed that at higher Reynolds number the picture is more complex. Their results suggest that strong nonlinear interactions dominate the evolution of the post-contraction anisotropic flow, with a key result being that the shapes of the component velocity spectra evolve in a non-trivial manner. There is a need to understand the underlying physical mechanisms contributing to the observations of AW, and DNS is perfectly suited for this task. This chapter presents a numerical study of turbulence subjected to axisymmetric contraction, with an emphasis on comparing the results to the experiments of AW. The complete description of the flow field provided by DNS is used to understand the evolution of the spectra in a manner that is difficult to measure experimentally.

The contents of this chapter are available in the published work described in Appendix A (Clay & Yeung, 2016), and were also presented at the 68^{th} Annual Meeting of the Division of Fluid Dynamics of the American Physical Society as described in Appendix B. Beginning in §2.1, the numerical method, the time-dependent strain rate used to model the wind tunnel contraction of AW, and the evolution equations for spectra are presented. In §2.2, details are given on the pre-simulation approach along with a listing of several simulations conducted for both physical (Reynolds number) and numerical (domain size and grid resolution) reasons. In §2.3, the response of

initially isotropic turbulence to the axisymmetric contraction is studied, and in §2.4 results are presented for the relaxation phase that begins when the strain rate is turned off. A number of one-dimensional spectra are shown which display features similar to those reported by AW. Finally, conclusions are summarized in §2.5.

2.1 Mathematical formulation and numerical approach

The use of a solution domain that deforms according to an axisymmetric time-dependent strain rate, which is in turn constructed to model a laboratory turbulent flow with a spatially-evolving cross-section, requires some special care in both the conduct of the simulation and the subsequent data analysis, as discussed below.

2.1.1 Solution algorithm in a deforming, anisotropic domain

Rogallo (1981) introduced the coordinate transformation $\xi_i = B_{ij}(t)x_j$, in which the deforming coordinate system ξ_i is related to the laboratory coordinates x_i through the metric tensor $B_{ij}(t)$. Summation over repeated Latin indices is implied throughout the text. The deforming coordinates move with the mean flow, according to

$$\frac{dB_{ij}}{dt} + B_{ik}\frac{\partial \langle U_k \rangle}{\partial x_j} = 0 \ , \tag{2.1}$$

where $\langle U_i \rangle$ is the mean velocity. In this coordinate system the turbulence is homogeneous, and periodic boundary conditions are applicable provided the mean velocity gradients are uniform in space. The continuity and momentum equations become

$$B_{ji}\frac{\partial u_i}{\partial \xi_j} = 0 \tag{2.2}$$

$$\frac{\partial u_i}{\partial t} + u_j\frac{\partial \langle U_i \rangle}{\partial x_j} + B_{kj}\frac{\partial u_i u_j}{\partial \xi_k} = -B_{ji}\frac{1}{\rho}\frac{\partial p}{\partial \xi_j} + \nu B_{kj}B_{lj}\frac{\partial^2 u_i}{\partial \xi_k \partial \xi_l} \ , \tag{2.3}$$

where u_i is the fluctuating velocity, p is the fluctuating pressure, ρ is the density, and ν is the kinematic viscosity. Nonlinear terms are evaluated using a Fourier pseudo-spectral method, and the numerical solution in Fourier space is advanced in time using a second-order predictor-corrector scheme. Aliasing errors are mitigated by using truncation and grid shifting in wavenumber space (Rogallo, 1981).

For irrotational axisymmetric contraction, the mean deformation tensor is given by (Lee & Reynolds, 1985)

$$\frac{\partial \langle U_i \rangle}{\partial x_j} = \frac{2}{\sqrt{3}} S(t) \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix} \tag{2.4}$$

with $S(t) \geq 0$. The domain is orthogonal at all times, such that only the diagonal elements of $B_{ij}(t)$ are nonzero. Solving (2.1) leads to (for $\alpha = 1, 2, 3$, no summation)

$$B_{\alpha\alpha}(t) = B^0_{\alpha\alpha} e^{-f_\alpha(t)} \ , \tag{2.5}$$

where $B^0_{\alpha\alpha} \equiv B_{\alpha\alpha}(0)$, and $\exp[f_\alpha(t)]$ is the total strain in each direction with

$$f_\alpha(t) = \int_0^t \frac{\partial \langle U_\alpha \rangle}{\partial x_\alpha} \, d\tau \ . \tag{2.6}$$

Figure 2.1 shows a schematic of how the domain is deformed during the application of strain. At any time t, the length of the domain on each side given by $L_\alpha = 2\pi / B_{\alpha\alpha}(t)$ varies as $L_\alpha(t) = L^0_\alpha \exp[f_\alpha(t)]$ where L^0_α is the initial length. While the number of grid points is fixed, the grid spacings and hence also resolution in each direction vary. Correspondingly, the wavenumbers represented in each direction of the domain are given by $k_\alpha(n_\alpha, t) = n_\alpha B_{\alpha\alpha}(t)$, where $n_\alpha = -N_\alpha/2 + 1, \cdots, N_\alpha/2$, and N_α is the number of grid points in the x_α direction. The maximum resolved wavenumber in each direction after truncation is $k_{\max,\alpha}(t) = \sqrt{2} N_\alpha B_{\alpha\alpha}(t)/3$. As the simulation

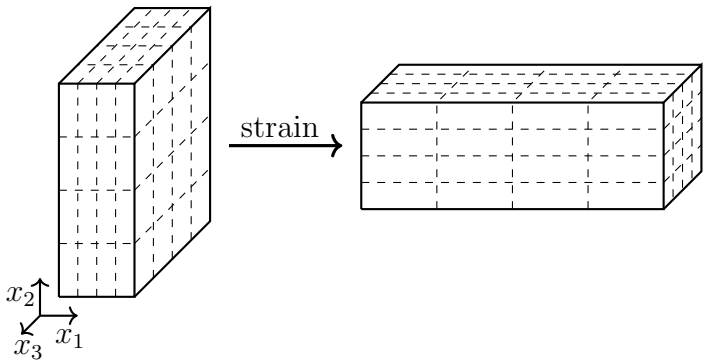

Figure 2.1: Grid deformation under axisymmetric contraction with a total elongation of 4 in the x_1 direction. The grid metrics in the x_2 and x_3 directions are equal.

proceeds and the domain is deformed from its original shape, it is necessary to check that the large scales continue to be well contained within the domain in all directions, while the small scales likewise remain well resolved (Gualtieri & Meneveau, 2010).

2.1.2 A time-dependent strain as a model for experiment

In most wind tunnel experiments the change in cross-section is gradual, which results in a smooth variation of the mean velocity with distance downstream along the wind tunnel centerline (where measurements for nearly homogeneous turbulence are made). The mean strain rate along the wind tunnel centerline is thus readily established as a function of the spatial coordinates. However, as a fluid element passes through the wind tunnel, the spatially-varying mean strain rate is experienced in a time-dependent manner. To model the mean strain rates from experimental wind tunnels in the time-dependent DNS, a spatially uniform, but time-dependent mean strain rate is applied which corresponds to the local value of the mean strain rate that a fluid element experiences as it passes through the wind tunnel (Pearson, 1959). This is accomplished by introducing a convective time t as the time taken for a fluid element traveling with the mean flow to reach a position x from a reference location x_a. (Here x alone, or x_a etc., without tensor subscripts shall refer to distances measured along

the wind tunnel centerline.) The relation between t and x is

$$t = \int_{x_a}^{x} \frac{d\xi}{\langle U_1(\xi) \rangle} \ . \tag{2.7}$$

According to (2.5-2.6), by time t the grid metric in the extensional direction will be

$$B_{11}(t) = B_{11}^0 \exp\left[- \int_0^t \frac{\partial \langle U_1 \rangle}{\partial x_1} \, d\tau \right] \ . \tag{2.8}$$

The integral in (2.8) can be evaluated through a change of variables, $d\tau = dx/\langle U_1(x) \rangle$ (suggested by (2.7)). After some simplifications, the metric in x_1 is given by

$$B_{11}(t) = B_{11}^0 \frac{\langle U_1(x_a) \rangle}{\langle U_1(x) \rangle} \ . \tag{2.9}$$

Since the applied mean strain is volume-preserving, the product of the three diagonal metric factors $B_{11}B_{22}B_{33}$ is fixed, while axisymmetry requires $B_{22} = B_{33}$. As a result, the change in the transverse grid metrics is given by

$$\frac{B_{22}(t)}{B_{22}^0} = \frac{B_{33}(t)}{B_{33}^0} = \sqrt{\frac{\langle U_1(x) \rangle}{\langle U_1(x_a) \rangle}} \ . \tag{2.10}$$

Since in the simulation the turbulence is advanced in time instead of space, at each time step from t_n to $t_n + \Delta t$, it is necessary to solve (2.7) for a new spatial location x, so that the new metric factors can be evaluated according to (2.9-2.10). In addition, at every time step the metric factors are used to form the viscous integrating factors in Rogallo's algorithm. The solution to (2.7) and the calculation of the integrating factors are carried out using `QUADPACK` (Piessens *et al.*, 1983).

In the experiments of AW, the mean streamwise velocity $\langle U_1(x) \rangle$ along the centerline of the wind tunnel is well described by an error function, which yields a Gaussian profile for the mean extensional strain (see figure 2 of AW). Consider the functional

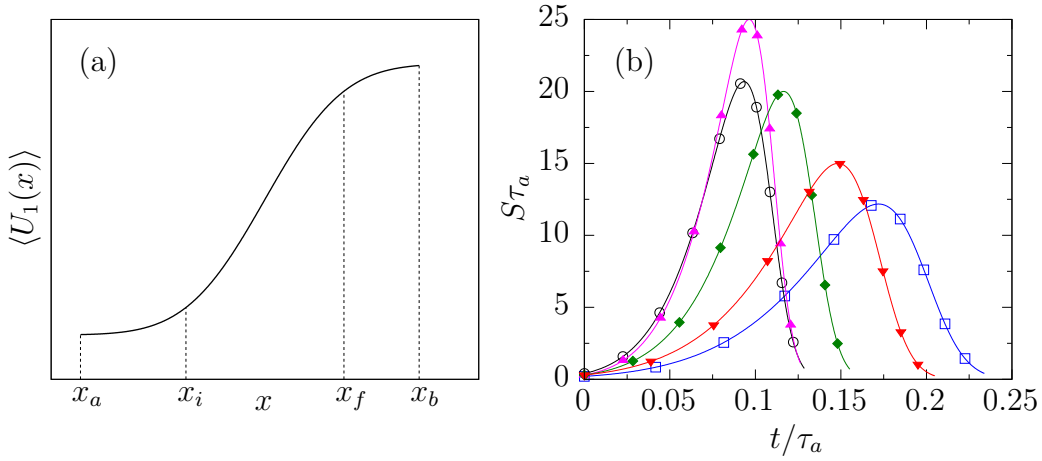

Figure 2.2: Schematic of (a) mean velocity profile in experiments, and (b) examples of non-dimensional strain rate $S\tau_a$ as function of convective time in experiments and DNS. In (b): ○ for AW $Re_\lambda = 260$ experiment, □ for AW $Re_\lambda = 40$ experiment, ▲ for $S_0^* = 25$ numerical, ◆ for $S_0^* = 20$ numerical, and ▼ for $S_0^* = 15$ numerical.

form

$$\langle U_1(x)\rangle = a\,\mathrm{erf}(bx) + c\ , \quad x \in [x_a, x_b]\ , \tag{2.11}$$

where the five parameters a, b, c, x_a, and x_b are chosen so the mean strain closely matches the experiments in a non-dimensional sense, and the origin of the x-axis has been placed at the location of maximum mean velocity gradient. In the experiments, some appreciable strain also existed immediately upstream and downstream of the physical beginning and ending locations of the contraction. To incorporate this feature in the DNS, two intermediate locations x_i and x_f are specified satisfying $x_a < x_i < x_f < x_b$ (x_i and x_f correspond to the vertical lines in figure 2 of AW). Since these two locations are about equally far from the location of maximum strain rate it suffices to take $x_i = -x_f$. The straining "period" in the DNS is considered to be the entire phase of the simulation during which there is nonzero mean strain, corresponding to the physical distance between x_a and x_b. Figure 2.2(a) presents a sketch of an error function mean velocity profile with these locations marked for later reference. While x_i and x_f are used to specify the mean velocity variation, pre- and post-contraction

statistics are reported at x_a and x_b, respectively.

For a systematic procedure for choosing the five curve-fit parameters noted above, five constraints are needed to control both the spatial spread of the profile and its maximum velocity gradient (which is proportional to the product ab). First, since the value of c does not affect the strain rate, it is arbitrary provided it is large enough to ensure that $\langle U_1(x)\rangle > 0$ for all x. Second, the velocity ratio $\langle U_1(x_f)\rangle/\langle U_1(x_i)\rangle$ is specified at a value close to that from the experiment. Third, the strain rate S at $x = 0$, denoted by S_0, corresponds to a non-dimensional strain S_0^* formed from S_0 and a large-eddy time scale of the turbulence before strain is applied. Specifically, define $S_0^* = S_0\tau_a$, where $\tau_a = (2K/\langle\epsilon\rangle)_a$, with K and $\langle\epsilon\rangle$ being the turbulence kinetic energy and the mean energy dissipation rate, respectively. Fourth, the starting location x_a is chosen to give a desired non-dimensional strain rate at x_a based on the experiments. Finally, the ending location x_b is chosen to achieve a desired velocity ratio $\langle U_1(x_b)\rangle/\langle U_1(x_a)\rangle$, i.e., deformation, for the entire straining period.

Figure 2.2(b) shows a qualitative comparison of the non-dimensional strain rate profiles obtained numerically from 3 values of S_0^* with experimental data derived from AW at two values of the pre-strain Reynolds number. Experimental data in this figure are obtained first by applying curve fits of a form similar to (2.11) and then differentiating. The non-dimensional strain rates are plotted at convective times t in the laboratory facility obtained using (2.7) and normalized by large-eddy turnover times obtained from table 1 of AW. The numerical strain rates are obtained using the aforementioned procedure with $c = 10$, $bx_f = 0.9$, a velocity ratio $\langle U_1(x_f)\rangle/\langle U_1(x_i)\rangle = 2.85$, a starting non-dimensional strain rate of 0.3 at x_a, and an overall velocity ratio $\langle U_1(x_b)\rangle/\langle U_1(x_a)\rangle = 4$. The resulting strain rate profiles are similar to those in the experiments, suggesting that the procedures described here model the mean velocity variation in the wind tunnel contraction well.

2.1.3 Spectral evolution and rapid-distortion theory

To understand the dynamical processes underlying the change in shape of the energy spectrum under applied strain, it is helpful to compute various terms in the spectral evolution equations. In isotropic turbulence, computation of the nonlinear transfer spectrum of the kinetic energy (e.g. Domaradzki & Rogallo (1990); Yeung *et al.* (1995)) in spherical wavenumber shells is sufficient. However, this work must also consider pressure-strain correlations and systematic anisotropy, which requires individual components of spectra in one- and two-dimensional partitions of wavenumber space. In addition, the mean strain distorts the wavenumbers on a deforming domain, leading to transport of the spectrum in Fourier space (Pope, 2000).

In Fourier space, the fluctuating velocity on a deforming domain evolves by

$$\frac{d\hat{u}_i(\mathbf{k})}{dt} = -\hat{u}_j(\mathbf{k})\frac{\partial\langle U_i\rangle}{\partial x_j} - \mathrm{i}k_i\hat{p}(\mathbf{k}) - G_i(\mathbf{k}) - \nu k^2\hat{u}_i(\mathbf{k}) \ , \tag{2.12}$$

where the metric factors in (2.3) are expressed through time-dependence of the wavevector $\mathbf{k}$, $\hat{p}(\mathbf{k})$ is the Fourier coefficient of the fluctuating pressure, $G_i(\mathbf{k}) = \mathrm{i}k_j\widehat{u_iu_j}$, and $\mathrm{i} = \sqrt{-1}$. It is important to distinguish between rapid pressure and slow pressure, which respond to the mean flow and the velocity fluctuations respectively. The pressure is written as $\hat{p} = \hat{p}^r + \hat{p}^s$, with both components obtained from well-known Poisson equations. The spectral covariance $E_{ij}(\mathbf{k}) \equiv \langle\hat{u}_i^*(\mathbf{k})\hat{u}_j(\mathbf{k})\rangle$ (where superscript $*$ denotes complex conjugate) is complex-valued, although because mean shear is absent only the diagonal (real) components are relevant. Its equation is

$$\frac{dE_{ij}(\mathbf{k})}{dt} = P_{ij}(\mathbf{k}) + \Pi_{ij}^r(\mathbf{k}) + \Pi_{ij}^s(\mathbf{k}) + T_{ij}(\mathbf{k}) - D_{ij}(\mathbf{k}) \ . \tag{2.13}$$

where terms on the right-hand side represent, respectively, production due to the mean velocity gradient, redistribution due to rapid pressure and slow pressure, spectral

transfer due to nonlinear terms, and dissipation due to viscosity. Specifically, these terms are:

$$P_{ij}(\mathbf{k}) = -\frac{\partial \langle U_j \rangle}{\partial x_m} E_{im}(\mathbf{k}) - \frac{\partial \langle U_i \rangle}{\partial x_m} E^*_{jm}(\mathbf{k}) \ , \tag{2.14}$$

$$\Pi^r_{ij}(\mathbf{k}) = \mathrm{i}k_i \langle \hat{u}^*_j(\mathbf{k}) \hat{p}^r(\mathbf{k}) \rangle^* - \mathrm{i}k_j \langle \hat{u}^*_i(\mathbf{k}) \hat{p}^r(\mathbf{k}) \rangle \ , \tag{2.15}$$

$$\Pi^s_{ij}(\mathbf{k}) = \mathrm{i}k_i \langle \hat{u}^*_j(\mathbf{k}) \hat{p}^s(\mathbf{k}) \rangle^* - \mathrm{i}k_j \langle \hat{u}^*_i(\mathbf{k}) \hat{p}^s(\mathbf{k}) \rangle \ , \tag{2.16}$$

$$T_{ij}(\mathbf{k}) = -\left[\langle \hat{u}^*_i(\mathbf{k}) G_j(\mathbf{k}) \rangle + \left\langle \hat{u}^*_j(\mathbf{k}) G_i(\mathbf{k}) \right\rangle^* \right] \ , \tag{2.17}$$

$$D_{ij}(\mathbf{k}) = 2\nu k^2 E_{ij}(\mathbf{k}) \ . \tag{2.18}$$

The angled brackets in these equations represent averaging over multiple realizations, i.e., ensemble averaging, which is performed for the majority of the simulations detailed later. When axisymmetric contraction is applied the production term is negative for $E_{11}(\mathbf{k})$, but positive for $E_{22}(\mathbf{k})$ and $E_{33}(\mathbf{k})$, thus causing anisotropy directly, especially at the large scales. The pressure-strain term exchanges energy among the diagonal components $E_{\alpha\alpha}(\mathbf{k})$ and is traceless due to incompressibility, for both rapid and slow pressures. The rapid pressure is present only while strain is applied, while slow pressure is also important during relaxation. Since nonlinear-transfer redistributes energy in Fourier space, each component of $T_{ij}(\mathbf{k})$ integrates to zero over wavenumber space. Integrating (2.13) over all wavenumbers gives the Reynolds stress transport equation for homogeneous turbulence, which is

$$\frac{d\langle u_i u_j \rangle}{dt} = -\langle u_j u_k \rangle \frac{\partial \langle U_i \rangle}{\partial x_k} - \langle u_i u_k \rangle \frac{\partial \langle U_j \rangle}{\partial x_k} + \frac{2}{\rho} \langle p^r s_{ij} \rangle + \frac{2}{\rho} \langle p^s s_{ij} \rangle - 2\nu \left\langle \frac{\partial u_i}{\partial x_k} \frac{\partial u_j}{\partial x_k} \right\rangle \ , \tag{2.19}$$

where s_{ij} is the fluctuating strain rate.

In (2.13), only the production and rapid pressure-strain terms depend directly on the mean strain rate. As a result, if the strain rate is very strong, i.e., very rapid compared to the timescales of the turbulence, this equation can be simplified

by retaining only $P_{ij}(\mathbf{k})$ and $\Pi^r_{ij}(\mathbf{k})$, while neglecting viscous dissipation and the nonlinear effects of slow pressure and spectral transfer. The behavior of the energy spectrum tensor can then be described by inviscid rapid distortion theory (RDT) (Savill, 1987; Townsend, 1976). Although in practice the strain rates in experiments and simulations are necessarily finite, comparisons with RDT theory are still useful for a better understanding.

The use of a deforming domain, and hence a time-dependent set of wavevectors in the simulations, requires some care when interpreting spectra. It would be useful to relate the 1-D spectrum $E_{\alpha\alpha}(k_\beta)$ to its equivalent representation $E^0_{\alpha\alpha}(k^0_\beta)$ as a function of the pre-strain wavenumbers k^0_β. This would clearly show how a set of modes (those perpendicular to an initial k_1) are affected by the strain. The integrals of these spectra over the corresponding wavenumbers k_β and k^0_β are both equal to $\langle u^2_\alpha \rangle$. At any time t during the straining period, the current and pre-strain wavenumbers are related by

$$k_\beta(t) = k^0_\beta B_{\beta\beta}(t)/B^0_{\beta\beta} \ , \tag{2.20}$$

where $B^0_{\beta\beta}/B_{\beta\beta}(t)$ is the total deformation in the x_β direction. A change of variables between the integrals in k_β and k^0_β then gives the relation

$$E^0_{\alpha\alpha}(k^0_\beta) \equiv E_{\alpha\alpha}(k_\beta)B_{\beta\beta}(t)/B^0_{\beta\beta} \ . \tag{2.21}$$

Following AW, most of the 1-D spectral results are presented in terms of the pre-strain wavenumbers during the application of strain, and in terms of the post-strain wavenumbers (which are fixed) during the relaxation period.

Statistics in axisymmetric turbulence are rotationally symmetric about a preferred direction $\boldsymbol{\lambda}$ (Batchelor, 1946), which in this study is the extensional direction. Figure 2.3 shows a decomposition of Fourier space motivated by this rotational symmetry, such that spectral quantities are expressed as functions of k_1 and the wavenumber

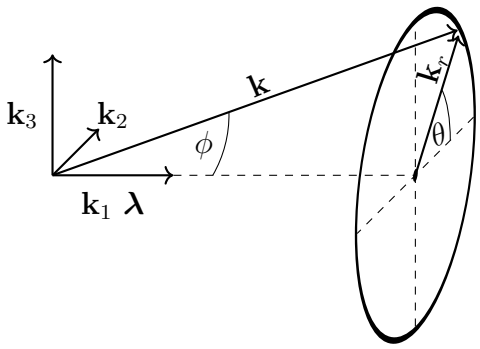

Figure 2.3: Ring decomposition of wavenumber space for axisymmetric turbulence. The axis of axisymmetry $\boldsymbol{\lambda}$ is in the $\mathbf{k}_1$ direction, and the ring is perpendicular to $\boldsymbol{\lambda}$. The longitude with respect to $\mathbf{k}_2$ is θ and the colatitude with respect to $\boldsymbol{\lambda}$ is ϕ.

magnitude in the cross-sectional plane $k_r = \sqrt{k_2^2 + k_3^2}$. The axisymmetry can also be expressed by $k_r = k \sin\phi$, where $k = \sqrt{\mathbf{k} \cdot \mathbf{k}}$, and $0 \leq \phi \leq \pi$ is the colatitude with respect to $\boldsymbol{\lambda}$. The spectral covariance $E_{ij}(\mathbf{k})$ defined earlier is a discrete version of the velocity spectrum tensor $\Phi_{ij}(\mathbf{k})$ (whose integral in wavenumber space gives $\langle u_i u_j \rangle$). The axisymmetric spectrum tensor is defined as

$$A_{ij}(k_1, k_r) = \int_0^{2\pi} \Phi_{ij}(\mathbf{k}) k_r \, d\theta \ , \tag{2.22}$$

from which the 1-D spectra can be recovered by

$$E_{ij}(k_1) = 2 \int_0^{\infty} A_{ij}(k_1, k_r) \, dk_r \ , \tag{2.23}$$

where the factor of 2 is used to collect contributions from both positive and negative values of k_1. Axisymmetric representations of the spectra are useful in studies of rotating (Clark di Leoni *et al.*, 2014) and stably stratified turbulent flows (Godeferd & Staquet, 2003). Evolution equations for the axisymmetric spectra and 1-D spectra are readily obtained by integration of (2.13) over rings and planes in wavenumber space, respectively. For isotropic turbulence, the contours of the axisymmetric energy spectrum $E_A(k_1, k_r) \equiv \frac{1}{2} A_{ii}(k_1, k_r)$ multiplied by $1/\sin\phi$ are circles in the

(k_1, k_r) plane (Mininni *et al.*, 2012). In practice, the axisymmetric spectra are formed by summing over Fourier modes residing in rings of finite thickness. Because the computational space is Cartesian, the distribution of modes in rings at small k_r is significantly uneven, which can lead to some noise in contours of the axisymmetric spectra. A node-density correction factor is applied in a manner similar to those used for three-dimensional spectra collected into discrete spherical shells in wavenumber space (Eswaran & Pope, 1988).

2.2 Pre-simulation and the choice of numerical parameters

In the simulations the effects of strain must be applied to a velocity field that is physically well-developed, well-sampled, and well-resolved. To produce such an initial state, a "pre-simulation" is carried out for decaying isotropic turbulence evolving from a Gaussian velocity field with a specified energy spectrum. Usually, the mean strain rate is turned on when the turbulence shows clear evidence of a power-law decay in its kinetic energy, and of non-Gaussianity in the statistics of velocity gradient fluctuations. The initial energy spectrum chosen is (Pope, 2000, pp. 232–234)

$$E(k) = C_K \langle\epsilon\rangle^{2/3} k^{-5/3} f_L(kL) f_\eta(k\eta) \ , \tag{2.24}$$

where $f_L(kL)$ controls the shape of the energy containing range, $f_\eta(k\eta)$ gives exponential decay in the dissipation range, k is the wavenumber magnitude, C_K is the Kolmogorov constant (taken as 1.62 (Yeung & Zhou, 1997)), $\langle\epsilon\rangle$ is the mean dissipation rate, L is a measure of the size of the large scales, and $\eta = (\nu^3/\langle\epsilon\rangle)^{1/4}$ is the Kolmogorov length scale. Other constants appearing in the model spectrum functions include $p_0 = 2$, $\beta = 5.2$, $c_\eta = 0.4$, and $c_L = (1.262C_K)^3 \approx 8.55$. With $p_0 = 2$, the fitting function $f_L(kL)$ gives $E(k) \sim k^2$ for small k.

The non-cubic and deforming nature of the solution domain implies that both

large-scale sampling and small-scale resolution are dependent on time and direction. The shape of the pre-simulation domain is illustrated in the left of figure 2.1. Large-scale sampling is measured by comparing the integral length scales with the shortest side of the domain, and small-scale resolution by comparing the Kolmogorov scales with the coarsest grid spacing. As the turbulence decays and the length scales grow, large-scale sampling worsens, but small-scale resolution improves. To ensure that the large scales are initially well-sampled, the spectrum parameter L is chosen to be a small fraction of the shortest dimension L_1^0. Resolution of the small scales in cubic domains is often expressed by the non-dimensional parameters $\Delta x/\eta$ and $k_{\max}\eta$. Although good resolution requires $\Delta x/\eta \lesssim 2$ (corresponding to $k_{\max}\eta \gtrsim 1.5$) (Donzis *et al.*, 2008), a larger initial value of $\Delta x/\eta$ for a pre-simulation is acceptable since the resolution improves as the turbulence decays. In this work, directional resolution parameters Δ_α/η ($\Delta_\alpha \equiv \Delta x_\alpha$ is the grid spacing in the x_α direction) must be considered because the grid spacing in each direction varies. For the pre-simulation, the grid spacing is coarsest in the x_2 and x_3 directions, so an initial value of η is specified such that Δ_2/η (which equals Δ_3/η) takes an acceptable value.

Table 2.1 summarizes the pre-simulation initial conditions in this work. Run 1 is a baseline, low Reynolds number simulation comparable to the lowest Reynolds number experiment reported by AW. The modest size of this run allows for multiple independent realizations (Overholt & Pope, 1996), which is useful for statistical reliability since time averaging is not applicable. Runs 2 to 5 are effectively at the same Reynolds number as Run 1, but designed to check the influence of the domain size and grid resolution. In Runs 2 and 3, the domain size is unchanged, but the grid spacing is refined by a factor of 2 first along the x_1 direction, and then the x_2 and x_3 directions. Run 4 shows, relative to Run 1, the effects of improved small-scale resolution in the x_1 direction accompanied by improved large-scale sampling in the x_2 and x_3 directions. Run 5 presents a case in which the domain size is doubled in

Table 2.1: Initial conditions for the pre-simulations. For each run, "count" is the number of independent simulations used for ensemble averaging. In the first block, number of grid points and initial grid metric factors are N_α and $B^0_{\alpha\alpha}$, respectively. The second block lists the Taylor-scale Reynolds number and indicators of large-scale sampling and small-scale resolution. Longitudinal integral length scales are $\ell_{\alpha\alpha}$ and η is the Kolmogorov length scale. Domain length and grid spacing in the x_α direction are given by L_α and Δ_α, respectively. Length scales L and η_0 are inputs to the model spectrum function. Kinematic viscosity for all runs is $\nu = 2.8\times10^{-3}$ (arbitrary units).

Run	1	2	3	4	5	6	7	8	9
Count	16	4	4	4	4	4	4	1	1
N_1	512	1024	1024	1024	1024	1024	2048	4096	4096
$N_2 = N_3$	512	512	1024	1024	1024	1024	2048	4096	4096
B^0_{11}	1	1	1	1	1/2	1	1	1	1/2
$B^0_{22} = B^0_{33}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$
R^0_λ	39.7	39.6	39.0	39.7	39.7	67.7	113	112	113
L^0_1/L	8	8	8	8	16	8	8	8	16
$(L_1/\ell_{11})_0$	16.7	16.8	16.6	16.7	33.4	18.3	19.0	19.1	37.5
$(L_2/\ell_{22})_0$	47.7	47.2	48.3	93.1	92.5	50.2	54.1	112.8	106.7
$(\Delta_1/\eta)_0$	1.31	0.65	0.65	0.65	1.31	1.31	1.31	0.65	1.31
$(\Delta_2/\eta)_0$	3.70	3.70	1.85	3.70	3.70	3.70	3.70	3.70	3.70

all directions compared to Run 1, while the grid spacings are unchanged.

In Runs 6 and 7, the Reynolds number is increased by using the same domain size (i.e., the same grid metrics) as Run 1, and refining the grid spacing to resolve a smaller initial Kolmogorov scale. Two simulations on 4096^3 grids (Runs 8 and 9) are also conducted to investigate the influence of the domain size and grid resolution on Run 7. Compared to Run 7, Run 8 gives improved small-scale resolution in the x_1 direction, and improved large-scale sampling in the x_2 and x_3 directions. Finally, Run 9 improves large-scale sampling compared to Run 7 by doubling the domain length in all directions.

Table 2.2 summarizes the state of each run at the end of the pre-simulation period, just before strain is applied. The turbulence kinetic energy, dissipation rate, and Taylor-scale Reynolds number at this time (designated by subscript or super-

Table 2.2: Simulation parameters at the onset of straining, labeled with subscript or superscript a. Parameters with subscript 0 taken from initial conditions for the pre-simulations (see Table 2.1 for R_λ^0). In the second block ℓ_{11} and ℓ_{22} are longitudinal integral length scales, and ℓ_{21} is the transverse integral length scale in the x_1 direction. The domain does not deform during the pre-simulation, so $L_1^a = L_1^0$ and $L_2^a = L_2^0$. With the shorthand $u_{i,j} = \partial u_i/\partial x_j$ the third block shows the skewness (S) and flatness (F) of longitudinal velocity gradients.

Run	1	2	3	4	5	6	7	8	9
K_a/K_0	0.463	0.463	0.462	0.463	0.463	0.447	0.446	0.446	0.447
$\langle\epsilon\rangle_a/\langle\epsilon\rangle_0$	0.233	0.232	0.224	0.233	0.233	0.248	0.290	0.289	0.290
R_λ^a	38.1	38.1	38.1	38.1	38.1	60.9	93.4	93.4	93.8
$(L_1/\ell_{11})_a$	12.9	12.9	12.7	12.9	25.7	14.6	15.4	15.2	30.5
$(L_1/\ell_{21})_a$	25.5	24.8	25.2	25.3	50.6	28.0	31.0	30.7	62.4
$(L_2/\ell_{22})_a$	37.2	36.5	37.8	73.1	71.7	40.4	42.3	91.4	84.6
$(\Delta_1/\eta)_a$	0.90	0.45	0.45	0.45	0.90	0.92	0.97	0.48	0.96
$(\Delta_2/\eta)_a$	2.54	2.54	1.27	2.54	2.54	2.61	2.73	2.73	2.73
$S(u_{1,1}^a)$	-0.513	-0.509	-0.508	-0.513	-0.511	-0.516	-0.530	-0.530	-0.529
$S(u_{3,3}^a)$	-0.491	-0.489	-0.505	-0.490	-0.489	-0.492	-0.497	-0.499	-0.498
$F(u_{1,1}^a)$	4.15	4.14	4.17	4.16	4.15	4.60	5.10	5.12	5.10
$F(u_{3,3}^a)$	4.08	4.07	4.16	4.07	4.07	4.50	4.97	4.97	4.96

script a) are all (as a result of decay) lower than their initial values (designated by subscript or superscript 0). Large-scale sampling has worsened by this time, as indicated by the larger integral length scales, but small-scale resolution has improved. The integral length scales approximately satisfy $\ell_{11} = 2\ell_{21}$, which is indicative of isotropy. Non-Gaussianity is evident in the skewness factors of longitudinal velocity gradients reaching about -0.5, and the flatness factors reaching values that increase with Reynolds number. It appears that $u_{1,1} \equiv \partial u_1/\partial x_1$ is slightly more non-Gaussian than $u_{3,3} \equiv \partial u_3/\partial x_3$, which may be the result of better small-scale resolution in the x_1 direction compared to the x_2 and x_3 directions. In all runs, the ratios of transverse to longitudinal velocity gradient variances are within 0.5% of the isotropic value of 2. All components of the Reynolds stress anisotropy tensor are 3×10^{-3} or smaller, showing a state of isotropic turbulence despite the non-cubic shape of the domain.

Table 2.3: Post-contraction (subscript or superscript b) parameters and post- to pre-contraction parameter ratios. See tables 2.1 and 2.2 for descriptions of symbols.

Run	1	2	3	4	5	6	7	8	9
B^b_{11}	1/4	1/4	1/4	1/4	1/8	1/4	1/4	1/4	1/8
$B^b_{22} = B^b_{33}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$
$(L_1/\ell_{11})_b$	60.1	59.6	59.1	60.3	121	75.3	83.9	81.4	168
$(L_1/\ell_{21})_b$	19.0	18.5	18.8	18.8	37.7	20.5	22.1	22.2	44.3
$(L_2/\ell_{22})_b$	24.1	23.6	24.8	47.6	46.6	26.3	27.1	57.9	54.3
$(\Delta_1/\eta)_b$	4.34	2.17	2.17	2.17	4.34	4.31	4.30	2.15	4.30
$(\Delta_2/\eta)_b$	1.53	1.53	0.77	1.53	1.53	1.52	1.52	1.52	1.52
K_b/K_a	1.56	1.57	1.57	1.57	1.57	1.57	1.59	1.59	1.59
$\langle\epsilon\rangle_b/\langle\epsilon\rangle_a$	2.13	2.12	2.13	2.13	2.13	1.85	1.54	1.54	1.54
$\langle u_1^2\rangle_b/\langle u_1^2\rangle_a$	0.201	0.199	0.200	0.201	0.202	0.222	0.244	0.246	0.244
$\langle u_2^2\rangle_b/\langle u_2^2\rangle_a$	2.25	2.26	2.24	2.25	2.25	2.24	2.26	2.26	2.26
$\langle u_{2,1}^2\rangle_b/\langle u_{1,1}^2\rangle_b$	4.27	4.27	4.27	4.28	4.27	3.05	2.27	2.28	2.27
$\langle u_{3,2}^2\rangle_b/\langle u_{2,2}^2\rangle_b$	2.98	2.98	2.98	2.98	2.98	2.96	2.90	2.90	2.90
$S(u^b_{1,1})$	0.0154	0.0497	0.0632	0.0447	0.0175	0.345	0.397	0.493	0.399
$S(u^b_{3,3})$	-0.0591	-0.0594	-0.0579	-0.0596	-0.0613	-0.102	-0.178	-0.180	-0.178
$F(u^b_{1,1})$	6.55	6.85	6.90	6.73	6.53	9.72	11.2	12.7	11.2
$F(u^b_{3,3})$	3.43	3.42	3.43	3.43	3.43	3.57	3.82	3.85	3.82

2.3 Application of strain

Following the pre-simulation a time-dependent axisymmetric contraction is applied until the domain elongates by a factor of 4 in the x_1 direction. Using the approach and parameters detailed in §2.1.2, the simulations are run with mean strain rates characterized by a peak non-dimensional strain $S_0^* = 25$. In §2.3.1 single-point moments are presented, and in §2.3.2 the evolution of the spectra is studied.

2.3.1 Single-point moments

First, table 2.3 provides results that summarize the post-contraction state of each run, and will be followed by figures that show the evolution of important statistics over the complete straining period. The post-contraction shape of the domains is as

shown in the right of figure 2.1, with a decrease in B_{11} and increase in B_{22} and B_{33} relative to their pre-contraction values in table 2.1. As the domain deforms and the physical length scales of the turbulence evolve, large-scale sampling and small-scale resolution (second block in table 2.3) change. The post-contraction domains are still large compared with the integral length scales, indicating that large-scale sampling is still adequate. It is worth noting that the ratio L_1/ℓ_{21} has dropped compared to its pre-contraction value, despite a factor of 4 increase in L_1. This implies a substantial increase in the transverse integral length scale ℓ_{21}, which is consistent with the formation of coherent longitudinal vortices in the extensional direction (Rogers & Moin, 1987). As expected, small-scale resolution worsens in the direction where the domain is stretched (x_1), but improves in the directions of compression (x_2 and x_3).

Continuing in table 2.3, the turbulence kinetic energy is amplified in all cases, with the amplification ratio nearly independent of the Reynolds number. The dissipation rate also increases, but less so at higher Reynolds number. It is clear that the turbulence becomes anisotropic, as velocity fluctuations in the extensional and compressive directions are suppressed and amplified, respectively. Anisotropy at the small scales is also seen in the statistics of velocity gradient fluctuations, such as the ratio of transverse to longitudinal velocity gradient variances, which differ from the isotropic value of 2. Apparently, at higher Reynolds numbers, the ratio $\langle u_{2,1}^2\rangle/\langle u_{1,1}^2\rangle$ becomes less anisotropic but $\langle u_{3,2}^2\rangle/\langle u_{2,2}^2\rangle$ remains close to 3, which (as discussed later) is an indication of quasi two-dimensionality in the cross-sectional plane.

Third and fourth moments of longitudinal velocity gradients are also included in table 2.3. In 3-D incompressible isotropic turbulence the skewness of the longitudinal velocity gradient is negative, and related to the phenomenon of vortex stretching (Batchelor, 1953; Tavoularis *et al.*, 1978). However, as reported by AW and others (Mills & Corrsin, 1959; Sjögren & Johansson, 1998), the simulations show that axisymmetric contraction causes the skewness of $\partial u_1/\partial x_1$ to undergo a change in sign,

Table 2.4: Post-contraction skewness and flatness of longitudinal velocity gradients predicted by rapid-distortion theory.

Run	1	2	3	4	5	6	7	8	9
$S(u^b_{1,1})$	-0.665	-0.663	-0.660	-0.665	-0.663	-0.683	-0.706	-0.706	-0.706
$S(u^b_{3,3})$	-0.0247	-0.0221	-0.0245	-0.0257	-0.0260	-0.0271	-0.0251	-0.0253	-0.0255
$F(u^b_{1,1})$	5.08	5.06	5.05	5.08	5.07	5.64	6.26	6.26	6.24
$F(u^b_{3,3})$	4.18	4.17	4.28	4.18	4.17	4.67	5.16	5.16	5.16

with its magnitude increasing with the Reynolds number. In contrast, the skewness of $\partial u_3/\partial x_3$ remains negative but its magnitude is reduced compared to that observed in isotropic turbulence. At the same time, there is an increase in the flatness of $\partial u_1/\partial x_1$ and a decrease in the flatness of $\partial u_3/\partial x_3$ during the contraction. The flatness of $\partial u_1/\partial x_1$ also increases with Reynolds number, which is consistent with AW. A comparison of the post-contraction flatness of $\partial u_1/\partial x_1$ between runs with different resolution (Run 2 and 8 versus 1 and 7, respectively) suggests that higher-order derivative statistics in this work are affected by finite resolution in the extensional direction. However, the main interest of this study is lower-order quantities such as spectra, for which the lower-resolution simulations are adequate.

It is also useful to compare DNS, which uses a finite strain rate, to RDT. Following the pre-simulation, the Fourier coefficients of the velocity field are evolved according to well-known relations for inviscid RDT (Townsend, 1976; Lee & Reynolds, 1985). Each velocity field is subjected to a 4:1 area ratio axisymmetric contraction. Statistics are then ensemble averaged over all realizations for each run, e.g., over the 16 simulations comprising Run 1. RDT predicts $K_b/K_a = 2.1$ and $\langle\epsilon\rangle_b/\langle\epsilon\rangle_a = 5.5$ for all runs, which are considerably different than the DNS results presented in table 2.3. The extent of large-scale anisotropy is predicted well by RDT; it gives $b^b_{11} = -0.3$ and $b^b_{22} = 0.15$ for all runs. There is more discrepancy between the DNS and RDT when examining velocity derivative statistics. For example, RDT estimates $\langle u^2_{2,1}\rangle_b/\langle u^2_{1,1}\rangle_b = 8$,

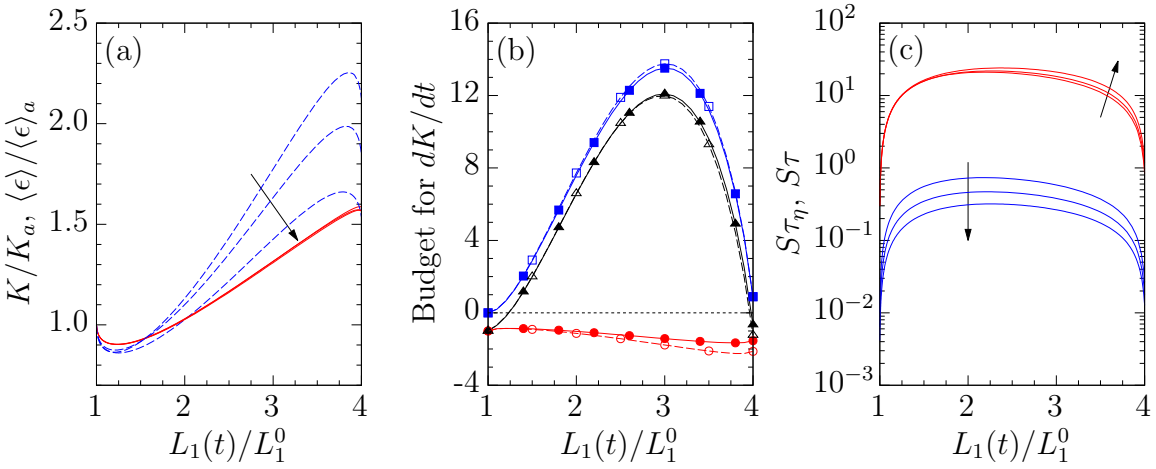

Figure 2.4: Evolution of (a) K and $\langle\epsilon\rangle$ normalized by initial values for Runs 4, 6, and 8, (b) budget for dK/dt normalized by $\langle\epsilon\rangle_a$ for Runs 4 and 8, and (c) non-dimensional strain rates for Runs 4, 6, and 8. In (a), solid curves (red) for $K(t)/K_a$ for the three runs are almost coincident, and dashed curves (blue) are for $\langle\epsilon(t)\rangle/\langle\epsilon\rangle_a$, with R_λ increasing in the direction of the arrow. In (b), dashed curves with open symbols for Run 4, and solid curves with filled symbols for Run 8: ■ and □ for production, ● and ○ for minus the dissipation, and ▲ and △ for overall rate of change. In (c), mean strain rate normalized by large eddy turnover time τ (upper red curves) and Kolmogorov time scale τ_η (lower blue curves), with R_λ increasing in the directions of arrows.

which is markedly different from the DNS results. The velocity derivative statistic $\langle u_{3,2}^2\rangle_b/\langle u_{2,2}^2\rangle_b$, however, takes a post-contraction value of 3 with RDT, which is very similar to the DNS results. Table 2.4 shows the RDT predictions for higher-order velocity derivative statistics. While RDT fails to predict a positive value for the skewness of $\partial u_1/\partial x_1$, a small negative value of the skewness of $\partial u_3/\partial x_3$ is consistent with the DNS at lower Reynolds numbers. The large increase in the flatness of $\partial u_1/\partial x_1$ at high Reynolds numbers, and the decrease in the flatness of $\partial u_3/\partial x_3$ observed in the DNS are not predicted by RDT.

For information on the evolution of the turbulence during the period of axisymmetric contraction, and to facilitate comparison with previous studies, it is useful to show results against the total deformation at a given time, rather than time itself. Figure 2.4 shows, as a function of $L_1(t)/L_1^0$ (which ranges from 1 to 4 for a 4:1 contraction ratio), in (a) the evolution of turbulence kinetic energy and dissipation

rate, in (b) various terms in the turbulence kinetic energy budget, and in (c) the non-dimensional mean strain rates. To focus on the effects of the Reynolds number results are selected from Runs 4, 6, and 8 (see table 2.1). For homogeneous turbulence the turbulence kinetic energy budget is governed by production and dissipation, as

$$\frac{dK}{dt} = -\langle u_i u_j \rangle \frac{\partial \langle U_i \rangle}{\partial x_j} - \langle \epsilon \rangle \ . \tag{2.25}$$

It can be seen that both K and $\langle \epsilon \rangle$ initially decay since it takes finite time for production (initially zero) to grow to exceed the dissipation. Subsequently, as both variables grow, the evolution of K is almost independent of the Reynolds number, while dissipation grows less rapidly at high Reynolds number. Because of the form of the strain rate profile (figure 2.2), both K and $\langle \epsilon \rangle$ decrease towards the end of the straining period when the strain rate is weak. The production term in frame (b) is driven by the mean flow and is essentially the same for all Reynolds numbers simulated. It is also dominant over dissipation during the straining period, which explains why the kinetic energy evolution is nearly identical for all runs. The non-dimensional strain rates in frame (c) indicate that the strain rates are strong compared to the time scales of the large-scale motions, but actually weak from the perspective of the small scales.

The production of turbulence kinetic energy by mean strain requires anisotropy in the Reynolds stresses, which is expressed by the anisotropy tensor $b_{ij} = \langle u_i u_j \rangle / (2K) - \delta_{ij}/3$, and its second and third coordinate-frame invariants, given by

$$\eta = (b_{ij} b_{ji}/6)^{1/2} \ ; \quad \xi = (b_{ij} b_{jk} b_{ki}/6)^{1/3} \ , \tag{2.26}$$

where here η is not to be confused with the Kolmogorov scale. For isotropic turbulence subjected to axisymmetric contraction the invariants satisfy

$$\xi = -\eta \ . \tag{2.27}$$

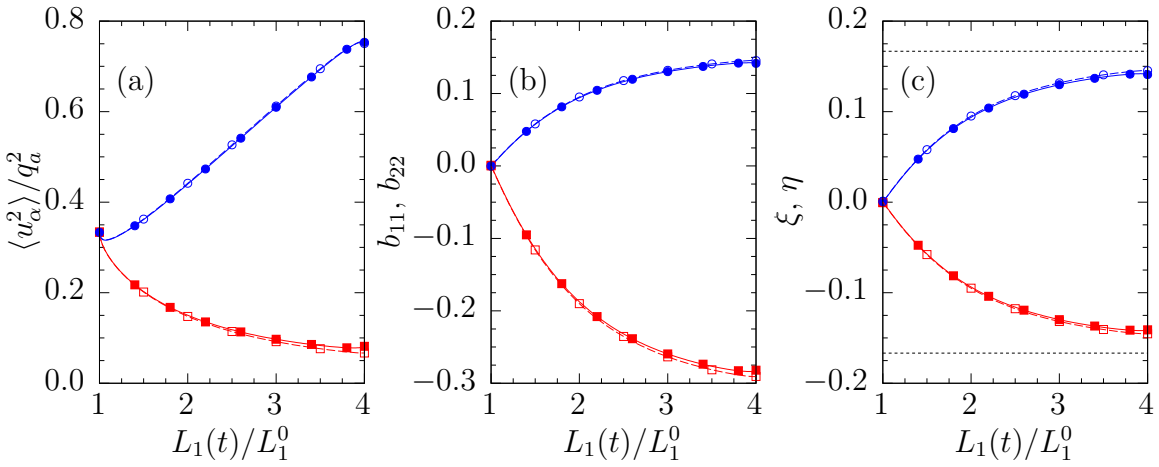

Figure 2.5: Evolution of (a) Reynolds stresses normalized by $q_a^2 = 2K_a$, (b) components of the Reynolds stress anisotropy tensor, and (c) anisotropy tensor invariants for Run 4 (dashed curves with open symbols) and Run 8 (solid curves with filled symbols). Symbols ■ and □ for $\langle u_1^2\rangle/q_a^2$, b_{11}, and ξ in each respective figure. Symbols ● and ○ for $\langle u_2^2\rangle/q_a^2$, b_{22}, and η in each respective figure. Dashed lines in (c) for two-dimensional isotropic limit.

Figure 2.5 shows for Runs 4 and 8, (a) the evolution of the Reynolds stresses, (b) the evolution of the anisotropy tensor elements b_{11} and b_{22}, and (c) the invariants ξ and η. The results suggest that the large-scale anisotropy is completely determined by the total deformation at any time t, but is independent of the Reynolds number. Since the large-scale anisotropy depends only on the total deformation, the anisotropy development is the same as observed in simulations with constant strain rates (Lee & Reynolds, 1985), and can be predicted almost exactly by RDT. Although (2.27) is in principle exact, in numerical results it is not guaranteed if the large scales contributing the most to the Reynolds stress tensor are not sampled well in a domain of finite size. The present results show that the large scales are sufficiently well-sampled.

The development of anisotropy in the Reynolds stress tensor can be analyzed further using the Reynolds stress transport equation (2.19). Figure 2.6 presents the terms in the balance equations for (a) $\langle u_1^2\rangle$ and (b) $\langle u_2^2\rangle$, using data from Runs 4 and 8. For the production terms, in this flow $P_{11} < 0$ whereas $P_{22} > 0$. The rapid pressure strain term quickly counteracts the anisotropy generated by the production term. The

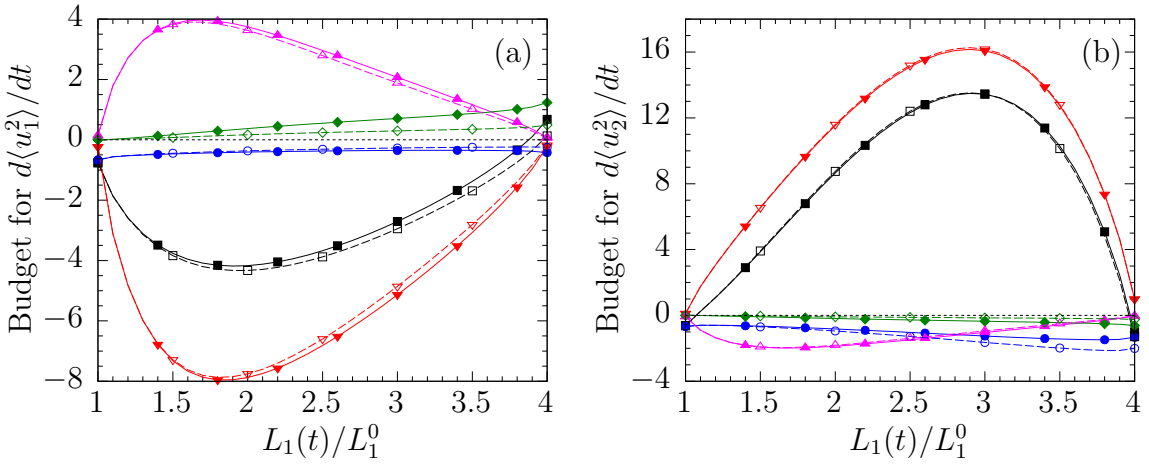

Figure 2.6: Reynolds stress budgets normalized by initial dissipation rate $\langle\epsilon\rangle_a$ during the application of strain for Run 4 (dashed curves with open symbols) and Run 8 (solid curves with filled symbols). Budget terms for $d\langle u_1^2\rangle/dt$ in (a), and budget terms for $d\langle u_2^2\rangle/dt$ in (b): ▼ and ▽ for production, ▲ and △ for rapid pressure-strain, ◆ and ◇ for slow pressure-strain, ● and ○ for dissipation, and the sum of all budget terms marked by ■ and □.

slow pressure strain term, on the other hand, becomes significant only at later times, and is more important at higher Reynolds number. In frame (b) the production effect is clearly the dominant term in the Reynolds stress budget for $\langle u_2^2\rangle$, being resisted only weakly by the rapid pressure at early times and viscous dissipation at later times. In both frames the production terms (of either sign) reach peak amplitude at intermediate times, which is a consequence of the time-dependent strain rate and is different from results from simulations of constant strain (Lee & Reynolds, 1985).

Although large-scale statistics show little dependence on the Reynolds number, small-scale statistics, such as the dissipation rate in figure 2.4(a), show a strong Reynolds number dependence. It was already observed in table 2.3 that the small scales become anisotropic during the straining period. The geometry of axisymmetric contraction also suggests that vorticity components in different directions (ω_1 versus ω_2 and ω_3) will have different statistics. It is therefore important to investigate the behavior of both vorticity and velocity gradient statistics, with data given in figure 2.7.

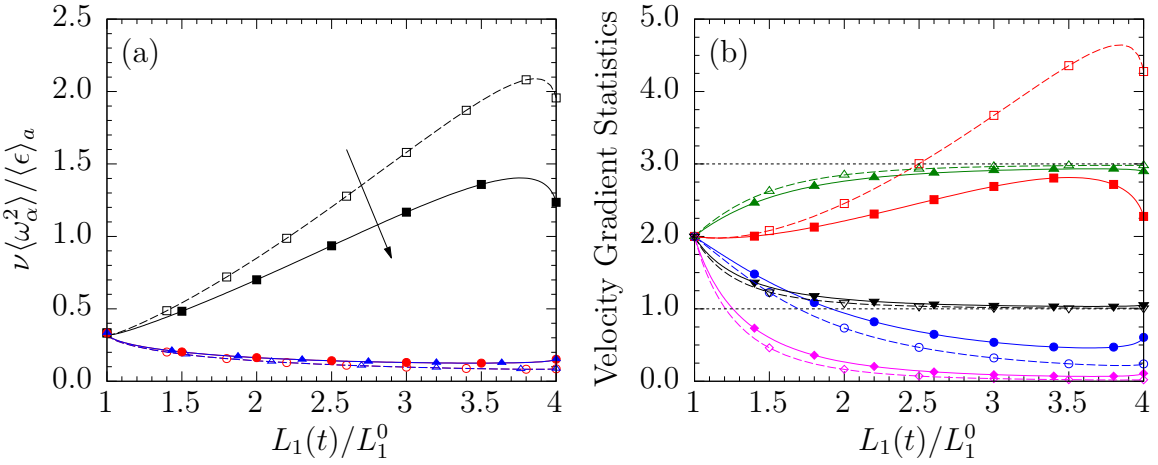

Figure 2.7: Evolution of (a) mean-square vorticities normalized by dissipation rate and (b) velocity derivative statistics for Run 4 (dashed curves with open symbols) and Run 8 (solid curves with filled symbols). In (a), ■ and □ for $\nu\langle\omega_1^2\rangle/\langle\epsilon\rangle_a$, ● and ○ for $\nu\langle\omega_2^2\rangle/\langle\epsilon\rangle_a$, and ▲ and △ for $\nu\langle\omega_3^2\rangle/\langle\epsilon\rangle_a$. Let $u_{i,j} = \partial u_i/\partial x_j$. In (b), ■ and □ for $\langle u_{2,1}^2\rangle / \langle u_{1,1}^2\rangle$, ● and ○ for $\langle u_{1,2}^2\rangle / \langle u_{2,2}^2\rangle$, ◆ and ◇ for $-\langle u_{1,1}^2\rangle / \langle u_{2,3}u_{3,2}\rangle$, ▲ and △ for $\langle u_{3,2}^2\rangle / \langle u_{2,2}^2\rangle$, and ▼ and ▽ for $-\langle u_{2,2}^2\rangle / \langle u_{2,3}u_{3,2}\rangle$. Values for $\langle u_{3,2}^2\rangle / \langle u_{2,2}^2\rangle$ and $-\langle u_{2,2}^2\rangle / \langle u_{2,3}u_{3,2}\rangle$ in two-dimensional isotropic turbulence marked by horizontal dashed lines at 3 and 1, respectively.

In homogeneous turbulence one can write

$$\langle\epsilon\rangle = \nu\left(\langle\omega_1^2\rangle + \langle\omega_2^2\rangle + \langle\omega_3^2\rangle\right) \quad . \tag{2.28}$$

Figure 2.7(a) shows, for Runs 4 and 8, the three mean-squared vorticities, normalized by the viscosity and the pre-contraction dissipation rate. As expected from the geometry, axisymmetric contraction amplifies and aligns vorticity in the extensional direction while reducing vorticity in the compressive directions (Rogers & Moin, 1987), but the effects are weaker at higher Reynolds number. This dependence on the Reynolds number can be understood by noting that (lower curves in figure 2.4(c)) as the Reynolds number increases (in the order Runs 4, 6, 8) the value of the non-dimensional strain rate $S\tau_\eta$ becomes smaller. In other words, as the range of time scales in the flow widens with increasing Reynolds number, the strain rate becomes weaker with respect to the small scales. Hence, at higher Reynolds numbers, the

fluctuating vorticity is less responsive to the applied strain, as reflected in the weaker amplification of $\langle \omega_1^2 \rangle$ in figure 2.7(a).

To further assess departures from local isotropy using statistics of the velocity gradients, it is useful to compare the simulation data with a number of relations that apply for 3-D incompressible isotropic turbulence, such as (with $\alpha \neq \beta$)

$$\langle (\partial u_\alpha / \partial x_\beta)^2 \rangle = 2 \langle (\partial u_\alpha / \partial x_\alpha)^2 \rangle \ , \tag{2.29}$$

$$\langle (\partial u_\alpha / \partial x_\alpha)^2 \rangle = -2 \langle (\partial u_\alpha / \partial x_\beta)(\partial u_\beta / \partial x_\alpha) \rangle \ . \tag{2.30}$$

Figure 2.7(b) shows the evolution of ratios formed from (2.29) and (2.30) during the straining period. The velocity derivative statistics are initially isotropic, but depart from isotropy as the strain is applied. The statistics are more anisotropic at the lower Reynolds number because the strain rate is more rapid with respect to the small scales. In their experiments, AW measured $\langle u_{2,1}^2 \rangle / \langle u_{1,1}^2 \rangle$ for a wide range of Reynolds numbers (see figure 8 in AW). The post-contraction values for this statistic in the DNS are similar to those reported by AW, and also remain closer to the isotropic value as the Reynolds number is increased.

It is worth noting that figure 2.7(b) shows that during straining, $\langle u_{3,2}^2 \rangle / \langle u_{2,2}^2 \rangle$ and $-\langle u_{2,2}^2 \rangle / \langle u_{2,3} u_{3,2} \rangle$ approach asymptotic limits 3 and 1, respectively. This can be explained by generalizing (2.29) and (2.30) to n-dimensional isotropic turbulence, where $n \geq 2$. For $\alpha \neq \beta$ (Pope, 2000; Gotoh *et al.*, 2007), the relations are

$$\langle (\partial u_\alpha / \partial x_\beta)^2 \rangle = (n+1)/(n-1) \langle (\partial u_\alpha / \partial x_\alpha)^2 \rangle \ , \tag{2.31}$$

$$\langle (\partial u_\alpha / \partial x_\alpha)^2 \rangle = -(n-1) \langle (\partial u_\alpha / \partial x_\beta)(\partial u_\beta / \partial x_\alpha) \rangle \ . \tag{2.32}$$

Substituting $n = 2$ into (2.31) and (2.32), the respective limiting values observed in figure 2.7(b) for $\langle u_{3,2}^2 \rangle / \langle u_{2,2}^2 \rangle$ and $-\langle u_{2,2}^2 \rangle / \langle u_{2,3} u_{3,2} \rangle$ are obtained. This suggests that the small-scales under an axisymmetric contraction of sufficient strength tend to

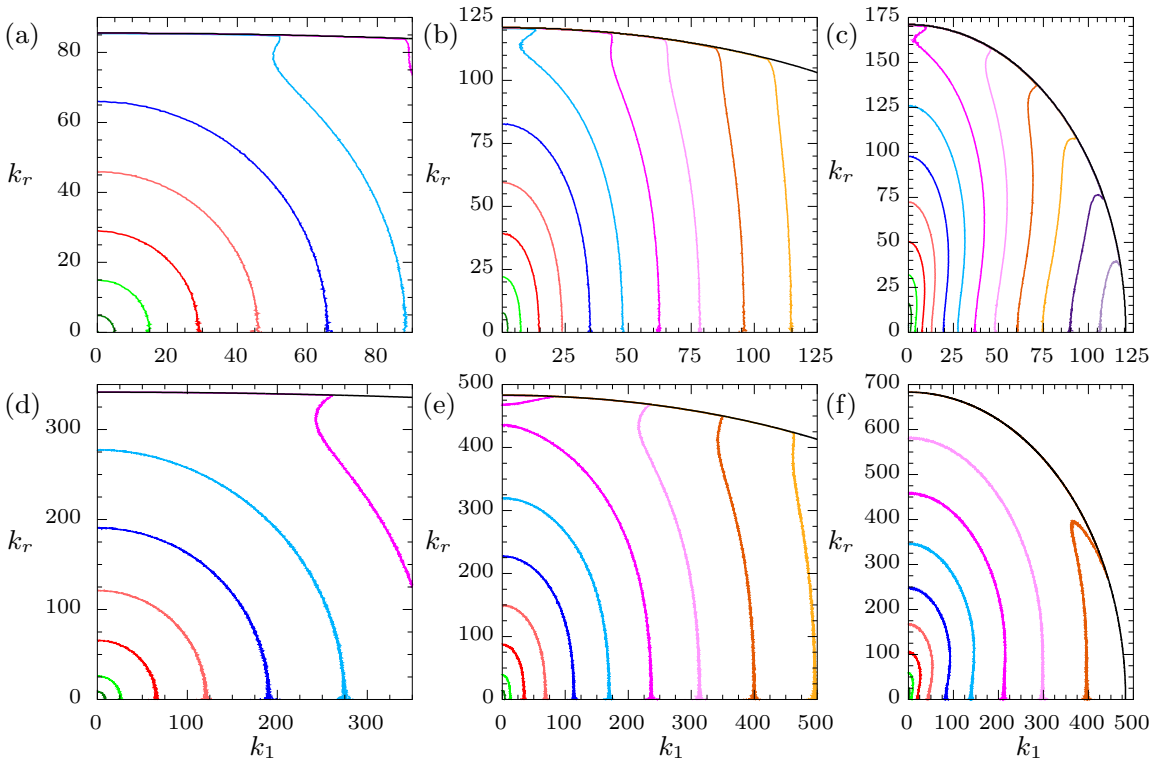

Figure 2.8: Axisymmetric energy spectrum for (top row) Run 4 and (bottom row) Run 8 (left column) before the application of strain, (middle column) half-way through straining ($L_1/L_1^0 = 2$), and (right column) at end of the straining period. Contour levels decrease by a factor of 10. Spectra plotted against the instantaneously distorting wavenumbers, and multiplied by 2 to recover K when integrating over k_r and non-negative k_1. Simulation cutoff wavenumbers marked by outermost black boundary. Spectra normalized by $\sin\phi = k_r/k$ to obtain circular contours in isotropic turbulence; data for $k_r = 0$ omitted from plot.

asymptotically approach the limiting state of isotropic turbulence in two dimensions. At the same time, a number of relations for velocity gradient statistics conforming to a state of local axisymmetry (George & Hussein, 1991) are well verified in the DNS data. In other words, the post-contraction state of the small scales is one of local axisymmetry with velocity gradients in the extensional direction becoming asymptotically small compared to gradients in the compressive directions. (This is supported by the line for the ratio $-\langle u_{1,1}^2 \rangle / \langle u_{2,3} u_{3,2} \rangle$ in the figure approaching almost zero on the scales chosen.)

2.3.2 Spectral evolution

Results discussed above indicate that anisotropy in the mean flow and the large scale motions ultimately leads to strong anisotropy in the small scales. This observation implies that isotropy in the flow is scale-dependent, which is best explored through spectral quantities in wavenumber space. In this section results are reported for an axisymmetric representation of the energy spectrum, 1-D spectra like those measured in the experiments of AW, and various terms in the spectral energy budget.

Figure 2.8 shows contour plots of the axisymmetric energy spectrum $E_A(k_1, k_r)$ for Runs 4 and 8 before the straining period (left column), half-way through the straining period (middle column), and at the end of the straining period (right column). If the turbulence is isotropic the energy spectrum would depend only on $k = (k_1^2 + k_r^2)^{1/2}$, which means isocontours of $E_A(k_1, k_r)$ (after the normalization discussed in §2.1.3) would be circles in the (k_1, k_r) plane. This is indeed the case for the pre-contraction spectra in frames (a) and (d), except that the shape of the contours is distorted near the simulation cutoff wavenumbers, which are marked by the outermost black elliptical boundaries in the plots. As the strain is applied (from the left column to the right column), the spectra change significantly. The post-contraction energy spectra in frames (c) and (f) are highly anisotropic at all scales of motion (with non-circular contours), which is consistent with the large-scale and small-scale anisotropy observed for single-point moments in §2.3.1. The contours in frame (c) for Run 4 show evidence of stronger anisotropy than those in frame (f) for Run 8. This is in agreement with the results for single-point moments, which showed that the small scales become more anisotropic during the application of strain at lower Reynolds number. During straining, the domain is lengthened in the x_1 direction but shortened in x_2 and x_3. This results in wavenumber distortion, where the wavenumbers in k_1 decrease while those in k_2 and k_3 increase. Close observation of figure 2.8 reveals that the cutoff wavenumber boundary for each simulation is distorted as the strain

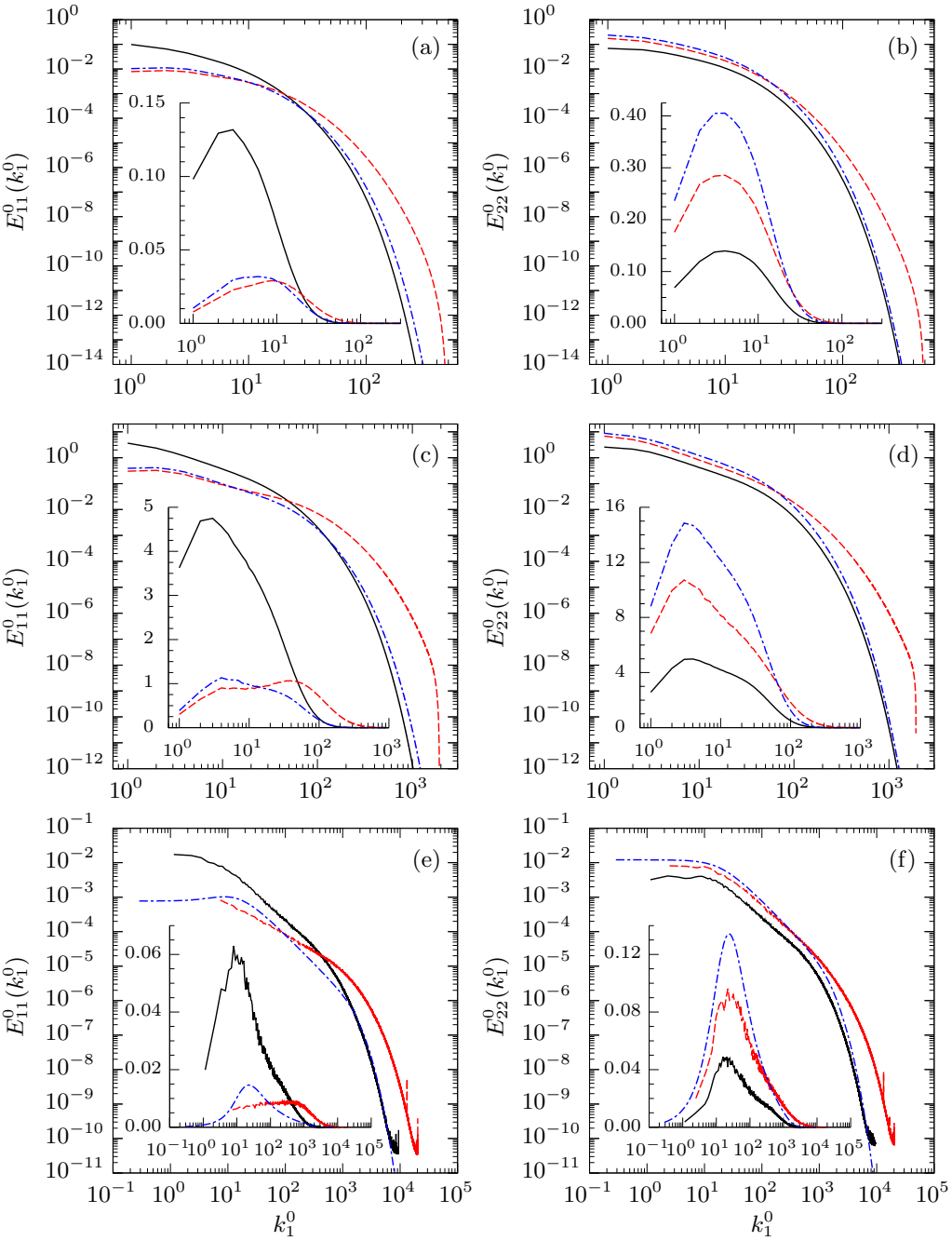

Figure 2.9: Comparison of pre- and post-contraction longitudinal (left column) and transverse (right column) 1-D spectra, for DNS Run 4 at low Reynolds number (top row), DNS Run 8 at higher Reynolds number (middle row), and the high Reynolds number experiment of AW (bottom row), all shown as functions of pre-contraction wavenumbers. Experimental data are reproduced from figure 11 of AW by permission of the authors. Solid lines (black) for pre-contraction DNS or experiment, dashed (red) for post-contraction DNS or experiment, and dashed-dotted (blue) for post-contraction RDT. Insets in each frame show the same spectra but multiplied by the wavenumber (e.g., $k^0_1 E^0_{22}(k^0_1)$), such that the areas under the curve on log-linear scales give the mean-squared velocities. For the DNS, the transverse spectra $E^0_{22}(k^0_1)$ and $E^0_{33}(k^0_1)$ are averaged with each other when producing frames (b) and (d).

is applied. Wavenumber distortion causes energy to accumulate in long-wavelength (low wavenumber) modes in the extensional direction, which is consistent with the formation of long coherent vortical structures in the extensional direction (Rogers & Moin, 1987).

To characterize the scale-dependent anisotropy between different velocity components, figure 2.9 shows the 1-D spectra of the u_1 and u_2 velocity fluctuations for low and high Reynolds numbers in the DNS (top and middle rows, respectively), compared with the highest-Reynolds-number data in the AW experiments (bottom row, from AW figure 11). The spectra are, based on rationale discussed earlier in §2.1.3, all shown as functions of the initial (pre-contraction) wavenumbers. The effect of strain on $E_{11}^0(k_1^0)$ is a decrease at low wavenumbers but an increase at high wavenumbers. As the Reynolds number increases, a pronounced rightward shift is seen to develop in the peak of $k_1^0 E_{11}^0(k_1^0)$ in both the DNS and experiment (red dashed lines in the insets of frames (c) and (e)). The effect of the mean strain on $E_{22}^0(k_1^0)$ is, in contrast, an increase in the spectrum at all values of k_1^0, with a milder change in the spectrum shape and a weaker Reynolds number dependence. The suppression of $\langle u_1^2 \rangle$ and amplification of $\langle u_2^2 \rangle$ are also indicated by changes in the area under the curves in the inset to each figure.

Figure 2.9 also contains inviscid RDT results for comparison. While the theory appears to reasonably predict the shape of $E_{11}^0(k_1^0)$ at low wavenumbers, it can be seen that the theory fails to capture the rightward shift in $k_1^0 E_{11}^0(k_1^0)$ at high Reynolds number discussed above, while it over-predicts the amplification of $E_{22}^0(k_1^0)$ at low wavenumbers. This indicates the physical mechanisms neglected in RDT, namely nonlinear energy transfer, slow pressure-strain, and viscous dissipation, play a significant role in the evolution of the spectral structure of the Reynolds stresses.

The roles and relative importance of physical processes governing the evolution of the spectra described above can be studied using the spectral budget equation (2.13),

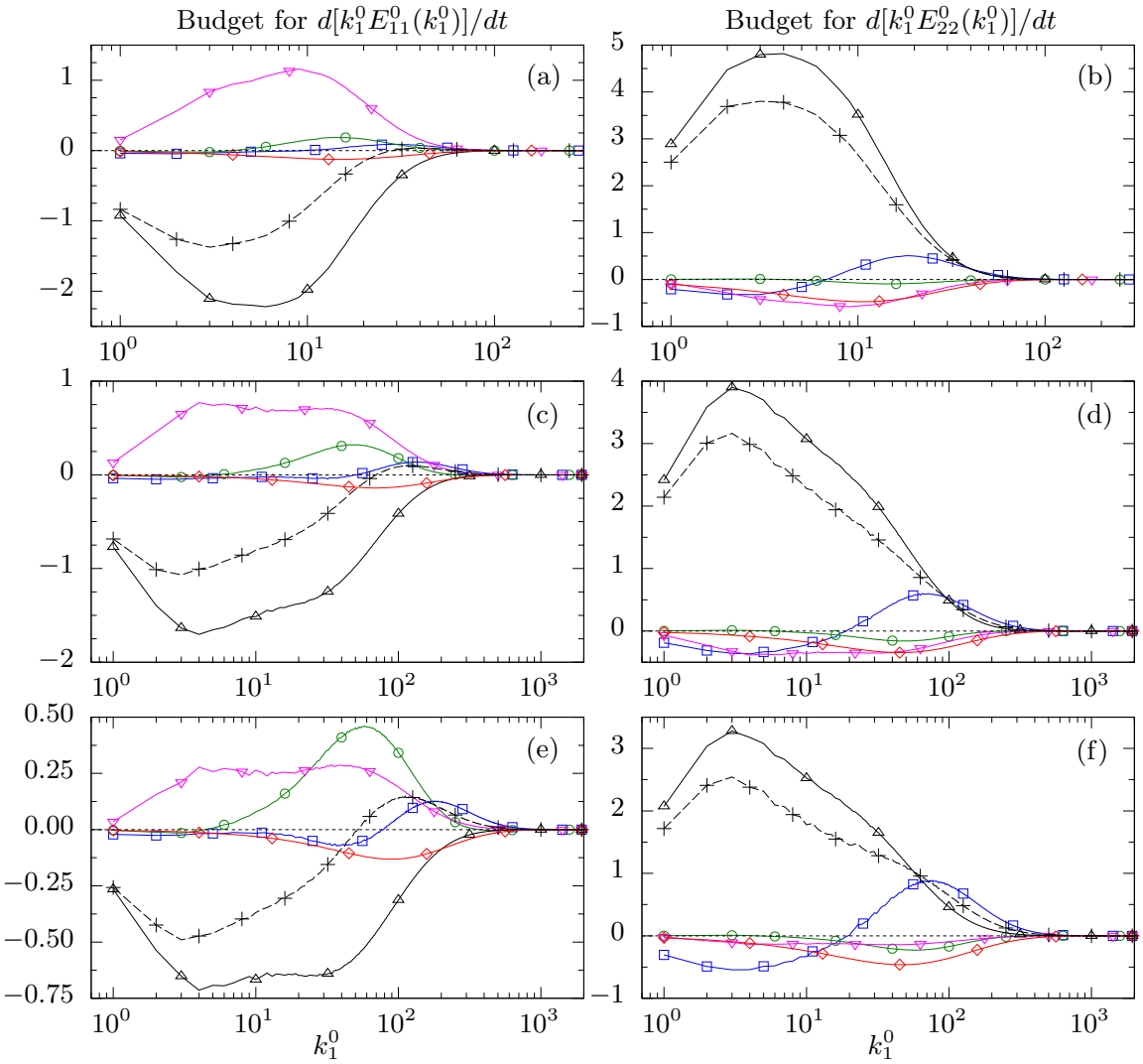

Figure 2.10: Balance terms from (2.13) contributing to the evolution of the 1-D compensated spectra, normalized by the pre-contraction dissipation rate and shown as functions of pre-contraction wavenumbers. Top row: $L_1(t)/L_1^0 = 2.5$ from Run 4; middle row: $L_1(t)/L_1^0 = 2.5$ from Run 8; bottom row: $L_1(t)/L_1^0 = 3.5$ from Run 8. Left and right columns show data for $k_1^0 E_{11}^0(k_1^0)$ and $k_1^0 E_{22}^0(k_1^0)$, respectively. In all frames: $\triangle$ (black solid) for production, $\triangledown$ (magenta) for rapid pressure-strain, $\bigcirc$ (green) for slow pressure-strain, $\square$ (blue) for nonlinear transfer, $\diamond$ (red) for minus the dissipation, and $+$ (black dashed) for total rate of change. Data for $k_1^0 E_{22}^0(k_1^0)$ averaged with data for $k_1^0 E_{33}^0(k_1^0)$ due to the axisymmetry of the turbulence.

which was written for the case of a single Fourier mode. The balance equations for the 1-D spectra parameterized by the initial wavenumbers are obtained by integrating (2.13) over planes perpendicular to k_1, and then dividing by the total deformation (like in (2.21)). The terms are then multiplied by k_1^0 to study the evolution of the

compensated spectra (the insets in figure 2.9). It is important to determine what causes the rightward shift in the peak of $k_1^0 E_{11}^0(k_1^0)$ at high Reynolds numbers, and what causes the general disagreement between the RDT and DNS spectra at high wavenumbers in all simulations. In figure 2.10 the balance terms are shown at different times and different Reynolds numbers. As expected for axisymmetric contraction, the production term is negative for the spectrum of u_1, but positive for the spectrum of u_2. The rapid pressure-strain counteracts the generation of anisotropy by taking the opposite sign of the production term, but a net tendency for anisotropy still persists. The nonlinear term is negative for low k_1^0 and positive for high k_1^0, indicating that there is a forward cascade of energy to high k_1^0. This forward cascade opposes the tendency for energy to pile up near the $k_1^0 = 0$ plane during straining (see figure 2.8). The overall magnitude of the nonlinear transfer term is greater for $k_1^0 E_{22}^0(k_1^0)$ compared to $k_1^0 E_{11}^0(k_1^0)$, which is likely due to the fact that $\langle u_2^2 \rangle$ is amplified during the straining, while $\langle u_1^2 \rangle$ is suppressed.

In the simulations, the magnitude of the strain reaches a maximum and then decreases. To see how this is reflected in the spectral budgets, results are presented for Run 8 at two deformations in the bottom two rows of figure 2.10. The production and rapid pressure-strain terms (which depend on the mean strain rate) in the bottom row are smaller than those in the middle row. As the production terms weaken, the relative importance of the nonlinear terms (especially at intermediate and high wavenumbers) increases. This is also the case for the slow pressure-strain term, which peaks at intermediate wavenumbers close to the peak observed for $k_1^0 E_{11}^0(k_1^0)$. It may be concluded, therefore, that the slow pressure strain, which is neglected in RDT theory, is the main contributor to rightward shift in $k_1^0 E_{11}^0(k_1^0)$ both in the DNS and the experiments of AW. The increasing importance of slow pressure strain at later times noted here is also consistent with a similar feature in figure 2.6 addressed in §2.3.1. On the other hand, the slow pressure-strain term is not as important to the

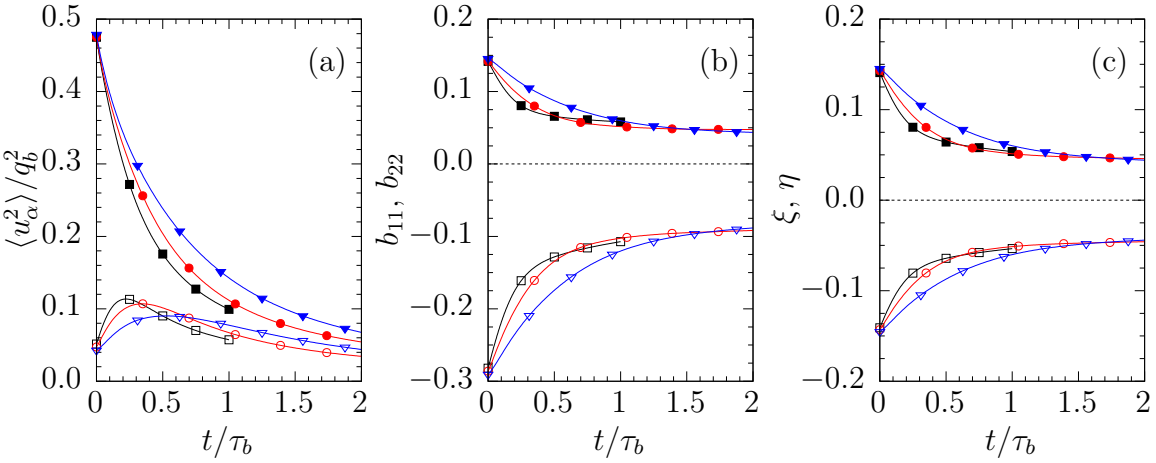

Figure 2.11: Relaxation of (a) Reynolds stresses normalized by $q_b^2 = 2K_b$, (b) components of the anisotropy tensor, and (c) anisotropy tensor invariants for Runs 4 (▼ and ▽), 6 (● and ○), and 8 (■ and □). Upper curves for $\langle u_2^2\rangle/q_b^2$, b_{22}, and η in each figure, respectively. Lower curves for $\langle u_1^2\rangle/q_b^2$, b_{11}, and ξ in each figure, respectively. Dashed lines at 0 in (b) and (c) denote values for isotropic turbulence.

evolution of $k_1^0 E_{22}^0(k_1^0)$, which (besides production) is heavily influenced by nonlinear transfer at intermediate and high wavenumbers.

2.4 Relaxation of axisymmetric turbulence

The results in §2.3 show that application of strain causes anisotropy at both the large and small scales. This section focuses on the relaxation that occurs when the mean strain is removed and the turbulence decays. Varying degrees of return to isotropy as seen in single-point moments and axisymmetric and 1-D spectra are examined. The effects of nonlinear transfer and slow pressure fluctuations are also investigated.

Figure 2.11 shows the component energy ratios and Reynolds stress anisotropy tensor information as functions of time since the end of strain, normalized by the time scale $\tau = 2K/\langle\epsilon\rangle$ at post-strain conditions. It is clear that anisotropy decreases significantly in the earlier stages of relaxation, slightly more rapidly if the Reynolds number is higher (Runs 6 and 8). However, the data also strongly suggests that the large scales will either not return to isotropy fully or will take almost an indefinitely

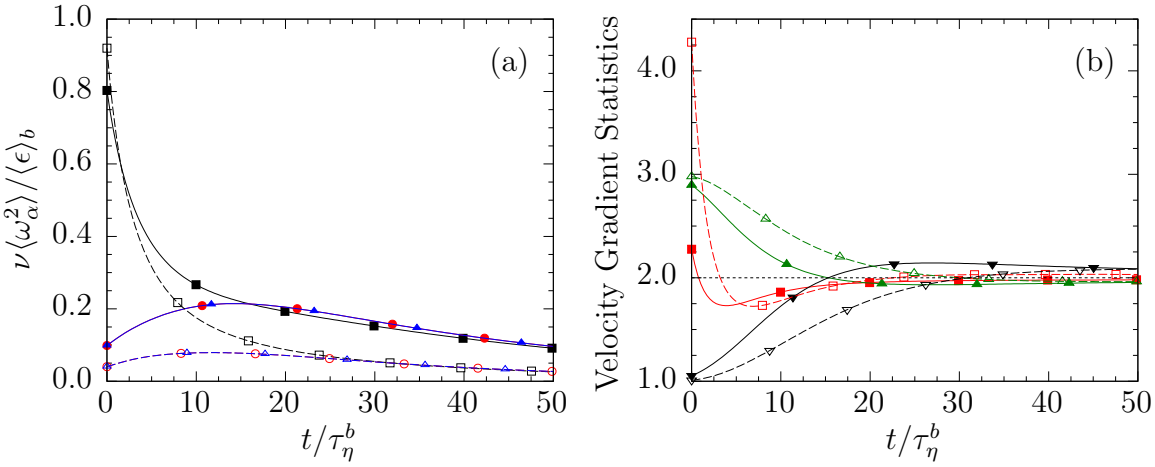

Figure 2.12: Relaxation of (a) vorticity contributions to dissipation rate and (b) velocity gradient statistics for Run 4 (dashed curves with open symbols) and Run 8 (solid curves with filled symbols). In (a), ■ and □ for $\nu\langle\omega_1^2\rangle/\langle\epsilon\rangle_b$, ● and ○ for $\nu\langle\omega_2^2\rangle/\langle\epsilon\rangle_b$, and ▲ and △ for $\nu\langle\omega_3^2\rangle/\langle\epsilon\rangle_b$. Let $u_{i,j} = \partial u_i/\partial x_j$. In (b), ■ and □ for $\langle u_{2,1}^2\rangle / \langle u_{1,1}^2\rangle$, ▲ and △ for $\langle u_{3,2}^2\rangle / \langle u_{2,2}^2\rangle$, ▼ and ▽ for $-\langle u_{2,2}^2\rangle / \langle u_{2,3}u_{3,2}\rangle$, and horizontal line at 2 for three-dimensional isotropic turbulence.

long time to do so. This is consistent with the DNS of Davidson *et al.* (2012) and large-eddy simulations of Chasnov (1995), which both showed persistent anisotropy at the large scales for axisymmetric Saffman turbulence ($E(k) \sim k^2$ as $k \to 0$). This consistency in trend is perhaps not surprising, since the pre-simulation initial conditions are also of the Saffman type (due to the choice $p_0 = 2$ in (2.24)). Long-time effects have also been checked by extending Runs 1 and 5 to relaxation times several times longer than shown in the figure. As the integral length scales grow during the extended relaxation period, the axisymmetry property $\xi = -\eta$ does not hold as well for the smaller domain (Run 1), but a finite level of anisotropy is likely to persist even at asymptotically large times.

The concept of local isotropy in turbulence suggests the small scales may become isotropic during relaxation. Figure 2.12 shows the mean-square vorticity components and several ratios of derivative covariances during relaxation for Runs 4 and 8. Since these are small-scale quantities time is normalized by the (post-contraction) Kolmogorov time scale. In frame (a), $\langle\omega_1^2\rangle$ initially decreases rapidly, while $\langle\omega_2^2\rangle$ and

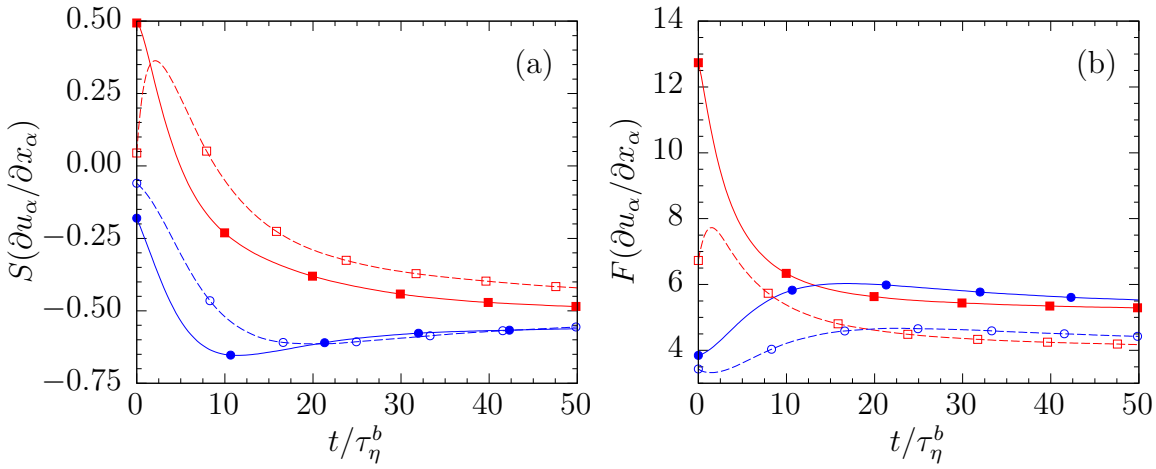

Figure 2.13: Relaxation of (a) skewness and (b) flatness of longitudinal velocity gradients for Run 4 (dashed curves with open symbols) and Run 8 (solid curves with filled symbols): ■ and □ for statistics of $\partial u_1/\partial x_1$, and ● and ○ for statistics of $\partial u_3/\partial x_3$.

$\langle\omega_3^2\rangle$ increase before decreasing. By about 20 τ_η, which for Run 8 corresponds to $t/\tau_b \approx 0.22$, the mean-square vorticities are almost equal (while the Reynolds stresses in figure 2.11 are still very anisotropic). In frame (b), velocity derivative variance and covariance ratios also return to their isotropic value of 2. Similar to AW (their figure 24), the ratio $\langle u_{2,1}^2\rangle/\langle u_{1,1}^2\rangle$ initially undershoots, and then increases toward the isotropic value.

For higher-order moments, figure 2.13 shows the post-contraction evolution of skewness and flatness factors of the longitudinal velocity gradients. The general trend is towards a skewness in the neighborhood of -0.5, which is typical of isotropic turbulence, and a flatness factor higher than 3 showing a noticeable increase with the Reynolds number. For Run 4 the skewness and flatness factors show a transient overshoot, which then gives way to the trend noted above. It is also interesting that, as a measure of isotropy, the flatness factors of $\partial u_1/\partial x_1$ and $\partial u_3/\partial x_3$ become nearly equal faster than the corresponding skewness factors. The relatively slow equilibration of the skewness factors could be due to their connection with the energy cascade, which also depends on the large scales.

The fact that the large scales and the small scales return to isotropy (at least par-

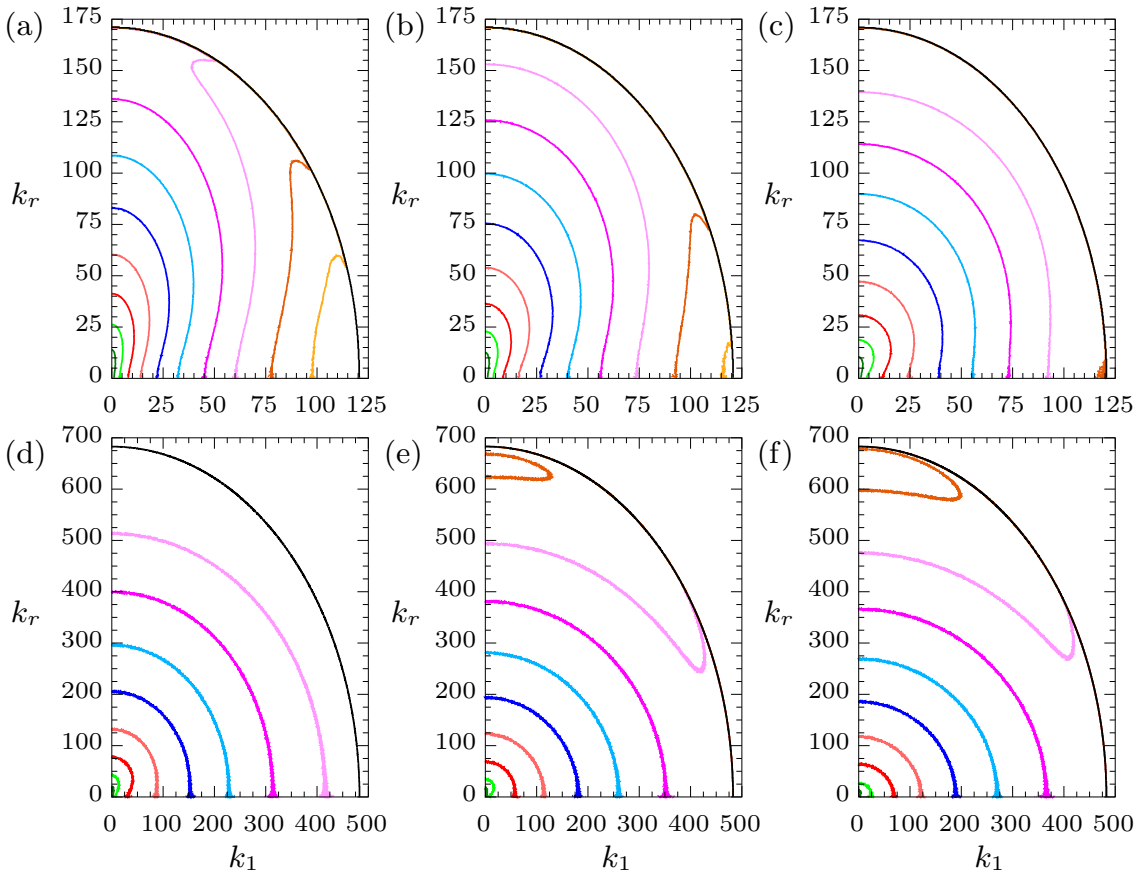

Figure 2.14: Relaxation of axisymmetric energy spectrum for (top row) Run 4 and (bottom row) Run 8 at (left column) $t/\tau_b = 0.05$, (center column) $t/\tau_b = 0.1$, and (right column) $t/\tau_b = 0.2$. Contour levels decrease by a factor of 10. Spectra plotted against post-contraction wavenumbers, and multiplied by two to recover K when integrating over k_r and non-negative k_1. Simulation cutoff wavenumbers marked by outermost black boundary. Spectra normalized by $\sin\phi = k_r/k$ to obtain circular contours in isotropic turbulence; data for $k_r = 0$ not plotted.

tially) at different rates in time suggest that, at a given time in the relaxation phase, the spectra may display a series of non-trivial shapes. Figure 2.14 presents the post-contraction evolution of the axisymmetric energy spectrum for Runs 4 and 8 (top and bottom rows, respectively), at three relatively early time instants corresponding to $t/\tau_b = 0.05$ (left column), $t/\tau_b = 0.1$ (middle column), and $t/\tau_b = 0.2$ (right column). While the axisymmetric spectra immediately following the contraction are highly anisotropic (right column in figure 2.8), a trend towards a more isotropic appearance (circular contours in the (k_1, k_r) plane) is evident at high wavenumbers during relax-

ation. This relaxation occurs faster for Run 8 because it has a greater contrast in time scales between the large scales and the small scales. Because the energy spectrum evolves only according to dissipation and energy redistribution (through the nonlinear term), the increase in the axisymmetric energy spectrum at high k_1 implies that there is a strong energy transfer to higher k_1 during the initial relaxation period. Consistent with large-scale statistics in figure 2.11, anisotropy persists at low k_1 and low k_r during relaxation.

The strong energy transfer to higher wavenumbers in the extensional direction is expected to have a significant effect on both longitudinal and transverse 1-D spectra, which are shown in figure 2.15 for DNS Runs 4 and 8 (top and middle rows, respectively) at different times (increasing in the directions of the arrows) during relaxation. To facilitate comparison with experiment, data from figure 19 of AW is included (with permission) in the bottom row. The longitudinal spectrum $E_{11}(k_1)$ initially increases during relaxation (primarily at high wavenumbers), corresponding to an increase of both $\langle u_1^2 \rangle$ (see figure 2.11(a)) and $\langle (\partial u_1/\partial x_1)^2 \rangle$ during the early relaxation period. In contrast, the transverse spectrum $E_{22}(k_1)$ shows a decrease at low wavenumbers accompanied by an increase at high wavenumbers, corresponding to a reduction of $\langle u_2^2 \rangle$ and $\langle u_3^2 \rangle$ even though mean-square transverse velocity gradients increase. At high Reynolds numbers the compensated spectrum $k_1 E_{22}(k_1)$ in the inset of frame (d) develops a "double-peak" structure, which was a major finding in the AW experiments (frame (f)). The evolution of this part of the spectrum is non-monotonic in time, with the feature being most prominent in at $t/\tau_b = 0.2$ (frame (d) dark blue). As noted by AW, this "double-peak" structure during relaxation appears to be a distinctive result of high Reynolds number. The observations here confirm that the simulations are successfully reproducing key flow physics in the experiments. The physical mechanisms contributing to this double-peak can be elucidated by analyzing the spectral energy budget, as below.

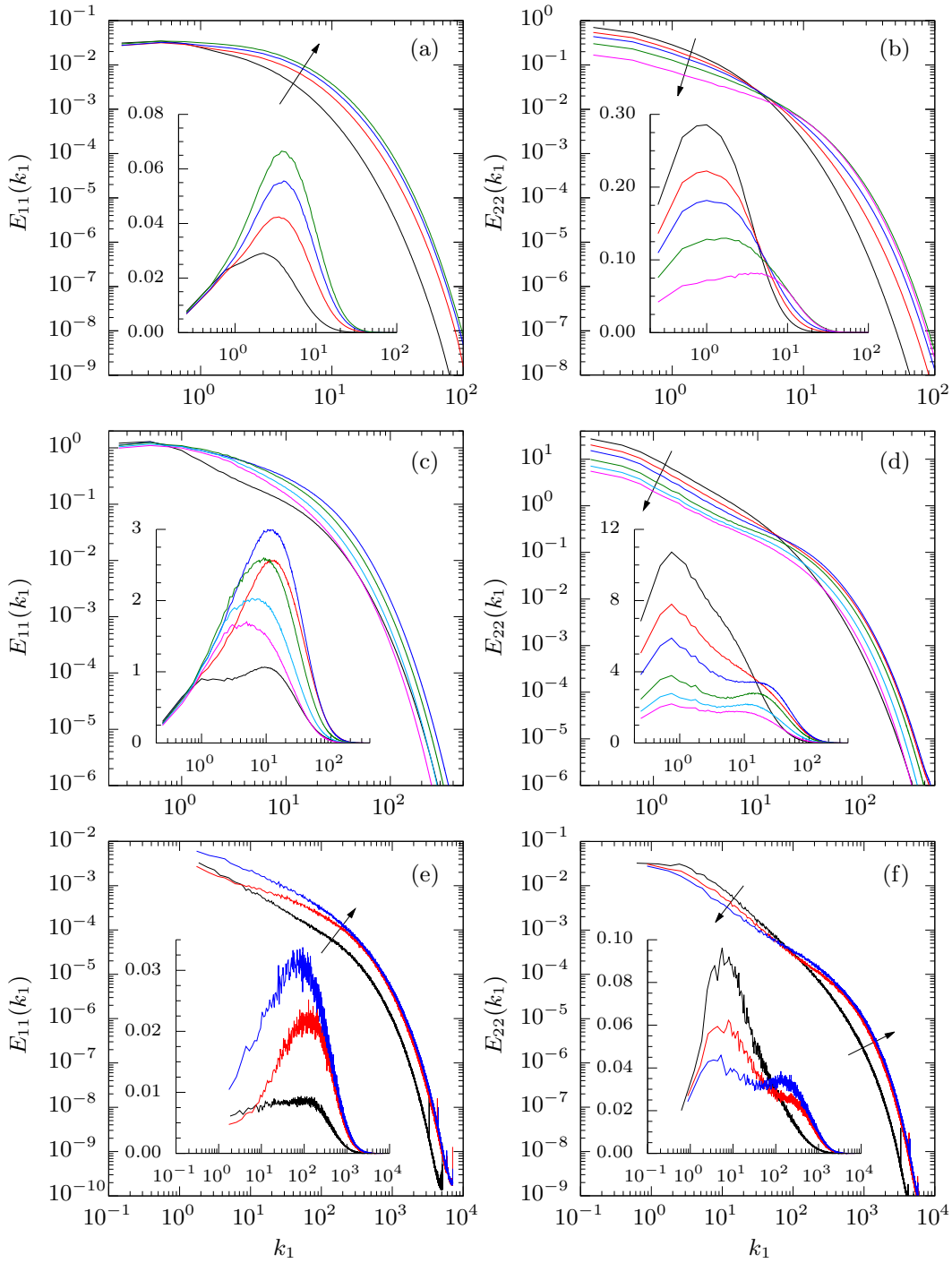

Figure 2.15: Relaxation of (left column) longitudinal and (right column) transverse 1-D spectra, for (top row) DNS Run 4, (middle row) DNS Run 8, and (bottom row) high Reynolds number AW experiment. Experimental data are reproduced from figure 19 of AW by permission of the authors. Insets multiply spectra by k_1. Time (for the DNS) or downstream evolution (for the experiments) increasing in directions of arrows. For DNS, curves at $t/\tau_b = 0$ (black), $t/\tau_b = 0.1$ (red, not in (c) log-log plot for clarity), $t/\tau_b = 0.2$ (blue), $t/\tau_b = 0.4$ (green), $t/\tau_b = 0.6$ (light blue, not in top row for clarity), and $t/\tau_b = 0.8$ (magenta, not in (a) for clarity). For DNS, $E_{22}(k_1)$ and $E_{33}(k_1)$ averaged with each other for frames (b) and (d) due to axisymmetry.

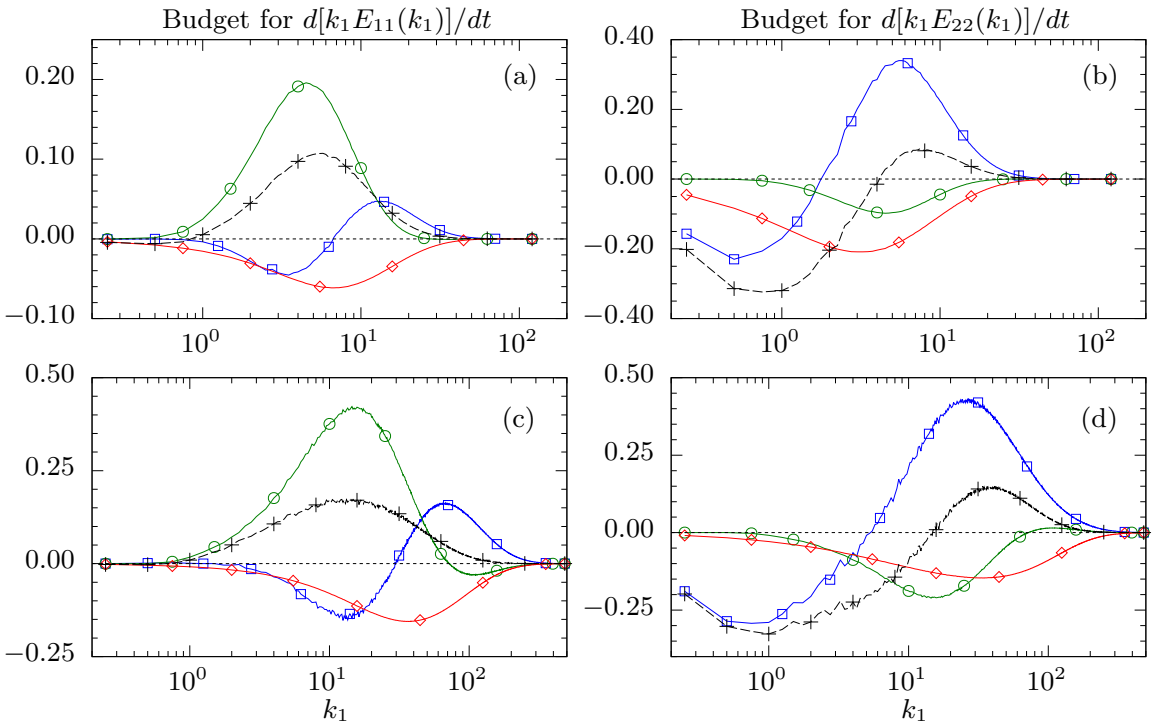

Figure 2.16: Terms from (2.13) contributing to the evolution of 1-D compensated spectra for (top row) Run 4 and (bottom row) Run 8 at $t/\tau_b = 0.05$ into relaxation, normalized by dissipation rate $\langle\epsilon\rangle_b$. Terms for $k_1E_{11}(k_1)$ in left column, and terms averaged for $k_1E_{22}(k_1)$ and $k_1E_{33}(k_1)$ in right column: ○ (green) for slow pressure-strain, □ (blue) for nonlinear transfer, ◇ (red) for minus the dissipation, and + (black dashed) for total rate of change.

As done in figure 2.10 for turbulence during the application of strain, figure 2.16 presents the balance of terms that govern the evolution of the 1-D compensated spectra early in the relaxation period ($t/\tau_b = 0.05$) for Runs 4 and 8. For both runs, the evolution of $k_1E_{11}(k_1)$ (in frames (a) and (c)) is dominated by the slow pressure-strain term (circles in green), which is positive over a wide range of wavenumbers. The integral of the pressure-strain term gives the pressure-strain correlation, which promotes isotropy by increasing $\langle u_1^2\rangle$ while decreasing $\langle u_2^2\rangle$ and $\langle u_3^2\rangle$. For both $k_1E_{11}(k_1)$ and $k_1E_{22}(k_1)$, the nonlinear term (squares in blue) shows the characteristics of a forward cascade in k_1, being negative at low k_1, but positive at high k_1. This forward cascade from low k_1 to high k_1 during relaxation is likely a consequence of a prior accumulation of energy near the $k_1 = 0$ plane during the contraction (see figure 2.8 frames (c)

and (f), and discussion for figure 2.14). A comparison between the left and right columns of this figure shows that the nonlinear term plays a more important role in the evolution of $k_1E_{22}(k_1)$ than for $k_1E_{11}(k_1)$. At intermediate and high wavenumbers, the nonlinear transfer is strong enough to exceed dissipation and pressure-strain combined, leading to an increase in $k_1E_{22}(k_1)$ during the early phase of relaxation. A comparison of frames (b) and (d) also indicates that as the Reynolds number increases, nonlinear spectral transfer becomes stronger, while the effects of viscous dissipation are shifted towards higher wavenumbers. Since (in frame (d)) nonlinear transfer is the term of largest overall magnitude, it is a principal contributor to the change in shape of the transverse spectrum observed in both the AW experiments and the numerical simulations.

While results on the balance terms for the compensated spectra in figure 2.16 explain the reduction in $k_1E_{22}(k_1)$ at low wavenumbers and the increase in $k_1E_{22}(k_1)$ at high wavenumbers, the occurrence of a double-peak structure in $k_1E_{22}(k_1)$ at higher Reynolds number is more subtle. It may be noted that a higher Reynolds number gives a wider range of scales, and that results in figure 2.14 show that at higher Reynolds number (frames in the bottom row) the high k_1 region of the axisymmetric energy spectrum increased quickly. A similar feature is seen in the transverse 1-D spectrum (figure 2.15(d)), which exhibited a rapid increase at high wavenumbers early in the relaxation period. This is in contrast with a slower increase of the spectrum at high wavenumbers for the lower Reynolds number case shown in figure 2.15(b). In other words, the double-peak structure can be interpreted as the result of a slower decrease of $E_{22}(k_1)$ at low k_1 combined with a faster increase at high k_1, provided the contrast in time scales between these two processes is sufficiently strong, thus requiring high Reynolds number.

The formation of two peaks in $k_1E_{22}(k_1)$ also depends on what processes influence the wavenumbers between the two peaks. For Run 8 this wavenumber range is

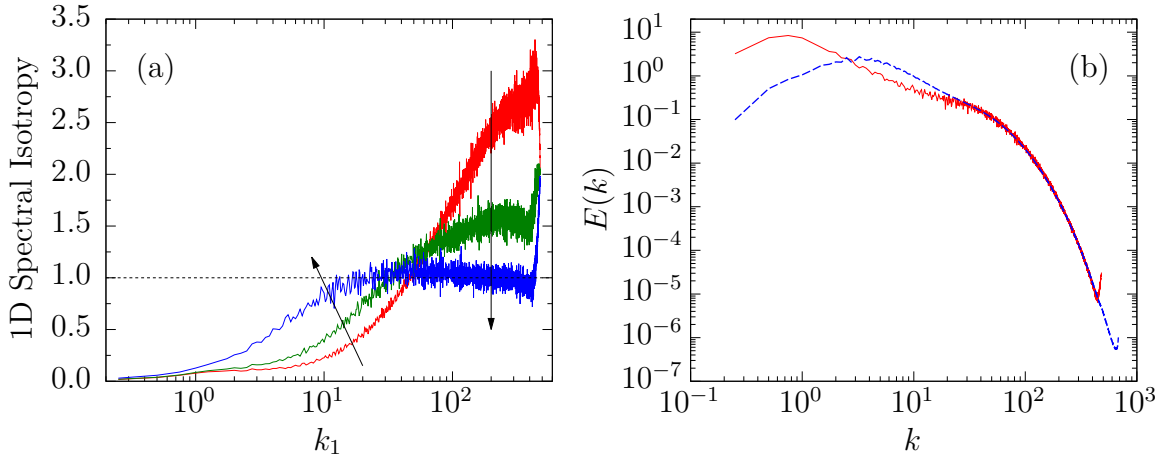

Figure 2.17: Post-contraction spectral isotropy for Run 8. In (a), isotropy of 1-D component spectra measured with (2.33) at (time increasing in the directions of the arrows) $t/\tau_b = 0$ (red, steepest curve), $t/\tau_b = 0.05$ (green, intermediate curve), and $t/\tau_b = 0.2$ (blue, plateau at 1 present); dashed line at 1 for isotropic turbulence. In (b), measured (dashed blue) 3-D energy spectrum compared with calculated spectrum using (2.34) (solid red) at $t/\tau_b = 0.2$.

approximately $2 \lesssim k_1 \lesssim 10$. If a forward cascade of energy occurs in k_1, the curve $T_{22}(k_1)$ should undergo a change in sign, which is evident in figure 2.16(d) at $k_1 \approx 5$ (although the figure plots $k_1 T_{22}(k_1)$). The wavenumber location of this change in sign is nearly fixed during the relaxation period, and is located between the two peaks that emerge in $k_1 E_{22}(k_1)$. In this wavenumber range as the nonlinear term is close to zero, the pressure-strain and dissipation terms become more important. Hence, while the strong decrease in the spectrum at low wavenumbers and the rapid increase in the spectrum at high wavenumbers are primarily results of strong nonlinear interactions, the formation of two clearly visible peaks in $k_1 E_{22}(k_1)$ depends subtly on pressure-strain and dissipation effects in the wavenumber region between the two peaks. This effect is likely to become more prominent at higher Reynolds numbers which will support an even wider range scales.

As suggested throughout this section, the small scales become isotropic during relaxation, while the large scales appear to retain a significant degree of anisotropy for a very long time. The scale-dependent degree of return to isotropy was first ob-

served for the axisymmetric energy spectrum in figure 2.14, where contours of kinetic energy at high wavenumbers became nearly circular for the higher Reynolds number simulation. To assess the scale-dependent anisotropy quantitatively, comparisons can be made with theoretical relations for spectra in isotropic turbulence, such as

$$E_{22}(k_1) = (1/2)[E_{11}(k_1) - k_1 dE_{11}(k_1)/dk_1] \ , \tag{2.33}$$

$$E(k) = -k\, d[E_{11}(k)/2 + E_{22}(k)]\, /dk \ . \tag{2.34}$$

It is convenient (Jiménez *et al.*, 1993; Yeung & Zhou, 1997) to form the ratio of the right- to left-hand side of (2.33), and to compare the actual 3-D spectrum with a result calculated from the 1-D spectra using (2.34). The derivatives in (2.33) and (2.34) are obtained using a simple central difference scheme, although some noise from numerical differentiation is inevitable. Figure 2.17 presents the spectral isotropy results for Run 8 focusing on the early relaxation period. In frame (a), the isotropy coefficient formed from (2.33) is shown at three times (increasing in the directions of the arrows) during relaxation corresponding to $t/\tau_b = 0$, $t/\tau_b = 0.05$, and $t/\tau_b = 0.2$. The 1-D spectra are initially very anisotropic (steep red curve), as indicated by the lack of a plateau at 1 for the isotropy coefficient. By $t/\tau_b = 0.2$ into the relaxation period, the isotropy coefficient forms a plateau of height close to 1 over a wide range of wavenumbers, suggesting that the 1-D spectra are becoming isotropic at intermediate and high wavenumbers. In frame (b), the energy spectrum calculated using (2.34) agrees well with the measured energy spectrum for $k \gtrsim 30$ as early as $t/\tau_b = 0.2$ into the relaxation period. The anisotropy at low wavenumbers seen in both frames shows that the large scales return to isotropy more slowly, as expected.

2.5 Summary

This chapter presents a numerical investigation of isotropic turbulence subjected to irrotational axisymmetric contraction and subsequent relaxation. A series of direct numerical simulations with grid resolution up to 4096^3 have been conducted. Special care was taken to mimic the flow physics in the wind tunnel experiments of Ayyalasomayajula & Warhaft (2006), which used an axisymmetric contraction with a 4:1 area ratio. The development of anisotropy and the physical mechanisms behind scale-dependent anisotropy during the contraction and subsequent relaxation are of fundamental interest.

Although homogeneous turbulence subjected to spatially uniform mean velocity gradients can be simulated in a solution domain moving with the mean flow, time-dependent mean strain rates are necessary to produce flow conditions corresponding to experiments in spatially-evolving wind tunnels. Accordingly, a technique is developed to specify a time-dependent strain rate based on the convective time for fluid traveling along the wind tunnel centerline. The resulting strain rates in the DNS are, in non-dimensional form, similar to the strain rate histories in the AW experiments. Before applying the strain, a pre-simulation is first carried out with a specified initial energy spectrum to obtain physically realistic conditions of unforced isotropic turbulence. The Reynolds numbers simulated in this work are limited by constraints on large-scale sampling and small-scale resolution, which are compounded by the non-cubic aspect ratio of the solution domains. A domain sufficiently large along its shortest dimension compared to the integral length scales is required to ensure that the numerical solution remains statistically axisymmetric at all times.

As expected, axisymmetric contraction leads to anisotropy in the Reynolds stress tensor. Velocity fluctuations are suppressed in the extensional direction but amplified in the compressive directions. The degree of anisotropy is independent of

Reynolds number but is a function of the total strain applied. The small scales also become anisotropic. Mean-square vorticity in the extensional direction is enhanced (figure 2.7), and changes are observed in the skewness of the longitudinal velocity gradients. Rapid distortion theory (RDT) predicts anisotropy at the large scales well but not at the small scales, which implies that nonlinear effects excluded in RDT play an important role. The axisymmetry about the x_1 direction motivates the use of spectra extracted as functions of k_1 and the wavenumber magnitude in the transverse plane $k_r = \sqrt{k_2^2 + k_3^2}$. The axisymmetric energy spectrum becomes anisotropic at all scales of motion during the application of strain, and shows that energy is accumulated in long-wavelength (low k_1) modes in the x_1 direction (figure 2.8). The effect of strain on the longitudinal 1-D spectrum of u_1 fluctuations is a decrease at low wavenumbers but an increase at high wavenumbers (figure 2.9). At high Reynolds number the compensated form of this spectrum shows a rightward shift to higher wavenumbers as a consequence of the nonlinear effects of slow pressure-strain.

Following the removal of the mean strain, the small scales relax faster than the large scales. The contrast in relaxation timescales becomes more apparent as the range of timescales in the flow increases with increasing Reynolds number. Statistics of velocity gradients show a return to local isotropy, whereas a residual level of anisotropy in the Reynolds stresses appears to persist indefinitely. The axisymmetric energy spectrum at high Reynolds number quickly becomes isotropic at high wavenumbers (small scales), whereas at low Reynolds number the spectrum is slow in its return to isotropy (figure 2.14). In close correspondence with the AW experiments, the compensated transverse spectrum $k_1 E_{22}(k_1)$ undergoes a qualitative change at higher Reynolds number, where a "double-peak" structure emerges at intermediate times during relaxation (figure 2.15(d)). Analyses of the spectral budget following the contraction (figure 2.16) indicate that the transverse 1-D spectrum is dominated by nonlinear energy transfer to high k_1 wavenumbers, and that slow pressure-strain

and viscous dissipation also play a role in establishing the double-peak structure in $k_1 E_{22}(k_1)$. The formation of the double-peak structure requires that the high wavenumbers (small scales) relax very quickly compared to the low wavenumbers (large scales), and thus requires a high Reynolds number and a wide range of scales.

In summary, the motivation for this chapter was to see whether DNS could reproduce the experimental finding by AW that turbulence under axisymmetric contraction and subsequent relaxation undergoes a qualitative change as the Reynolds number is increased. Through a series of computations using a time-dependent strain rate formulated to mimic the AW wind tunnel, numerical simulations are successful in reproducing and helping to explain the experimental observations. The behavior of turbulence under irrotational, axisymmetric straining is a canonical problem (Lumley & Newman, 1977) for which there is a continuing need for data at high Reynolds number (Warhaft, 2009). The results of this study have implications for engineering devices such as nozzles and diffusers where high Reynolds number turbulent flows are typically subjected to axisymmetric contraction, relaxation, or expansion. The mixing of passive scalars, especially small temperature fluctuations, in these flows (Gylfason & Warhaft, 2009) is also a subject of both theoretical significance and practical interest, and is studied in the next chapter.

CHAPTER III

TURBULENT MIXING UNDER AXISYMMETRIC CONTRACTION

The turbulent mixing of a passive scalar is a fundamental problem that is relevant to many engineering applications occurring in many different flow geometries and under a wide range of flow conditions. While many experimental (Warhaft, 2000) and computational (Gotoh & Yeung, 2013) studies of passive scalar mixing are known, they are often set in shear flows or isotropic turbulence, and much less is known for the canonical configuration of turbulent mixing under axisymmetric contraction (Warhaft, 1980; Gylfason & Warhaft, 2009). As seen in Chapter II, axisymmetric contraction can have a profound impact on the velocity field. When the strain rate is sufficiently large, anisotropy develops at all scales of motion, and during the subsequent relaxation the flow is dominated by strong nonlinear interactions at higher Reynolds numbers. This chapter continues to investigate the effects of axisymmetric contraction by introducing passive scalars in the flow and studying their evolution. The efforts focus on achieving similar conditions to the experiments of Gylfason & Warhaft (2009) (GW henceforth) and providing further validation for some of the relations derived in their work.

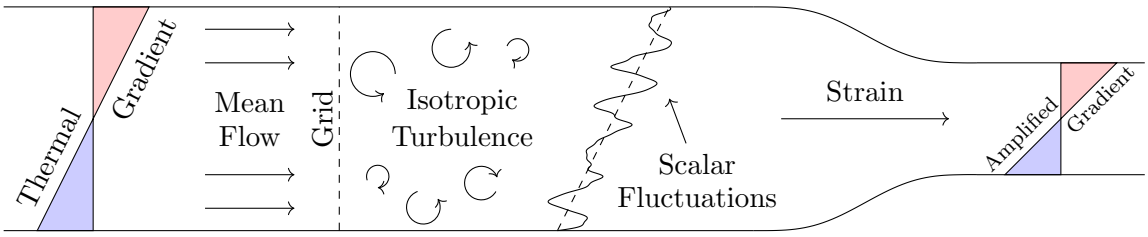

Figure 3.1: Two-dimensional illustration of the Gylfason & Warhaft (2009) experiment in which grid-generated isotropic turbulence generates scalar fluctuations through interactions with a mean scalar gradient before passing through a 4:1 area-ratio axisymmetric contraction.

It is instructive to review the experimental configuration of GW, sketched in figure 3.1, as the numerical simulations will follow it closely. Aside from the addition of axisymmetric contraction, the procedure to generate scalar fluctuations depicted in figure 3.1 is one that is widely used in the experimental community (Warhaft, 2000). Beginning upstream of the turbulence generating device, a mean thermal gradient is introduced into the flow. The mean flow then passes through a turbulence generator, e.g., a passive or active grid, which generates nearly-isotropic turbulent velocity fluctuations. Turbulent advection of the mean scalar field then produces scalar fluctuations, while the isotropy of the velocity fluctuations ensures that the mean scalar field remains intact (Corrsin, 1952). At this point, one may measure the evolution of scalar fluctuations in decaying isotropic turbulence, but GW proceeded to pass the flow through an axisymmetric contraction, which promotes anisotropy in the velocity field (see Chapter II) and the scalar field. When the strain rate is high enough, rapid distortion theory (RDT) can be used to predict the evolution of the velocity and scalar fields. The contraction also distorts the mean scalar gradient according to the geometry of the wind tunnel. For the case of transverse mean gradients, the contraction increases the magnitude of the gradient, and for streamwise mean gradients, which are very rarely studied (Budwig *et al.*, 1985), the mean gradient decreases. It is important that the numerical formulation for DNS of such flows properly takes into account the effect of the mean deformation on the mean scalar gradients, since they explicitly enter the equations used during the computations. This is unlike the equations for the velocity fluctuations, where mean velocity gradients do not explicitly appear following a transformation of variables (Rogallo, 1981).

The rest of this chapter is organized as follows. In §3.1, the mathematical formulation for passive scalars under irrotational mean strain is presented, and the important RDT results for rapid strain rates are summarized. In §3.2, the initial conditions for the strained simulations are described, the generation of which involves conduct-

ing a pre-simulation to develop scalar fluctuations under the presence of a uniform mean gradient. Because of an interest in comparing with GW, the simulations use a Schmidt number (or Prandtl number) of 0.7, which is similar to the Prandtl number for temperature fluctuations in air. Continuing, in §3.3 axisymmetric contraction of the form described in Chapter II is applied to study how scalar statistics, e.g., the scalar spectrum and scalar gradient statistics, evolve as the flow becomes anisotropic. In §3.4, the strain is removed, and the turbulence is allowed to relax back toward a more isotropic state. Finally, §3.5 summarizes the major results from this chapter.

3.1 Mathematical formulation and numerical approach

This section presents the mathematical formulation used for the computations of turbulent mixing under axisymmetric contraction, as well as some of the theoretical results for when the strain rate is very rapid. While the addition of passive scalars requires only straightforward extensions of Rogallo's method (for the computations) or rapid distortion concepts (for the theoretical results), the details are presented here since studies of turbulent mixing under axisymmetric contraction are scarce.

3.1.1 Governing equations in a deforming, anisotropic domain

The evolution of a passive scalar is governed by an advection-diffusion equation, where the instantaneous value (mean plus fluctuation) of the scalar Θ satisfies

$$\frac{\partial \Theta}{\partial t} + U_j \frac{\partial \Theta}{\partial x_j} = D \frac{\partial^2 \Theta}{\partial x_j \partial x_j} \ , \tag{3.1}$$

where U_j is the instantaneous velocity and D is the molecular diffusivity. The instantaneous velocity and scalar fields are decomposed into their mean and fluctuating components as

$$U_i = \langle U_i \rangle + u_i \ , \quad \Theta = \langle \Theta \rangle + \theta \ , \tag{3.2}$$

where $\langle U_i \rangle$ is the mean velocity, $\langle \Theta \rangle$ is the mean of the scalar, u_i is the fluctuating velocity, and θ is the scalar fluctuation. In homogeneous turbulence the mean fields $\langle U_i \rangle$ and $\langle \Theta \rangle$ have spatially uniform (but possibly time-dependent) gradients, and can thus be expressed as

$$\langle U_i \rangle = \frac{\partial \langle U_i \rangle}{\partial x_j} x_j \ , \quad \langle \Theta \rangle = \frac{\partial \langle \Theta \rangle}{\partial x_j} x_j \ . \tag{3.3}$$

Equation (3.1) can be expanded in terms of the mean and fluctuation as

$$\frac{\partial \langle \Theta \rangle}{\partial t} + \frac{\partial \theta}{\partial t} + \langle U_j \rangle \frac{\partial \langle \Theta \rangle}{\partial x_j} + \langle U_j \rangle \frac{\partial \theta}{\partial x_j} + u_j \frac{\partial \langle \Theta \rangle}{\partial x_j} + u_j \frac{\partial \theta}{\partial x_j} = D \frac{\partial^2 \langle \Theta \rangle}{\partial x_j \partial x_j} + D \frac{\partial^2 \theta}{\partial x_j \partial x_j} \ , \tag{3.4}$$

and after taking the mean of (3.4), for homogeneous turbulence the mean satisfies

$$\frac{\partial}{\partial t} \left[\frac{\partial \langle \Theta \rangle}{\partial x_j} \right] + \frac{\partial \langle U_k \rangle}{\partial x_j} \frac{\partial \langle \Theta \rangle}{\partial x_k} = 0 \ . \tag{3.5}$$

Denoting the time-dependent mean scalar gradient by $\langle \Theta_i(t) \rangle \equiv \partial \langle \Theta \rangle / \partial x_i$, (3.5) can be written as

$$\frac{d \langle \Theta_i \rangle}{dt} + \langle \Theta_k \rangle \frac{\partial \langle U_k \rangle}{\partial x_i} = 0 \ , \tag{3.6}$$

which is similar to equation (2.11) in GW. Given the initial mean scalar gradient $\langle \Theta_i \rangle_0$, (3.6) can be integrated in time to update the mean scalar gradient under the application of mean strain. For irrotational mean strain the mean gradient satisfies

$$\langle \Theta_\alpha(t) \rangle = \langle \Theta_\alpha \rangle_0 \exp \left[- \int_0^t \frac{\partial \langle U_\alpha \rangle}{\partial x_\alpha} \, d\tau \right] \ , \tag{3.7}$$

which is exactly like (2.5) for the metric tensor shown in Chapter II. Hence, during the simulation the deformation of the computational domain (expressed through the

grid metrics) can be readily used to update the mean scalar gradients by

$$\frac{\langle \Theta_\alpha(t) \rangle}{\langle \Theta_\alpha \rangle_0} = \frac{B_{\alpha\alpha}(t)}{B^0_{\alpha\alpha}} \ . \tag{3.8}$$

This result is perhaps geometrically intuitive, since the stretching (squeezing) of the computational domain in a given coordinate direction should act to decrease (increase) the mean scalar gradient in that direction.

With the mean scalar field given as a prescribed function of the mean strain (i.e., domain deformation), the equation for the scalar fluctuations must be derived. Subtracting the equation for the scalar mean from (3.4), the scalar fluctuations satisfy

$$\frac{\partial \theta}{\partial t} + \langle U_j \rangle \frac{\partial \theta}{\partial x_j} + u_j \langle \Theta_j \rangle + u_j \frac{\partial \theta}{\partial x_j} = D \frac{\partial^2 \theta}{\partial x_j \partial x_j} \ . \tag{3.9}$$

The challenge in (3.9) is the inclusion of the non-periodic mean flow advection term (second from left), which (if retained) precludes one from using Fourier pseudo-spectral methods for the scalar fluctuations. In similar fashion as for the velocity field, this complication is removed by switching to the deforming coordinate system introduced by Rogallo (1981), which relates the computational coordinates ξ_i with the laboratory coordinates x_i via $\xi_i = B_{ij}(t) x_j$, where $B_{ij}(t)$ is the time-dependent metric tensor. In the transformed coordinates the scalar fluctuations satisfy

$$\frac{\partial \theta}{\partial t} + u_j \langle \Theta_j \rangle + u_j B_{kj} \frac{\partial \theta}{\partial \xi_k} = D B_{kj} B_{lj} \frac{\partial^2 \theta}{\partial \xi_k \partial \xi_l} \ . \tag{3.10}$$

The velocity and scalar fluctuations can now be written as Fourier series in the transformed coordinates as

$$u_i(\boldsymbol{\xi}) = \sum_{\boldsymbol{\kappa}} \hat{u}_i(\boldsymbol{\kappa}) \exp(\mathrm{i}\boldsymbol{\kappa} \cdot \boldsymbol{\xi}) \ , \quad \theta(\boldsymbol{\xi}) = \sum_{\boldsymbol{\kappa}} \hat{\theta}(\boldsymbol{\kappa}) \exp(\mathrm{i}\boldsymbol{\kappa} \cdot \boldsymbol{\xi}) \ , \tag{3.11}$$

which transforms (3.10) to

$$\frac{d\hat{\theta}}{dt} + DB_{kj}B_{lj}\kappa_k\kappa_l\hat{\theta} = \hat{c} - \hat{u}_j\langle\Theta_j\rangle \ , \tag{3.12}$$

where $\hat{c}$ represents the contribution from the nonlinear terms. Molecular diffusion is then integrated exactly by defining the integrating factor

$$F(t) = \exp\left(\int_{t_n}^{t} DB_{kj}(\tau)B_{lj}(\tau)\kappa_k\kappa_l\, d\tau\right) \ , \tag{3.13}$$

where t_n represents the current time level in the computation, so that (3.12) can be written as

$$\frac{dF\hat{\theta}}{dt} = F\cdot(\hat{c} - \hat{u}_j\langle\Theta_j\rangle) \ . \tag{3.14}$$

Note that when using Rogallo's method it is not the velocities $\hat{u}_j$ which are computed, but rather the transformed velocities $\hat{u}^*_\alpha \equiv \hat{u}_\alpha/B_{\alpha\alpha}$, which is used to finalize (3.14) as

$$\frac{dF\hat{\theta}}{dt} = F\cdot(\hat{c} - \langle\Theta_1\rangle B_{11}\hat{u}^*_1 - \langle\Theta_2\rangle B_{22}\hat{u}^*_2 - \langle\Theta_3\rangle B_{33}\hat{u}^*_3) \ . \tag{3.15}$$

During the computations nonlinear terms are formed by calculating scalar gradients in Fourier space, and multiplying them with the velocity field in physical space. The same dealiasing strategies described in Chapter II are used for the current simulations.

3.1.2 Rapid distortion theory for the scalar field

One useful theory in turbulence research is the so-called rapid distortion theory (RDT), which describes the evolution of a turbulent flow subjected to very rapid mean strain rates (Batchelor & Proudman, 1954). The assumption in RDT is that when the mean strain rates are very large, the nonlinear (e.g., turbulent advection) and diffusive terms in the governing equations will not appreciably change the flow during the distortion (which occurs instantaneously when the strain rates are infi-

nite). Such terms are dropped from the governing equations, leaving equations which admit exact solutions. The RDT for the velocity field (Townsend, 1976) was used in Chapter II as a basis for comparison with the DNS results (e.g., see figure 2.9). While RDT for the velocity field is somewhat complex because incompressibility must be enforced by the rapid pressure at all times, as GW point out, RDT as applied to passive scalars is very straightforward.

Beginning directly with the equation for the Fourier coefficients of the scalar fluctuations in the deforming coordinate system given by (3.12), rapid distortion assumptions are used to drop diffusion and turbulent advection terms (the second and third terms, respectively). In this work, only small, finite deformations are considered (e.g., an extension by a factor of 4 in the x_1 direction), so the mean scalar gradients only change by modest factors according to (3.8), and never become "rapid" themselves. Hence, the mean-gradient source terms appearing in (3.12) can be dropped, giving the result that the Fourier coefficients of the scalar field remain unchanged, even though their corresponding wavenumbers are distorted by the mean deformation. As summarized by GW, if one considers the spectral covariance of the passive scalar $E_\theta(\mathbf{k}) = \langle \hat{\theta}^*(\mathbf{k})\hat{\theta}(\mathbf{k})\rangle$, its value (shape) at any point during the deformation can be determined from the pre-strain spectrum by

$$E_\theta(t, \mathbf{k}) = E_\theta(0, \mathbf{k}^0) \ , \tag{3.16}$$

where $\mathbf{k}^0$ is the corresponding initial (pre-strain) wavenumber for the wavenumber $\mathbf{k}$.

Of particular interest in scalar mixing studies are scalar gradient variances, which directly enter the expression for the mean scalar dissipation rate, given by

$$\langle \chi \rangle \equiv 2D \left\langle \frac{\partial \theta}{\partial x_i} \frac{\partial \theta}{\partial x_i} \right\rangle \ . \tag{3.17}$$

Given the previous discussion that the scalar field is only distorted by the mean

strain (i.e., there are no production or dissipation mechanisms changing the scalar field during rapid straining), it perhaps comes to no surprise that the scalar gradient variances can be related to their initial values. Although it can be derived through (3.16), it suffices to simply state the GW result that for a coordinate direction x_α which undergoes a total strain $\exp[f_\alpha(t)]$ (see (2.6) and surrounding discussion), the scalar gradient variance in that direction changes according to

$$\left\langle \frac{\partial\theta}{\partial x_\alpha}\frac{\partial\theta}{\partial x_\alpha} \right\rangle = e^{-2f_\alpha(t)} \left\langle \frac{\partial\theta}{\partial x_\alpha}\frac{\partial\theta}{\partial x_\alpha} \right\rangle_{t=0} . \tag{3.18}$$

Hence, in directions of extensional strain ($f_\alpha(t) > 0$) the scalar gradients weaken, and in directions of compressive strain ($f_\alpha(t) < 0$) the scalar gradients are amplified.

3.2 Pre-simulation and the development of the scalar fluctuations

For the passive scalar simulations, the interest is in studying the effects of multiple strain rate profiles on a single pre-strain initial condition, so as to better understand when the demanding assumptions of RDT become valid for the passive scalar. This contrasts the approach used in Chapter II for the velocity field in which a single non-dimensional mean strain rate profile was used for multiple pre-strain initial conditions (i.e., Reynolds numbers). The initial conditions for the strained simulations are once again generated by conducting a pre-simulation, where the velocity field is treated in the same manner as described in Chapter II. The numerical configuration for the current pre-simulation is the same as Run 7 described in table 2.1, but is augmented with three passive scalars of $Sc = 0.7$ with mean gradients in different coordinate directions, i.e., $\langle\Theta_i\rangle_0 = (1,0,0)$, $\langle\Theta_i\rangle_0 = (0,1,0)$, and $\langle\Theta_i\rangle_0 = (0,0,1)$ for the first, second, and third scalar, respectively. The scalar fluctuations begin from zero initial conditions, and with the axisymmetric contraction being oriented in the x_1 direction, the second and third scalars are expected to remain statistically similar throughout

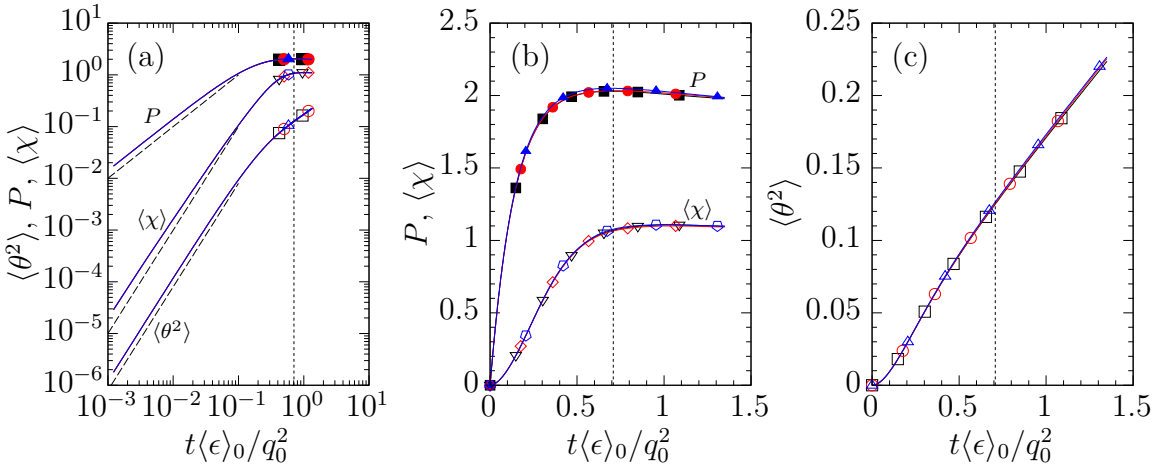

Figure 3.2: Pre-simulation evolution of (a) scalar variance and its budget terms, (b) scalar production rate and mean scalar dissipation rate, and (c) scalar variance. Curves and symbols in black for scalar with mean gradient in x_1, in red for scalar with mean gradient in x_2, and blue for scalar with mean gradient in x_3. Sloped dashed lines in (a) adjacent to scalar variance and mean scalar dissipation rate are proportional to t^2, and dashed line adjacent to scalar production rate proportional to t. Vertical dotted line indicates point at which strain is applied in the strained simulations.

all phases of the simulation, and to show significantly different behavior than the first scalar only after the strain is activated.

Before the mean strain can be applied, the velocity and scalar fields must be allowed to develop and attain characteristics of physical turbulence. In Chapter II for the velocity field, this was accomplished by monitoring small-scale statistics such as the velocity gradient skewness, and comparing them with well-known values in the literature. Because the velocity field during the pre-simulation is similar to Run 7 in Chapter II, velocity field statistics are not presented again. The scalar fluctuations, unlike the velocity fluctuations, begin with zero initial conditions and develop naturally through the interaction of the turbulent velocity fluctuations with the mean scalar gradients. Figure 3.2 presents some quantities related to the development of the scalar variance, which in homogeneous turbulence is governed by the equation

$$\frac{d\langle\theta^2\rangle}{dt} = -2\langle u_i\theta\rangle\frac{\partial\langle\Theta\rangle}{\partial x_i} - \langle\chi\rangle \ , \tag{3.19}$$

where the first term on the right-hand side is the production term (P), and the second is the mean scalar dissipation rate ($\langle\chi\rangle$), previously defined in (3.17). The results for all scalars are very similar due the isotropy of the initial velocity field, and the equal magnitudes of the various mean scalar gradients. When beginning from zero initial conditions, at small times the scalar fluctuations grow in linearly proportion to the simulation time, giving the expected result in frame (a) that the scalar variance and dissipation grow quadratically in time, while the scalar production rate grows linearly in time. Remarkably, in the presence of a uniform mean scalar gradient, the scalar variance is expected to increase monotonically. This was first predicted by Corrsin (1952) for stationary isotropic turbulence, and was later found to be true in decaying turbulence as well (Sullivan, 1976; Sirivat & Warhaft, 1983; Gibson & Dakos, 1993). In frame (b) after the initial development period, it is seen that production continues to exceed dissipation, resulting in the quasi-linear growth for the scalar variance as a function of time in frame (c). Linear growth was predicted and demonstrated by Sullivan (1976), and was also found in the experiments of Sirivat & Warhaft (1983) (albeit as functions of space, with uniform mean velocities).

To better determine if the scalar field is entering a fully-developed state, it is useful to examine non-dimensional quantities characterizing the large and small scales of the scalar fields. Beginning in figure 3.3, in frame (a) the velocity-scalar correlation coefficient, which is related to the production mechanism, approaches a value near -0.7, which is in very good agreement with past experiments (Sirivat & Warhaft, 1983; Budwig *et al.*, 1985). The ratio of scalar variance production to dissipation in frame (b) appears to asymptote to a value near 1.8, which is also in the range of values reported by experiments (Sirivat & Warhaft, 1983; Gibson & Dakos, 1993). Finally, the mechanical-to-scalar timescale ratio shown in frame (c) has also passed through initial transients, approaching a value greater than 2, which is in reasonable agreement with the pre-contraction values reported by GW. The development of the

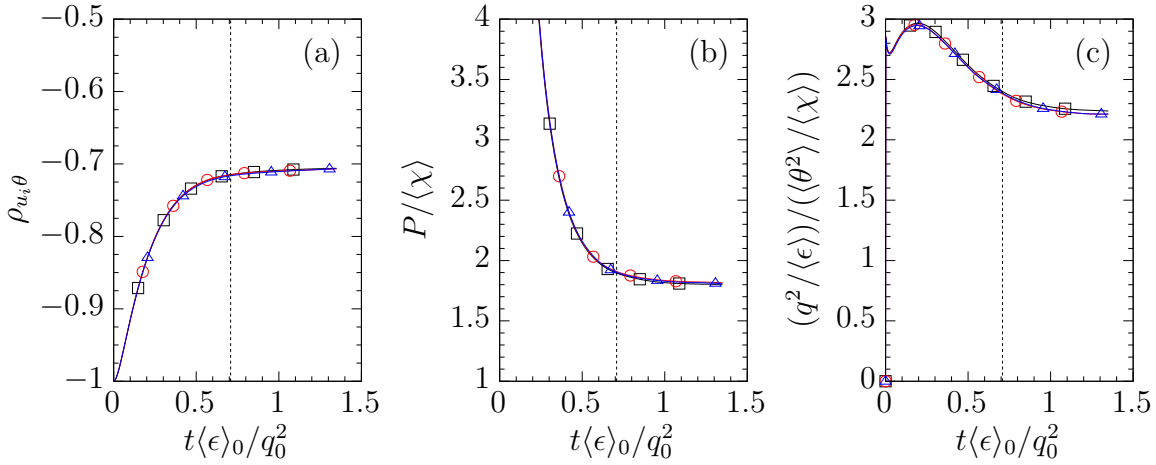

Figure 3.3: Pre-simulation evolution of (a) velocity-scalar correlation coefficient, (b) ratio of scalar production rate to scalar dissipation rate, and (c) mechanical-to-scalar timescale ratio. Black curves with open squares (□) for scalar with mean gradient in x_1, red curves with open circles (○) for scalar with mean gradient in x_2, and blue curves with open triangles (△) for scalar with mean gradient in x_3. Vertical dotted line indicates point at which strain is applied in the strained simulations.

small-scales of the scalar field is examined with scalar gradient statistics, in particular the scalar gradient anisotropy tensor

$$C'_{ij} = \frac{C_{ij}}{C_{kk}} - \frac{1}{3}\delta_{ij} \ , \tag{3.20}$$

where C_{ij} is the fluctuating scalar gradient covariance tensor defined as

$$C_{ij} = \left\langle \frac{\partial\theta}{\partial x_i}\frac{\partial\theta}{\partial x_j} \right\rangle \ . \tag{3.21}$$

Figure 3.4 shows the evolution of the diagonal components of the anisotropy tensor for each scalar during the pre-simulation. After an initial transient, the scalar gradients show a small level of anisotropy, with the trend being that the scalar gradient variance in the direction of the imposed mean gradient is slightly larger than the scalar gradient variances in the other directions. This result is expected, given similar levels of anisotropy in mean-gradient driven scalar fields of moderate Schmidt numbers reported in previous works (Overholt & Pope, 1996; Yeung *et al.*, 2002).

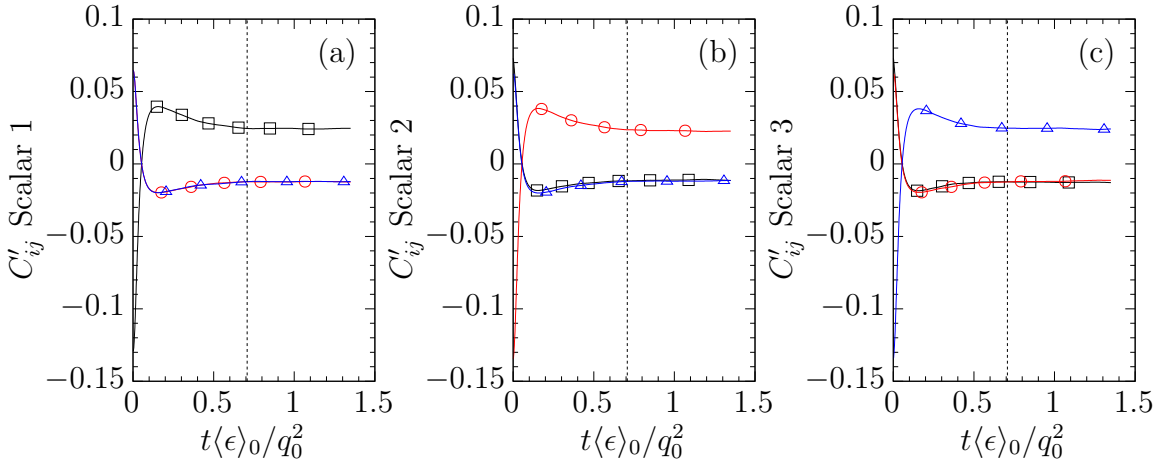

Figure 3.4: Pre-simulation evolution of fluctuating scalar gradient anisotropy tensor for (a) scalar with mean gradient in x_1, (b) scalar with mean gradient in x_2, and (c) scalar with mean gradient in x_3. Black curves with open squares (□) for C'_{11}, red curves with open circles (○) for C'_{22}, and blue curves with open triangles (△) for C'_{33}. Vertical dotted line indicates point at which strain is applied in the strained simulations.

The turbulence at $t \approx 0.7q_0^2/\langle\epsilon\rangle_0$ is taken as the initial condition for the strained simulations, as most of the transient behavior has passed, and the scalar fields (and the velocity field) are entering a fully-developed state.

3.3 Application of strain

In this section the initial conditions described in §3.2 are subjected to mean strain in the form of an axisymmetric contraction that models the GW wind tunnel. Unlike in Chapter II where a single non-dimensional strain rate was applied to multiple pre-strain initial conditions, here the focus is on the application of multiple non-dimensional strain rates to a single initial condition. Such tests allow one to confirm the validity of GW's RDT results for the scalar spectrum and scalar gradients, and to see what strain rates are required to satisfy the stringent assumptions of RDT. Specifically, simulations are conducted using mean velocity profiles with peak non-dimensional strain rates $S_0^* = 25$, 50, 100, and 200.

Because the velocity field is now subjected to multiple strain rates as well, the

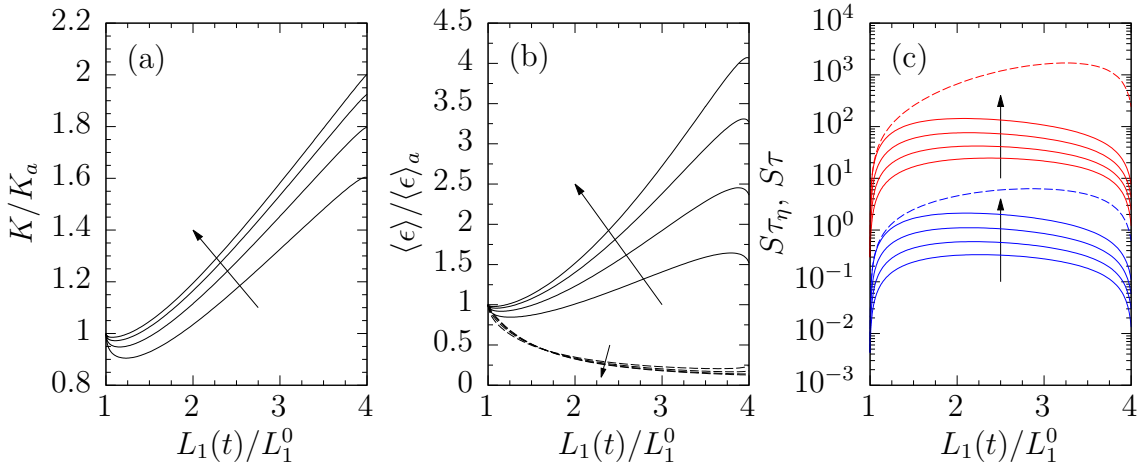

Figure 3.5: Evolution of (a) turbulence kinetic energy, (b) mean energy dissipation rate, and (c) non-dimensional mean strain rates during the application of mean strain. Turbulence kinetic energy and mean energy dissipation rate normalized by pre-contraction values. In (b), solid lines for true mean energy dissipation rate, and dashed lines for mean energy dissipation rate evaluated with isotropic surrogate $\langle\epsilon\rangle_{\text{exp}} = 5\nu(\langle u_{1,1}^2\rangle + \langle u_{2,1}^2\rangle)$. In (c), mean strain rate normalized by large-eddy turnover time τ (upper red curves) and Kolmogorov time scale τ_η (lower blue curves). Dashed lines in right frame formed by normalizing the strongest strain rate profile with τ and τ_η formed with $\langle\epsilon\rangle_{\text{exp}}$. In all frames, peak mean strain rate of mean velocity profile increasing in the directions of the arrows.

evolution of the turbulence kinetic energy, mean energy dissipation rate, and non-dimensional strain rates are briefly presented in figure 3.5. In this and the following figures, pre-contraction quantities used for normalization are marked with a subscript a. Recall that in the numerical wind tunnel, the mean strain rate initially increases very gradually, with the result being that the turbulence continues to decay early in the straining period. In agreement with expectations and RDT (Pope, 2000), as the strain rate increases, the amplification of the turbulence kinetic energy and dissipation rate also increases. Because part of the effort is to better understand the GW experiments, figure 3.5 also includes some quantities as they would be calculated with the experimental techniques. In particular, in frame (b) the mean energy dissipation rate, given by

$$\langle\epsilon\rangle = 2\nu\langle s_{ij}s_{ij}\rangle \ , \tag{3.22}$$

is presented, which is challenging to measure experimentally because many velocity derivative variances and covariances must be measured simultaneously. GW employ an isotropic surrogate of the form

$$\langle \epsilon \rangle_{\text{exp}} = 5\nu \left[\left\langle \left(\frac{\partial u_1}{\partial x_1} \right)^2 \right\rangle + \left\langle \left(\frac{\partial u_2}{\partial x_1} \right)^2 \right\rangle \right] , \tag{3.23}$$

which underestimates the mean energy dissipation rate because velocity gradients in the extensional direction are severely suppressed, while gradients in the compressive directions are amplified (see discussion near the end of §2.3.1). The errors in the surrogate impact the assessment of the non-dimensional strength of the wind tunnel contraction in frame (c), leading to an over prediction for the non-dimensional strain rates. In the future, axisymmetric surrogates for the energy dissipation rate, perhaps formed with relations derived by George & Hussein (1991) involving gradients in the compressive directions, might improve the estimates.

Continuing, the evolution of the scalars under axisymmetric contraction is examined by plotting the mean gradients, scalar variances, and production and dissipation rates in figure 3.6. In the figure, the top row contains results for the scalar with mean gradient in the x_1 direction, and the bottom row contains results for the scalars with mean gradients in the transverse directions. The evolution of the mean scalar gradients (left column) simply follows the geometry of the distortion caused by the mean strain: the mean scalar gradient in the x_1 direction in frame (a) decreases by a factor of 4 as the domain elongates in the x_1 direction, and the mean scalar gradients in the transverse directions shown in frame (d) increase by a factor of 2 as the domain contracts in the x_2 and x_3 directions. As discussed previously, under rapid distortion there is no physical mechanism (production or dissipation) which can act to appreciably change the scalar variance. This is borne out in frames (b) and (e), which show that as the peak strain rate is increased from $S_0^* = 25$ to 200,

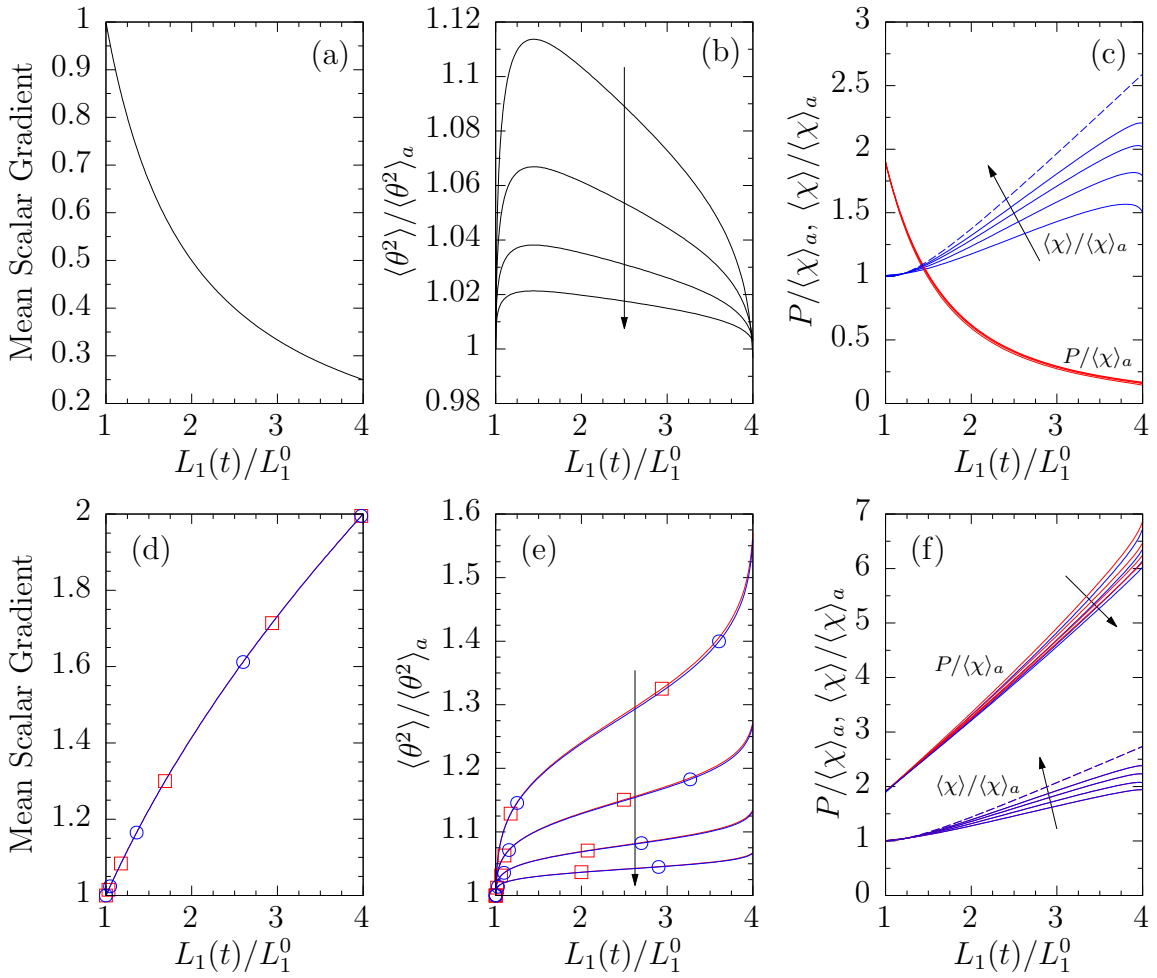

Figure 3.6: Evolution of (left column) the magnitude of the mean scalar gradient, (middle column) the scalar variance, and (right column) the production rate and mean dissipation rate for each scalar. Pre-contraction values used for normalization marked with subscript a. Top row for scalar with mean gradient in x_1, and bottom row for scalars with mean gradients in x_2 (red curves) and x_3 (blue curves). In right column, dashed lines for mean scalar dissipation rate calculated with RDT. In bottom left and middle frames, open red squares (□) for scalar with mean gradient in x_2, and open blue circles (○) for scalar with mean gradient in x_3. Peak mean strain rate increasing in the directions of the arrows.

the scalar variance deviates less and less from the pre-contraction value. Frames (c) and (f) show the production and dissipation rates of the scalar variance normalized by the initial value of the scalar dissipation rate. The overall trend for the scalar dissipation rate to increase during the contraction is expected, given that scalar gradients in the transverse directions are amplified during the contraction. (This can

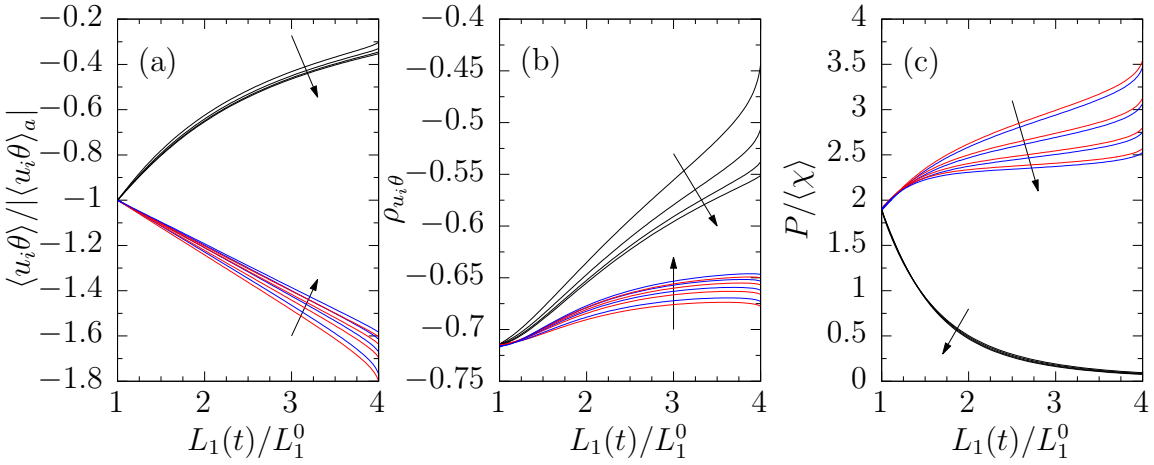

Figure 3.7: Evolution of (a) scalar flux in the direction of the mean scalar gradient normalized by the magnitude (so as to retain the negative sign) of the initial value for each respective scalar, (b) velocity-scalar correlation coefficient, and (c) ratio of scalar production rate to mean scalar dissipation rate during the application of strain. Black curves for scalar with mean gradient in x_1, red curves for scalar with mean gradient in x_2, and blue curves for scalar with mean gradient in x_3. Peak mean strain rate of mean velocity profile increasing in the directions of the arrows.

also be inferred by taking the trace of (3.18) assuming rapid distortion applies and that the initial scalar gradient variances are nominally isotropic.) Where a significant difference emerges between the scalars is the production rate. As seen in frame (c), for the scalar with mean gradient in x_1, the production rate drops by approximately a factor of 10, while in frame (f) the production rates for scalars with transverse mean gradients increase by factors of roughly 3. Of course, given the form for the production term in (3.19), any changes in the mean gradients for the scalars play an important role in the evolution of the production rate, but the significant drop in the production rate for the scalar with mean gradient in x_1 suggests that the scalar flux is also changing appreciably, which is examined next.

The investigation into the factors which influence the scalar variance is continued in figure 3.7. Shown in frame (a) is the evolution of the scalar flux for each scalar, normalized by the magnitude of the pre-contraction value. For scalars with mean gradients in the transverse directions, the scalar flux is strengthened, which when

coupled with the fact that the transverse mean scalar gradients increase during the contraction, leads to the overall increase in the production rate observed in frame (f) of figure 3.6. For the scalar with mean gradient in x_1, the picture is quite different. There is a significant weakening of the scalar flux, which exacerbates the effect that the reduction in the mean scalar gradient had on the production rate. Frame (b) of figure 3.7 shows that there is a reduction in the strength of the velocity-scalar correlation for all scalars, with the de-correlation being most significant for the scalar with mean gradient in x_1. In frame (c) the same results as frames (c) and (f) of figure 3.6 are presented, but now in the form of the ratio of the production rate to the scalar dissipation rate. It appears that in the RDT limit the ratio of production to dissipation for scalars with transverse mean gradients is not significantly altered from the pre-contraction value. For the scalar with a streamwise mean gradient, however, the ratio of production to dissipation drops below 1 early during the application of strain, and reaches a minimum value near 0.1 after the full application of strain. During the application of rapid strain, such changes in the production-to-dissipation ratio are immaterial, because the scalar variance is held constant during straining; however, this effect will clearly be felt in the post-contraction region of the flow. In fact, for some finite amount of time following the contraction there will be a significant "memory effect" in the flow, which will be the continued destruction of the scalar variance for the scalar with mean gradient in the streamwise direction.

Figure 3.6 showed that the DNS appear to support the RDT prediction that under very rapid straining, the scalar variance should remain constant. The scalar spectrum is now examined to verify that its shape as a function of the pre-contraction wavenumbers is preserved under very rapid straining. Much like in Chapter II, the focus is on the 1-D longitudinal spectrum because it is most readily measured in experiments, and was reported by GW. Similar to the presentation of the 1-D component velocity spectra, the 1-D scalar spectrum is presented as a function of the pre-contraction

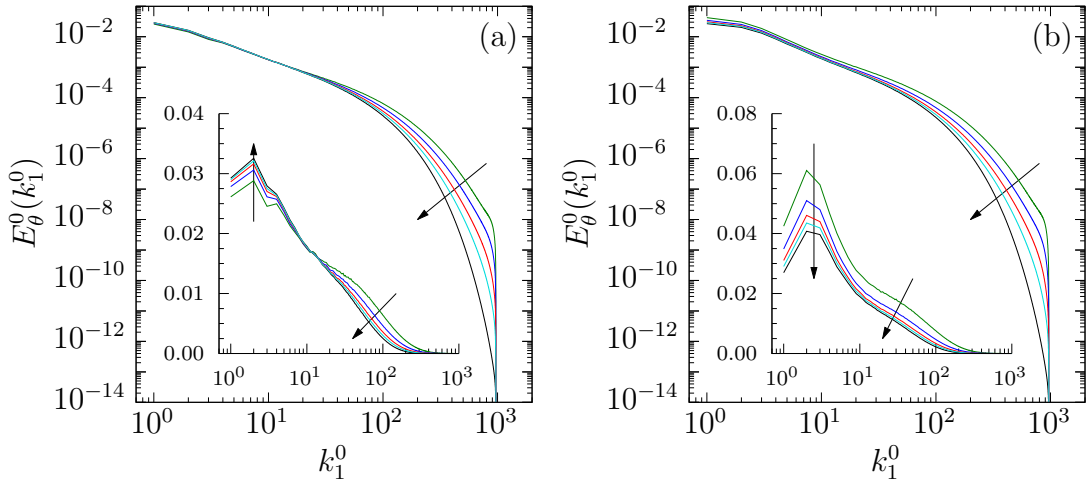

Figure 3.8: Comparison of pre- and post-contraction 1-D spectra for (a) passive scalar with mean gradient in the x_1 direction and (b) passive scalars with mean gradient perpendicular to x_1. The spectra in (b) are averaged over scalars with mean gradients in x_2 and x_3. Spectra shown as functions of pre-contraction wavenumbers. Insets show the compensated spectra of the form $k^0_1 E^0_\theta(k^0_1)$. Solid black lines for pre-contraction spectra and colored lines for post-contraction spectra from runs with different strain rate profiles: green for peak strain rate $S^* = 25$, blue for $S^* = 50$, red for $S^* = 100$, and cyan for $S^* = 200$. Arrows drawn in the direction of increasing strain rate (black curve excluded).

wavenumbers k^0_1 as

$$E^0_\theta(k^0_1) = E_\theta(k_1) B_{11}(t)/B^0_{11} \ , \tag{3.24}$$

where the evolution of the time-dependent grid metric ratio $B^0_{11}/B_{11}(t)$ indicates the extent of deformation (elongation) in the x_1 direction. Figure 3.8 presents the pre- and post-contraction 1-D spectra for the scalar with mean gradient in x_1 in frame (a), and for the scalars with transverse mean gradients in frame (b). The insets include the compensated spectra, the area under which is the scalar variance. In frame (a) it is seen that the scalar variance is never significantly changed from the pre-contraction value for the case of a mean gradient in x_1, which is expected given the result in frame (b) of figure 3.6. For the scalars with transverse mean gradients, in frame (b) there is an increase in the scalar spectrum over all wavenumbers, resulting in the

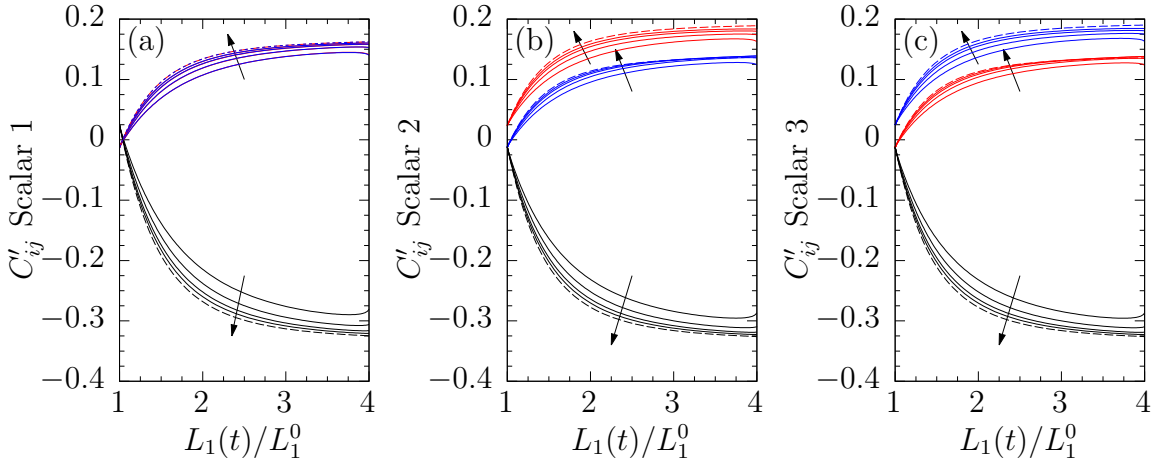

Figure 3.9: Evolution of scalar gradient anisotropy tensor during the application of strain for (left) scalar with mean gradient in x_1, (middle) scalar with mean gradient in x_2, and (right) scalar with mean gradient in x_3. Black curves for C'_{11}, red curves for C'_{22}, and blue curves for C'_{33}. Peak mean strain rate of mean velocity profile increasing in the directions of the arrows. Dashed curves calculated with RDT.

increased post-contraction scalar variance shown earlier in frame (e) of figure 3.6. In figure 3.8, the arrows are drawn in the direction of increasing strain rate, and a robust trend is that as the strain rate is increased, the spectrum agrees more with its pre-contraction shape, as predicted by RDT. At all strain rates, the simulations show deviations from RDT at high wavenumbers, in agreement with the experimental results reported by GW (see their figure 9). Such disagreement is expected, as even at the highest strain rates considered in this study, the strain rates are not very rapid compared to the small scales, as shown in frame (c) of figure 3.5.

Thus far single-point statistics and spectra have been examined, but not the development of anisotropy in the scalar field. Anisotropy in the scalar field, particularly at the small scales, can be understood by examining statistics of fluctuating scalar gradients, the first non-zero statistics of which appear at the second-order. Figure 3.9 presents the evolution of the fluctuating scalar gradient anisotropy tensor during the application of strain for each of the three scalars. The expectation that scalar gradients in the transverse directions should be amplified, while scalar gradients in the streamwise direction should decrease is satisfied for all scalars. It is also clear that

the slight anisotropy in the scalar gradients before the application of strain shown previously in figure 3.4 affects the post-contraction axisymmetry (or lack thereof) of the scalar gradients. For the scalar with mean gradient in x_1, the scalar gradients shown in frame (a) remain axisymmetric as gradients in the transverse directions are amplified equally. However, for the scalars with transverse mean gradients shown in frames (b) and (c), while both transverse fluctuating scalar gradients are amplified, their slightly anisotropic initial conditions result in a loss of axisymmetry as the strain is applied. Figure 3.9 also includes in the form of dashed curves the predicted evolution of the scalar gradient anisotropy tensor using fluctuating scalar gradient variances calculated according to the RDT result in (3.18). In all cases, the agreement with RDT improves as the strain rate is increased. The evolution of the scalar gradient anisotropy reported here is very similar to the result of GW (their figure 7), although they assumed perfectly isotropic initial conditions for the fluctuating scalar gradients when calculating the RDT-predicted evolution of the scalar gradient anisotropy.

3.4 Relaxation of axisymmetric turbulence

Now that strain has been applied to the turbulence to achieve a total strain (elongation) of 4 in the x_1 direction, it is released to allow the turbulence to relax towards a more isotropic state. Here the focus is on the single configuration with a peak non-dimensional strain rate $S_0^* = 25$, which contains a velocity field that is very similar to that studied in Chapter II. In §3.3 the application of strain was shown to lead to the development of anisotropy in the scalar field, and that significant differences emerge in certain large-scale quantities for scalars with mean gradients in different directions. Specifically, the scalar with mean gradient in the streamwise direction experienced a dramatic reduction in its production rate, while scalars with transverse mean gradients had their production rates amplified through the contraction. Such differences between scalars with mean gradients in different directions are then

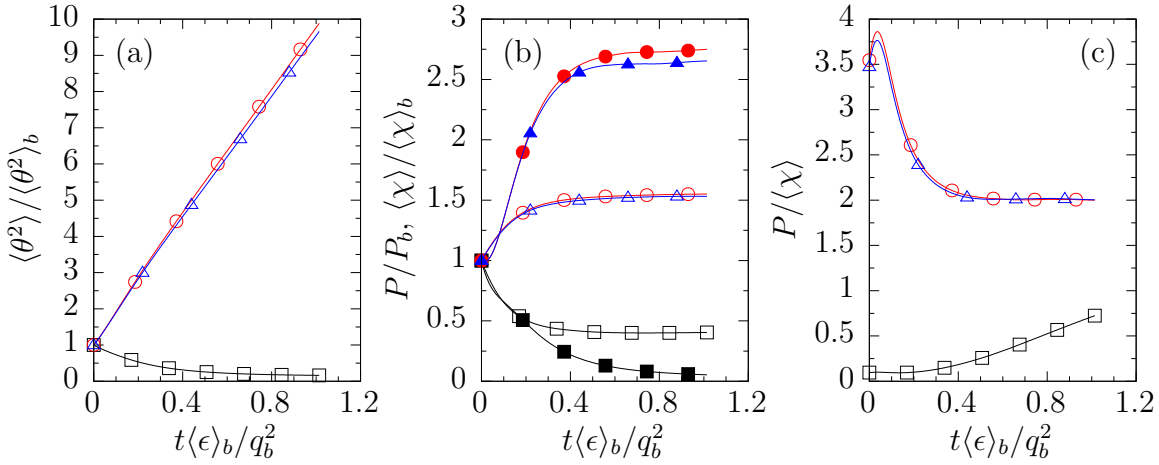

Figure 3.10: Post-contraction relaxation of (a) normalized scalar variance, (b) normalized scalar production rate (open symbols) and normalized mean scalar dissipation rate (closed symbols), and (c) ratio of production to dissipation. Black curves with squares (□ and ■) for scalar with mean gradient in x_1, red curves with circles (○ and ●) for scalar with mean gradient in x_2, and blue curves with triangles (△ and ▲) for scalar with mean gradient in x_3.

expected to persist in the post-contraction region.

Figure 3.10 begins with a presentation of the scalar variance evolution in the post-contraction flow, along with its production and dissipation rates. Following the notation of Chapter II, when normalizing quantities a subscript b denotes the value taken immediately after the application of strain, just at the start of relaxation. In frame (a), the post-contraction evolution of the scalar variance depends strongly on the direction of the mean gradient. While the scalars with transverse mean gradients quickly resume a quasi-linear growth rate like in the pre-contraction flow, the scalar with mean gradient in the streamwise direction experiences a continued destruction of scalar variance for at least one large-eddy turnover time into the relaxation. These behaviors are explained by examining the production and dissipation rates, as shown in frames (b) and (c). In frame (b), the production and dissipation rates for scalars with transverse mean gradients quickly adjust, such that the ratio of production to dissipation shown in frame (c) becomes close to 2. While the production rate for the scalar with mean gradient in the streamwise direction appears to level off in frame (b),

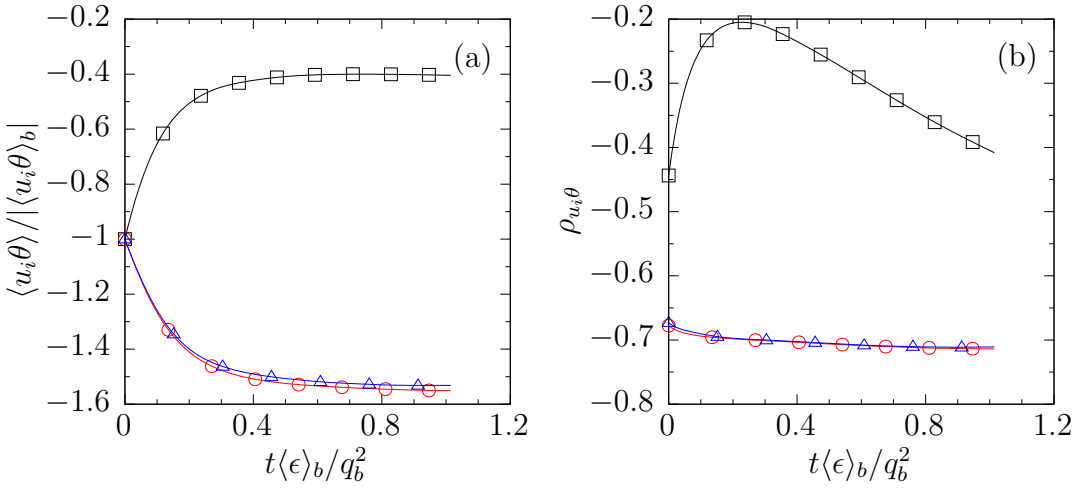

Figure 3.11: Post-contraction relaxation of (a) normalized scalar flux in the direction of the mean gradient and (b) velocity-scalar correlation coefficient. Black curves with open squares (□) for scalar with mean gradient in x_1, red curves with open circles (○) for scalar with mean gradient in x_2, and blue curves with open triangles (△) for scalar with mean gradient in x_3.

it appears that the slow decay of the scalar dissipation rate results in a production-to-dissipation ratio that slowly increases in frame (c). Although the production rates shown in frame (b) contain all of the information of the scalar flux, the normalized scalar flux is also shown in frame (a) of figure 3.11, which also includes the velocity-scalar correlation coefficient in frame (b). For scalars with transverse mean gradients, the correlation coefficient quickly takes on the nominal pre-contraction value; however, there is a very slow transient of the correlation coefficient for the scalar with a streamwise mean gradient. Clearly, longer simulations (potentially on larger domains) are required to determine how the first scalar equilibrates in the post-contraction flow. Perhaps after a much longer transient period the correlation coefficient will approach the nominal values attained by the other scalars.

The relaxation of the scalar gradients is shown in figure 3.12. In frames (b) and (c), the nominal amount of pre-strain anisotropy in the scalar gradients is quickly attained for the scalars with transverse mean gradients. GW show a qualitatively similar relaxation of the scalar gradients, however, their results are shown as a function of a thermal timescale. To perform a closer comparison with the work of GW, and to assess

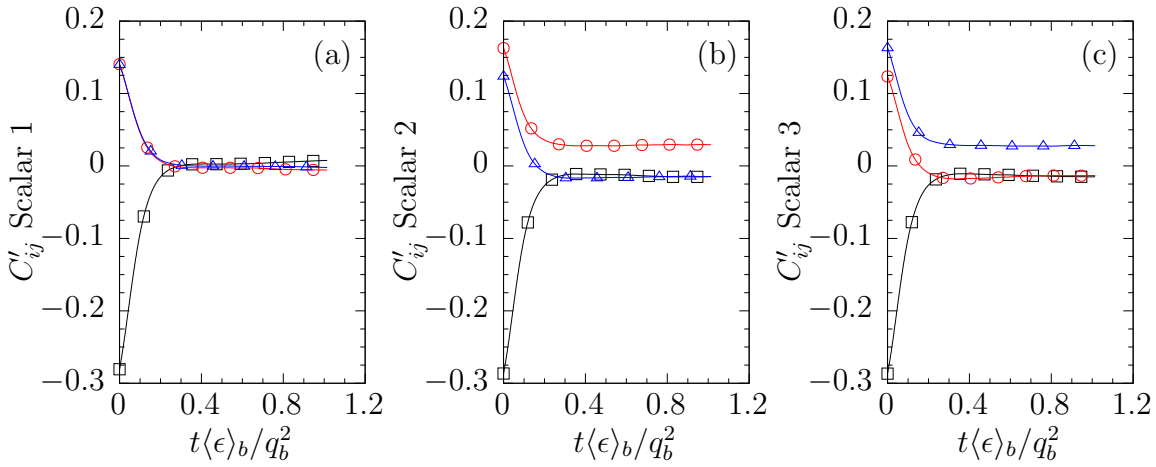

Figure 3.12: Post-contraction relaxation of scalar gradient anisotropy tensor for (a) scalar with mean gradient in x_1, (b) scalar with mean gradient in x_2, and (c) scalar with mean gradient in x_3. Black curves with open squares (□) for C'_{11}, red curves with open circles (○) for C'_{22}, and blue curves with open triangles (△) for C'_{33}.

how well their models capture the evolution of the DNS data, the thermal timescale reported below equation (2.22) in their work should be calculated and incorporated into the present results. Similar to the slow evolution of the large scale quantities in figures 3.10 and 3.11, the relaxation of the small scales is very slow for the scalar with a streamwise mean gradient.

Finally, the relaxation of the 1-D scalar spectrum is presented in figure 3.13. Like GW, the various spectra are normalized by the instantaneous value of the scalar variance, which allows one to better understand the changes in the shape of the spectrum. Similar to the results reported by AW for the component velocity spectra, GW reported a bump appearing the scalar spectrum at high k_1 wavenumbers following the contraction. The results in frame (b), which are similar to GW in that transverse mean gradients are used, shows very similar behavior (see their figure 13). The increased spectral content at high wavenumbers is related to the production of scalar gradients in the x_1 direction, which was heavily suppressed during the contraction (e.g., see the negative values that develop for C'_{11} during the contraction in figure 3.9). The spectrum for the scalar with a streamwise mean gradient also shows a rapid

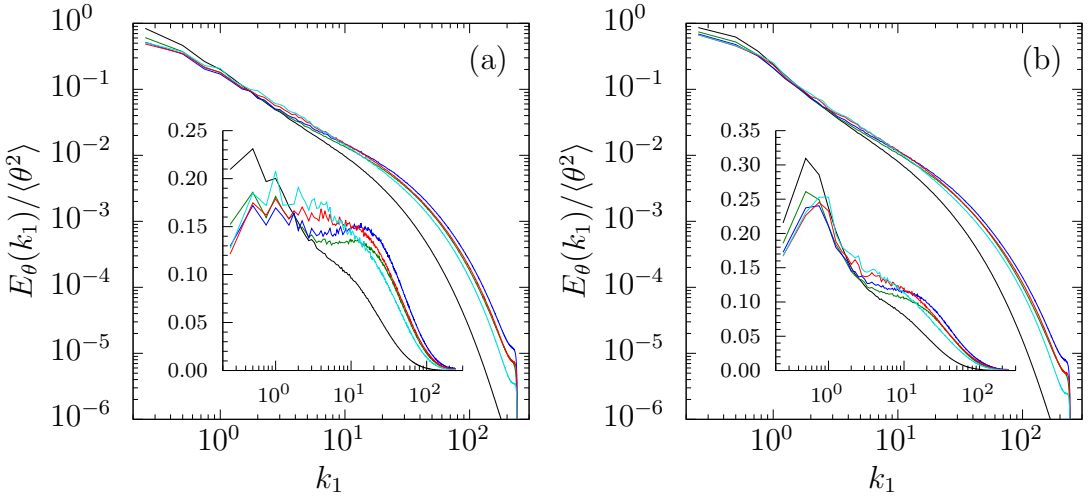

Figure 3.13: Post-contraction relaxation of 1-D scalar spectrum normalized by the instantaneous scalar variance for (a) scalar with mean gradient in x_1 and (b) scalars with mean gradients in x_2 and x_3 (averaged between the two scalars). Curves taken at times $t/\tau_b = 0$ (black), $t/\tau_b = 0.1$ (green), $t/\tau_b = 0.2$ (blue), $t/\tau_b = 0.4$ (red), and $t/\tau_b = 0.6$ (cyan) into the relaxation period. Insets show the spectra multiplied by the wavenumber k_1, e.g., $k_1 E_\theta(k_1)/\langle\theta^2\rangle$.

increase at high wavenumbers, but attains a shape at low wavenumbers which appears fundamentally different than the pre-contraction spectra shown in figure 3.8. Early in the relaxation period (e.g., the green and blue curves at $t/\tau = 0.1$ and $t/\tau = 0.2$, respectively), the scalar spectrum develops a more intricate shape, with a double-peak structure that strongly resembles the AW result for the transverse 1-D velocity spectrum. Additional analysis of the scalar spectral budget is required to understand the development of the spectrum in the intermediate wavenumber range where the double-peak occurs, where production and spectral transfer are expected to play an important role.

3.5 Summary

This chapter presents a study of turbulent mixing under axisymmetric contraction, with an emphasis on achieving similar conditions as the experiments of Gylfason &

Warhaft (2009). Because the GW wind tunnel is the same as that used by Ayyalasomayajula & Warhaft (2006), the simulations begin with a velocity field simulated in the same manner as detailed in Chapter II, with the addition of three passive scalars of $Sc = 0.7$ with uniform mean scalar gradients in different coordinate directions. During a pre-simulation, both the velocity and scalar fields attain characteristics of physical turbulence; while the velocity field decays (see Chapter II), the scalar fluctuations develop naturally under the presence of mean scalar gradients. Following the pre-simulation, mean strain is applied in the form of an axisymmetric contraction. The results show that rapid distortion theory predictions for the evolution of the scalar variance, scalar spectrum, and scalar gradient anisotropy hold increasingly well as the strain rate is increased. After a mean deformation (elongation) of 4 in the streamwise (x_1) direction is achieved, the strain is removed and the flow is allowed to relax back toward a more isotropic state.

One important finding from these simulations is that the direction of the imposed mean scalar gradient has a profound impact on the evolution of the scalar in the post-contraction flow. While the GW experiments only consider the case of a transverse mean scalar gradient, this study also includes a scalar with a mean streamwise scalar gradient. Although mean streamwise gradients are rarely studied (Budwig *et al.*, 1985), they are relevant in industrial applications in which a scalar is injected at varying rates from a fixed location into a flow. For the case of transverse mean scalar gradients, the contraction alters large-scale and small-scale quantities, e.g., the scalar variance production rate and the scalar gradient anisotropy, but when the strain is removed they quickly relax to values (in a non-dimensional sense) not too-far removed from those in the pre-contraction flow. It is the scalar with a mean streamwise scalar gradient which behaves in a qualitatively different manner. During the application of strain, scalar gradients are amplified to increase the scalar dissipation rate, much like for the other scalars, but the production rate is dramatically reduced through the

contraction (figure 3.6). This is the result of the mean scalar gradient being reduced during the deformation, the effect of which is compounded by a reduction in the scalar flux (figure 3.7). As a result of the reduced production rate and increased dissipation rate, the scalar variance is destroyed in the post-contraction flow for a significant amount of time (figure 3.10). Furthermore, the relaxation of this scalar appears to be much slower than the other scalars, for both large-scale and small-scale statistics.

At this point the capability to simulate passive scalars under axisymmetric contraction has been verified, and the simulations provide support for many of the RDT results derived by GW. Future efforts will focus on understanding the evolution of the scalar spectrum through spectral budgets, in the same way as was done in Chapter II for the component velocity spectra. One important quantity requiring additional attention is the scalar flux, which must be understood to develop models for turbulent mixing. The scalar flux, and its non-dimensional relative the velocity-scalar correlation coefficient, were observed to change in a non-trivial manner for the scalar with a streamwise mean scalar gradient. The budget equation for the scalar flux should be examined, as well as the scalar flux spectrum, to gain more insight into the scalar flux evolution. Also, the current study is limited to scalars with moderate Schmidt numbers. In many applications, especially in liquid mixing, the Schmidt number can be very high, which places the additional burden on the simulations that the Batchelor scales of the scalar field must be resolved. The next two chapters are focused on the development and use of a new parallel algorithm to study turbulent mixing at high Schmidt number.

CHAPTER IV

ALGORITHMS FOR PETASCALE SIMULATIONS OF TURBULENT MIXING AT HIGH SCHMIDT NUMBER

The study of the evolution of a high Schmidt number passive scalar in a turbulent flow is challenging for both experiments and DNS due to the demanding resolution requirements of the Batchelor scale η_B, which is $\sqrt{Sc}$ times smaller than the Kolmogorov scale of the velocity field. At very high Sc, when the separation of scales between the velocity and scalar fields is wide, it is computationally efficient in DNS to decouple the numerical methods and computational grids used for the velocity and scalar fields (Gotoh *et al.*, 2012). Instead of using a communication-intensive FPS scheme to compute both the velocity and scalar fields on a fine grid that resolves the Batchelor scale, dramatic savings can be obtained by computing the velocity field on a coarser grid of N_v^3 points which resolves the Kolmogorov scale, and then interpolating it to a finer grid of N_θ^3 points used for the scalar, when needed. The governing equation for a passive scalar (being an advection-diffusion equation) is also fundamentally different than the Navier-Stokes equations for the incompressible velocity field in that a global coupling term like the hydrodynamic pressure is not present. Because of the locality of the passive scalar equation, it is more efficient to employ numerical methods which do not require communication-intensive implementations like the FPS scheme. In particular, high-order combined compact finite differences, which compute first and second derivatives simultaneously, are suitable for this task.

In this chapter the ideas of Gotoh *et al.* (2012) are extended to include the physics of high Sc turbulent mixing as a driving factor in the design of a parallel code capable of simulating mixing at petascale problem sizes. At high Sc, due to the disparate computational requirements of the velocity and scalar fields, it is natural to separate

the processes which compute the evolution of the two fields into disjoint groups. The groups (i.e., communicators) are derived from the global set of processes used for the combined simulation (i.e., `MPI_COMM_WORLD`), and the size of each group is easily matched to the size of its problem. Because the passive scalar physics is one-way coupled, the process groups are also one-way coupled: the communicator computing the scalar field must retrieve the velocity field from the velocity communicator during time integration. A code based on these principles is developed from existing FPS and CCD code bases, and an emphasis is placed on the scalability of the algorithm to the core counts required for petascale simulations of turbulent mixing. The resulting code can run in homogeneous or heterogeneous computing environments, and makes heavy use of OpenMP in both to improve overall scalability.

Much of what is presented in this chapter is available in the published work (Clay *et al.*, 2017) described in Appendix C. The sections below begin with a description of the DNS algorithm for homogeneous computation (i.e., the CPU-only code), and then present how the code was ported to run in heterogeneous computing environments. In §4.1, background information is provided on the numerical schemes used for the governing equations for the fluctuating velocity and scalar fields. Since the computation of derivatives using the CCD scheme is the most expensive operation overall, §4.2 first focuses on the performance of several different implementations of the CCD routines. Scaling data is obtained from a kernel code for a wide range of problem sizes and number of processing elements (PEs, defined as the product of the number of MPI processes and the number of OpenMP threads per MPI process). Then §4.3 describes how computations for the velocity and scalar fields are combined within a dual-communicator approach. The acceleration of the code in heterogeneous computing environments using OpenMP 4.5 is discussed in §4.4. Finally, §4.5 concludes with a summary of what was achieved.

4.1 Governing equations and numerical method

As noted earlier, in the algorithm for turbulent mixing at high Sc a FPS scheme is used to compute the velocity field and a CCD scheme is used to compute the scalar field on periodic domains with N_v^3 and N_θ^3 grid points, respectively. This section provides a brief discussion of the numerical method and the parallel algorithm used for each scheme separately. A later section discusses how the two schemes are merged together to form a single code.

4.1.1 FPS scheme for the velocity field

The Navier-Stokes equations for conservation of mass and momentum in a viscous fluid are well known. For turbulent flow of a fluid with constant density (ρ) and kinematic viscosity (ν) and no mean velocity, the governing equations are

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{4.1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} + f_i \ , \tag{4.2}$$

where u_i is the fluctuating velocity, p is the fluctuating pressure, and f_i may represent an additional force of physical or numerical nature. Spectral methods are known for high accuracy (Canuto *et al.*, 2006) and are also very natural if information on scale size is desired. Equations (4.1) and (4.2) are transformed to Fourier (wavenumber) space, where the velocity Fourier coefficient $\hat{u}_i(k_i, t)$ (with k_i being the wavenumber vector, of magnitude $k = |k_i|$, carets denoting Fourier transforms, an no summation being implied when k_i appears as either a functional argument or a subscript) satisfies

$$\frac{\partial \hat{u}_i}{\partial t} = -ik_j \widehat{u_i u_j}|_{\perp k_i} - \nu k^2 \hat{u}_i + \hat{f}_i \ , \tag{4.3}$$

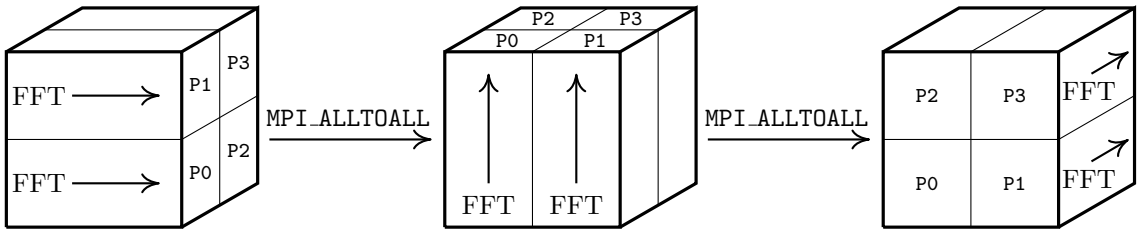

Figure 4.1: A schematic showing the 2-D domain decomposition and transposes required for a 3-D FFT from physical to wavenumber space (or vice versa). For simplicity the case of a 2×2 processor grid is shown, with processes labeled `P0` to `P3`. After each of the first two 1-D transforms a transpose via `MPI_ALLTOALL` communication is performed to regroup data into pencils along the next direction to be transformed.

where the Fourier transform of the nonlinear dyadic product $u_i u_j$ is projected onto a plane perpendicular to k_i. In the pseudo-spectral approach costly convolution sums are avoided in favor of multiplication in physical space followed by transformation to wavenumber space. Double and triple aliased contributions stemming from the nonlinear term are eliminated with a spectral truncation radius $k_{max} = \sqrt{2}N_v/3$ for a N_v^3 cubic domain of size $(2\pi)^3$ (Rogallo, 1981). The viscous term in (4.3) is treated exactly by an integrating factor. Time integration is done with the classical fourth-order Runge-Kutta (RK4) scheme, which offers good stability for the scalar field at high Sc. Courant number constraints based on the finer scalar grid for numerical stability are observed in choosing the time step Δt.

The heart of the pseudo-spectral algorithm is the 3-D fast Fourier transform (FFT). In the code FFTs are taken in each coordinate direction by calling 1-D FFT library routines (Frigo & Johnson, 2005). Memory transposes needed to align the data in different coordinate directions are accomplished with calls to `MPI_ALLTOALL` using a 2-D domain decomposition. Figure 4.1 shows an example in which the 3-D solution domain is distributed over $M_v = 4$ parallel processes using a 2-D domain decomposition. The data are stored as "pencils" and FFTs in each direction are taken with stride-one arrays (Donzis *et al.*, 2008). A complete 3-D transform requires two transposes between pencils in different orientations. The "row" and "column"

communicators used for the `MPI_ALLTOALL` calls are formed within the velocity field communicator.

4.1.2 CCD scheme for the scalar field

The scalar fluctuations θ evolve according to an advection-diffusion equation

$$\frac{\partial \theta}{\partial t} + u_i \frac{\partial \theta}{\partial x_i} = D \frac{\partial^2 \theta}{\partial x_i \partial x_i} + f_\theta \ , \tag{4.4}$$

where D is the molecular diffusivity and f_θ is a forcing term. However, to improve conservation properties for the scalar variance the skew-symmetric form (Durran, 2010, pp. 185–187) is employed, given by

$$\frac{\partial \theta}{\partial t} + \frac{1}{2}\left[u_i \frac{\partial \theta}{\partial x_i} + \frac{\partial u_i \theta}{\partial x_i} \right] = D \frac{\partial^2 \theta}{\partial x_i \partial x_i} + f_\theta \ . \tag{4.5}$$

A spatially uniform mean scalar gradient maintains the fluctuations, which gives $f_\theta = -u_i \partial \langle \Theta \rangle / \partial x_i$ with the mean scalar field $\langle \Theta \rangle$ fixed in time. Because the velocity field is on a coarse grid, in order to form the advective and mean-gradient terms the velocity must first be received from the velocity communicator and then interpolated onto the finer scalar grid. Tricubic interpolation is sufficiently accurate (Gotoh *et al.*, 2012). Integration in time is by the same explicit RK4 scheme described earlier for the velocity field. However, the scalar field is advanced in time in physical space instead of wavenumber space. It is thus necessary to calculate both first and second derivatives of the scalar (the latter appearing in the Laplacian operator) accurately.

The general properties of high-order compact finite difference (CD) schemes are well established (Lele, 1992). Here, the combined compact finite difference (CCD) scheme refers to methods that calculate both first and second derivatives simultaneously (Mahesh, 1998). Based on numerical tests reported by Gotoh *et al.* (2012), eighth-order CCD schemes are expected to have accuracy comparable to spectral

methods. For a signal f defined on a 1-D grid of N_θ points with uniform spacing $h = x_{i+1} - x_i$, the first and second derivatives f'_i and f''_i are determined by a system of simultaneous equations consisting of (for $i = 1, 2, \ldots, N_\theta$)

$$\begin{aligned} 51f'_{i-1} + 108f'_i + 51f'_{i+1} + 9h(f''_{i-1} - f''_{i+1}) &= \frac{107}{h}(f_{i+1} - f_{i-1}) - \frac{f_{i+2} - f_{i-2}}{h} \ , \\ 138(f'_{i+1} - f'_{i-1}) - h(18f''_{i-1} - 108f''_i + 18f''_{i+1}) &= \\ -\frac{f_{i+2} + f_{i-2}}{h} + \frac{352}{h}(f_{i+1} + f_{i-1}) - \frac{702}{h}f_i \ . & \end{aligned} \tag{4.6}$$

Appropriate modifications are made at the extreme end points to conform with periodic boundary conditions. Since the simulations are 3-D, the linear system above must be solved in all three coordinate directions to obtain all the necessary derivatives for (4.5). As can be seen from (4.6) the CCD scheme is implicit in that it couples the grid points in the direction the derivative is taken, i.e., the first and second derivatives are obtained simultaneously for all grid points on the grid line where the derivative is taken. If entire lines of data are in the memory of each parallel process then the solution for all unknowns on the left of these equations would be relatively straightforward. However, such an approach would require communication-intensive transposes of the type needed for the velocity field based on a 2-D decomposition shown in figure 4.1.

Instead of a transpose-based algorithm based on a 2-D (or 1-D) domain decomposition, a 3-D decomposition is used which is static and thus removes the need for transposes (Gotoh *et al.*, 2012). In this arrangement it is clear that no one processor has the complete information required to solve the entire linear system in any direction. However it is possible for parallel processes along a given dimension of the 3-D decomposition to communicate to solve the linear system in tandem. It is helpful to consider (4.6) as a periodic block tridiagonal matrix system with 2×2 block elements. The solution to the block matrix system is distributed among the parallel processes using the method of Nihei & Ishii (2003), which generalizes the approach of Mattor

Table 4.1: Operations required for the parallel solution to the CCD linear system (see Appendix D). The eighth-order CCD scheme requires two ghost layers to be filled in Operation A. The linear system for Operation B has size proportional to N_θ/P, where P is number of parallel processes in the given coordinate direction. The size of the linear system for Operation D is proportional to P. Communication calls are posted in the MPI communicator for each coordinate direction.

Operation	Operation Summary
A	Fill ghost layers for scalar field with `SEND` and `RECV` operations
B	Form right-hand side of linear system and obtain solution
C	Pack and distribute data for reduced system with `MPI_ALLTOALL`
D	Unpack data and solve reduced linear system
E	Pack and distribute data for final solution with `MPI_ALLTOALL`
F	Unpack data and finalize solution of CCD linear system

et al. (1995) for tridiagonal matrices. There are six operations required to solve the CCD linear system in parallel, which are summarized in table 4.1 and discussed in Appendix D. Clearly, ghost layer information must be exchanged among the processes as required by the finite difference stencil, and this is accomplished in Operation A. The linear system is then broken into three parts. First, in Operation B each processor solves a "large" block tridiagonal system of size proportional to the number of grid points the processor owns. Second, the solutions across multiple processes are coupled by a "small" reduced system of size proportional to the number of processes participating in the solution. In Operation C, the parallel processes exchange information required for the reduced system, and after the reduced system is solved in Operation D, the necessary components of the solution required by each processor are redistributed in Operation E. Finally, in Operation F the solutions from Operations B and D are combined to complete the solution to the CCD scheme. As noted by Gotoh *et al.* (2014), the parallel algorithm requires communication cost proportional to N_θ^2, or the surface area, rather than the volume of the domain. This major reduction in communication requirements at large problem sizes makes the CCD scheme and its parallel solution an attractive alternative to transpose-based approaches.

4.2 Parallel implementation and performance of the CCD scheme

The most expensive part of the computation in the dual-communicator production code is application of the CCD scheme. It is therefore useful to isolate the CCD routines in a kernel to study and improve their performance independently of other parts of the code. The subsections below begin with a presentation of baseline performance data, and then move on to discuss improvements achieved by overlapping computation with non-blocking communication, and by the use of OpenMP multithreading where a dedicated master thread performs communication while other threads compute concurrently. Scalability data obtained on Blue Waters is presented for a wide range of problem sizes and process counts.

4.2.1 Baseline performance of the CCD routines

To understand the performance characteristics of the codes, wall clock timings (via the function `MPI_WTIME`) are collected and analyzed over multiple steps of code execution. Timings are collected on every MPI process over a period of at least 10 steps. For the CCD kernel a "step" is the application of the CCD scheme. Since all MPI processes must be synchronized at the end of each step the time reported by the slowest (rate-limiting) process is used as the measure of performance. Because of factors such as network contention from other jobs running concurrently on the system some degree of statistical variability is expected. The best timing among all steps is used since it provides a measure of the best achievable performance for a given version of the code with minimal external interference.

The purpose of the CCD routine is to compute the first and second derivatives of the scalar field. In the original version of the code (prior to enhancements addressed later) the derivatives are formed sequentially, one direction at a time for the x_1, x_2 and x_3 directions. For each direction the code performs the six operations identified

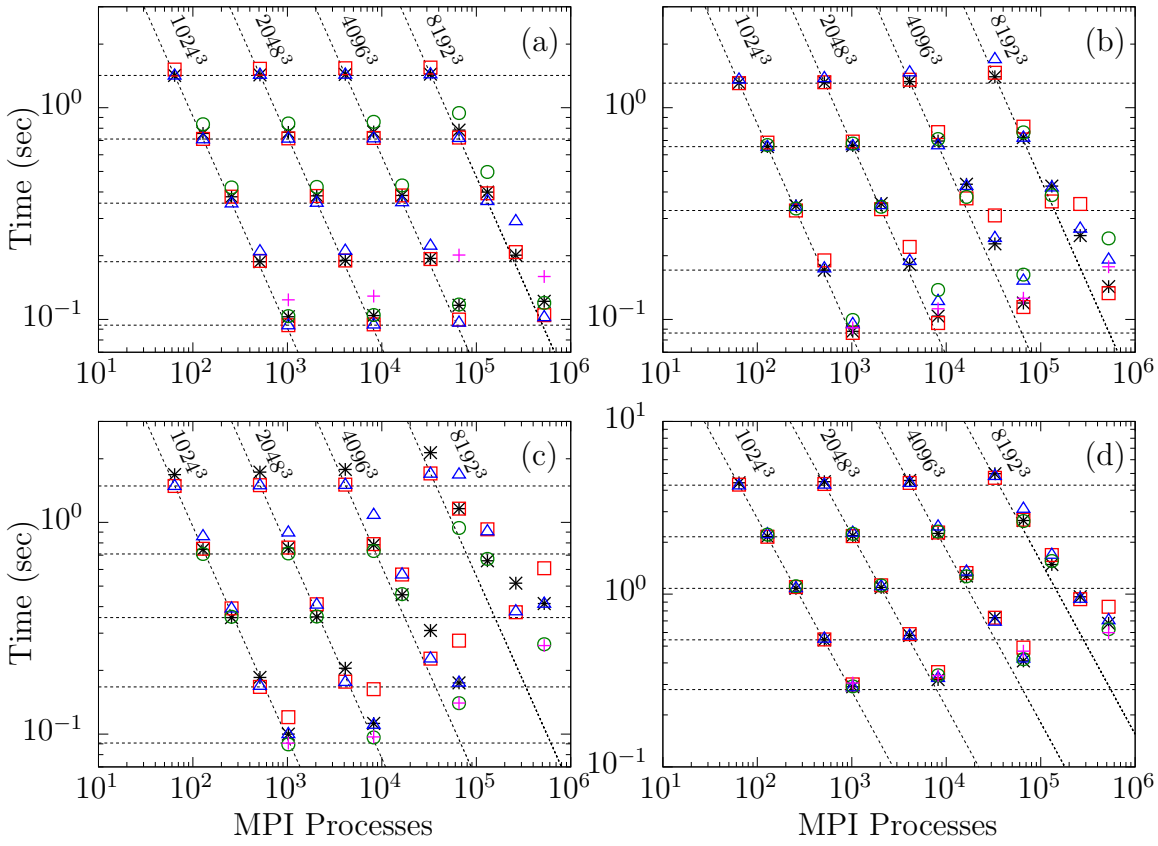

Figure 4.2: Timings for single threaded execution of CCD routines with no overlap between communication and computation for (a) derivative in x_1, (b) derivative in x_2, (c) derivative in x_3, and (d) overall timing of all three derivatives. See table 4.2 for description of symbols. The problem size, e.g., 1024^3, is given to the right of the sloped line for each strong scaling dataset.

in table 4.1, which require communication among neighboring MPI processes both for the ghost layer information and a parallel transpose-free solution algorithm for block tridiagonal equations. In this version the communication is blocking, and the total time used by the subroutine is the sum of computation and communication times.

Figure 4.2 shows elapsed wall clock times for the baseline CCD routines for derivatives in x_1, x_2 and x_3 directions (frames a-c), as well as the total time for all three directions (frame d). At each problem size tested (N_θ from 1024 to 8192) the core count M_θ is varied by factors up to 16. Perfect strong scaling is achieved when all data points for a given N_θ fall on a line of slope -1 on logarithmic scales, and perfect

Table 4.2: Number of MPI processes (M_θ) and process layouts used when testing single threaded performance of CCD routines for a 1024^3 problem size. For each larger problem size the number of processes is multiplied by 8, and each entry in the "Layout" column is multiplied by two. For example, for 4096^3 the smallest configuration is 4096 MPI processes with $16 \times 16 \times 16$, $32 \times 16 \times 8$, and $16 \times 32 \times 8$ process layouts.

M_θ	Layout	Mark
64	$4 \times 4 \times 4$	✳
	$8 \times 4 \times 2$	□
	$4 \times 8 \times 2$	△
128	$8 \times 4 \times 4$	✳
	$4 \times 8 \times 4$	□
	$4 \times 4 \times 8$	△
	$16 \times 4 \times 2$	○
256	$8 \times 8 \times 4$	✳
	$8 \times 4 \times 8$	□
	$4 \times 8 \times 8$	△
	$16 \times 4 \times 4$	○
512	$8 \times 8 \times 8$	✳
	$8 \times 16 \times 4$	□
	$16 \times 8 \times 4$	△
1024	$16 \times 8 \times 8$	✳
	$8 \times 8 \times 16$	□
	$8 \times 16 \times 8$	△
	$16 \times 16 \times 4$	○
	$32 \times 8 \times 4$	+

weak scaling is achieved when timings across different problem sizes lie on the same horizontal line. For each combination of N_θ and M_θ, different process layouts are represented by different symbols. This information is given explicitly for $N_\theta = 1024$ in table 4.2 and can be inferred for all other larger problem sizes according to the caption therein. For the targeted production problem at $N_\theta = 8192$, five core counts are tested, ranging from $32,768$ to $524,288$ with $M_\theta = 262,144$ being the value used for production. In this case, the process layouts for $M_\theta = 262,144$ are inferred from table 4.2 by scaling up each dimension from $N_\theta = 1024$ by a factor of $8192/1024 = 8$, giving black asterisk for $64 \times 64 \times 64$, red squares for $64 \times 128 \times 32$, and blue triangles for $128 \times 64 \times 32$.

In figure 4.2 it is clear that calculation of derivatives in the x_1 direction is the most robust, and with the appropriate process layout, scales better than the derivative calls in x_2 and x_3. The excellent strong scaling for x_1 is a result of superior communication performance in this direction. When the 3-D domain decomposition is initialized, MPI processes are placed on the compute nodes sequentially — first in x_1 direction, then x_2 and finally x_3. Communication in the x_1 direction therefore involves mostly intra-node communication, and a reduced number of nodes for inter-node communication compared to the other directions. For example, when using the configuration of 32

MPI processes per XE6 node with a $64 \times 64 \times 64$ process layout, communication in the x_1 direction involves 2 nodes, while communication in the x_2 and x_3 directions involves 64 nodes. Furthermore, inter-node communication for the x_1 direction is very robust, because the node ordering scheme used by the topologically-aware job scheduler on Blue Waters (Enos *et al.*, 2014) ensures that adjacent nodes are very close together in the torus network topology. One consistent trend for all three derivative routines is that performance typically improves when the number of MPI processes is reduced for that direction. Denoting the process layout $P_1 \times P_2 \times P_3$, when $N_\theta = 8192$ and $M_\theta = 262,144$ derivatives in x_1 give better timings if P_1 is the smallest dimension, while derivatives in x_3 perform better when P_3 is the smallest. This trend is understood by noting that when the number of processors in a given direction is reduced the communication requirements for that direction are also reduced. For example, if any one of P_1, P_2, or P_3 is halved the number of planes of data required to fill ghost layers in that direction is also halved, and the complexity of the `MPI_ALLTOALL` calls also reduces. However, if M_θ is fixed it is likely that while CCD performance in one direction is improved by reducing the processor grid dimension in that direction, performance in another direction may be worsened. This effect is strongest for the x_2 and x_3 directions.

Since derivatives in all three directions are required, frame (d) of figure 4.2 is used to assess the total CCD timings. Overall, sensitivity to the process layout is much less than that observed for individual directions, but close observation shows that process layouts which reduce communication complexity in the x_3 direction are still preferred, e.g., when $N_\theta = 4096$ and $M_\theta = 65,536$, $64 \times 32 \times 32$ and $32 \times 64 \times 32$ are both better than $32 \times 32 \times 64$, and when $N_\theta = 8192$ and $M_\theta = 524,288$, the process layouts $128 \times 128 \times 32$ and $256 \times 64 \times 32$ give better overall results than process layouts with 64 or more MPI processes in the x_3 direction. Although strong scaling is very good for 1024^3 and 2048^3, there is evidently a drop in scalability for 4096^3

and 8192^3 at the large core counts used in production simulations. This limitation in scaling performance of the baseline code provides the main motivation for pursuing performance improvements in this work.

4.2.2 Overlapping communication and computation in the CCD routines

A well-known paradigm to improve the scalability of parallel algorithms is to try to hide the cost of the communication calls by letting them occur simultaneously with computation. In the code, since CCD operations for different coordinate directions are independent of each other, there is a special opportunity to overlap communication for one direction with computation for another direction. In fact multiple opportunities exist since, as listed in table 4.1, for each coordinate direction the CCD scheme requires three communication operations (A,C,E) which are naturally interleaved with three other computational operations (B,D,F). To allow for overlapping, the code calls non-blocking versions of MPI routines, in particular `MPI_ISEND`, `MPI_IRECV`, and `MPI_IALLTOALL`, combined with matching `MPI_WAIT` calls to ensure that the required data are received and in place before they are used in computational operations. In principle, the potential for speedups through the use of overlapping is maximized when the two operations scheduled to occur simultaneously take equal amounts of time. With this algorithm, such a scenario is associated with cases where the fraction of time spent in communication is close to 50%, which is more likely at larger problem sizes. It may be noted that the opportunities for overlapping pursued here are not readily available for pseudo-spectral codes.

Table 4.3 shows a detailed schedule of 26 operations in the overlapping algorithm used to obtain first and second derivatives in all three directions. The number 26 is the sum of 6 operations each (per table 4.1) for each of the three directions and 8 calls to `MPI_WAIT` for synchronization following non-blocking communication calls. The algorithm starts with communication in the x_1 direction because communication in

Table 4.3: Rearrangement of CCD differentiation routine to overlap communication and computation in all three coordinate directions. The subroutine progresses sequentially from Operation 1 to Operation 26. The letters in the Operation column correspond to the entries in table 4.1, and the subscript indicates the coordinate direction for the operation, e.g., A_1 for filling the ghost layers in the x_1 direction.

Op.	x_1	x_2	x_3
1 A_1	SENDRECV		
2 A_2		ISEND/IRECV	
3 B_1	Compute		
4 C_1	IALLTOALL		
5 A_2		WAIT	
6 A_3			ISEND/IRECV
7 B_2		Compute	
8 C_2		IALLTOALL	
9 A_3			WAIT
10 B_3			Compute
11 C_3			IALLTOALL
12 C_1	WAIT		
13 D_1	Compute		
14 E_1	IALLTOALL		
15 C_2		WAIT	
16 D_2		Compute	
17 E_2		IALLTOALL	
18 C_3			WAIT
19 D_3			Compute
20 E_3			IALLTOALL
21 E_1	WAIT		
22 F_1	Compute		
23 E_2		WAIT	
24 F_2		Compute	
25 E_3			WAIT
26 F_3			Compute

this direction is the fastest. The code begins by filling ghost layers in the x_1 direction with a blocking `MPI_SENDRECV` call (Operation A_1). At this point the preparation of ghost layers in the x_2 direction (A_2) by using non-blocking `MPI_ISEND` and `MPI_IRECV` calls can proceed concurrently with a computation operation (B_1) for the x_1 direction. Once step B_1 completes, the code issues a non-blocking `MPI_IALLTOALL` (C_1) call on the results while it checks if the communication operation A_2 is completed. Once the results from A_2 are ready the code issues non-blocking `MPI_ISEND` and `MPI_IRECV` calls to fill ghost layers in the x_3 direction while it carries out the computational operation B_2. As the cycle progresses the code also checks if communication C_1 is complete, and then moves on with computation D_1 when ready. Similar cycles of operations are extended until the last operation (F_3) on the third direction is completed.

Table 4.4 compares CCD timings with and without the overlapping discussed above for problem sizes $N_\theta = 4096$ (top) and 8192 (bottom). The timings are taken from the process layout which gave the best overall performance. A measurable im-

Table 4.4: Overall CCD timings in seconds, and strong and weak scalings for (top) 4096^3 and (bottom) 8192^3 grids. Weak scaling is relative to timings at 1024^3. Each column is for a given number of MPI processes ($K \equiv 1024$). In each table, the top section is for the implementation with blocking communication, and the bottom section is for the implementation which overlaps communication and computation.

M_θ	$4K$	$8K$	$16K$	$32K$	$64K$
Blocking	4.403	2.249	1.268	0.695	0.411
Weak (%)	97.2	95.5	85.2	78.2	70.6
Strong (%)	—	97.9	86.8	79.2	67.0
Overlap	4.386	2.202	1.209	0.671	0.359
Weak (%)	97.6	97.4	89.0	81.1	77.8
Strong (%)	—	99.6	90.7	81.7	76.4

M_θ	$32K$	$64K$	$128K$	$256K$	$512K$
Blocking	4.717	2.651	1.484	0.936	0.599
Weak (%)	90.7	81.0	72.7	58.1	48.4
Strong (%)	—	89.0	79.4	63.0	49.2
Overlap	4.518	2.477	1.359	0.768	0.527
Weak (%)	94.7	86.5	79.2	70.8	53.0
Strong (%)	—	91.2	83.1	73.5	53.6

provement due to overlap is evident in all cases, and generally becomes more significant with increases in either N_θ or M_θ. For the largest case of interest for production runs, i.e., 8192^3 on $262,144$ cores, detailed profiling shows that the average fraction (over all MPI processes and steps) of time spent in communication is 48%. This suggests that the maximum possible reduction in timing is from 0.936 to 0.49 seconds. However perfect overlapping is not possible since the operations at a fine-grained level that are being overlapped (per table 4.3) can take different times. The actual overlap achieved corresponds to 38% of this theoretical maximum, which is reasonable and still represents a substantial improvement in performance overall.

4.2.3 Multithreading approach with dedicated communication threads

Most massively parallel processing systems available to the HPC community consist of multi-cored processors. Since communication overhead due to a large number of MPI processes is a common challenge to scalability, it is useful to explore if the introduction

of shared-memory programming can produce improvements at large core counts. The combination of distributed-memory and shared-memory programming is referred to as hybrid programming, where the shared-memory parts of the code can be implemented with OpenMP or MPI-3, among others. Such hybrid programming has improved the performance of many large user codes in the field of fluid dynamics (Mininni, 2011; Jagannathan & Donzis, 2012). When using OpenMP, a common practice is to operate in the so-called `MPI_THREAD_FUNNELED` mode in which all threads attached to an MPI process compute with shared memory, but only the master thread performs communication. If MPI communication calls (blocking or non-blocking) reside in the parallel region, they can be encapsulated between `!$OMP MASTER` and `!$OMP END MASTER`, with a subsequent `!$OMP BARRIER` if synchronization is needed. This is in fact the approach used to thread the routines described in §4.2.1 and §4.2.2.

An alternative hybrid approach is to overlap communication and computation explicitly by assigning certain threads to perform communication calls, while all other threads compute concurrently (Rabenseifner & Wellein, 2003; Hager *et al.*, 2011). With OpenMP, such an approach can be implemented by spawning a large parallel region that encompasses a subroutine of interest, and then dividing the region into communication and computation sections. Each thread can be classified as either a communication or computation thread and execute different lines of code accordingly. However, when threads are split in this manner, the computation threads cannot use standard OpenMP work-sharing directives, e.g., `!$OMP DO`, because some threads (the communication threads) do not encounter the directives (Rabenseifner *et al.*, 2009). While it is possible to divide the work manually based on thread identifier, doing so would be tedious and error prone because certain synchronization points such as the implied barrier at `!$OMP END DO` would no longer be present. Fortunately, these issues can be resolved by using nested OpenMP parallelism, which allows a new master thread to be defined within the group of computational threads and standard

work-sharing directives to be used within that group.

A multithreading strategy has been implemented with OpenMP to explicitly overlap communication and computation for derivatives in all three coordinate directions. The majority of the CCD routine is contained in one large OpenMP parallel region, which begins initially with 2 threads — one for communication and one for computation. The computation thread then spawns a nested OpenMP parallel region to use all of the remaining threads available. It is important to have a mechanism in place to enforce the proper order of execution for each part of the CCD scheme (detailed in table 4.1) and to ensure proper shared memory synchronization between the communication and computation threads. This coordination between the communication and computation threads is accomplished with OpenMP locks, which as of version 2.5 of OpenMP imply the necessary memory flushes (Hoeflinger & de Supinski, 2005) that make this approach viable. One OpenMP lock is created for each coordinate direction. The lock for each direction is set before any work (communication or computation) is performed for that direction, and unset when the work is completed so that other threads can subsequently set the lock and access the results.

The details of the algorithm are illustrated in figure 4.3. Similar to implementations without multithreading (§4.2.2), the CCD calculations begin with communication in the x_1 direction to fill the ghost layers in x_1. Since no computations can proceed until the x_1 direction ghost layers are filled, this communication call is left outside of the main OpenMP parallel region. Once the main parallel region is spawned, the nested parallel region is spawned for the computational thread team, and the threads move to set the OpenMP locks based on their initial work assignments: the communication thread will fill ghost layers in the x_2 and x_3 directions, while the computation threads will form and solve the initial linear system for the x_1 direction (Operation B_1 in table 4.3). To enforce the correct sequence of communication and computation operations for all coordinate directions, it must be ensured that

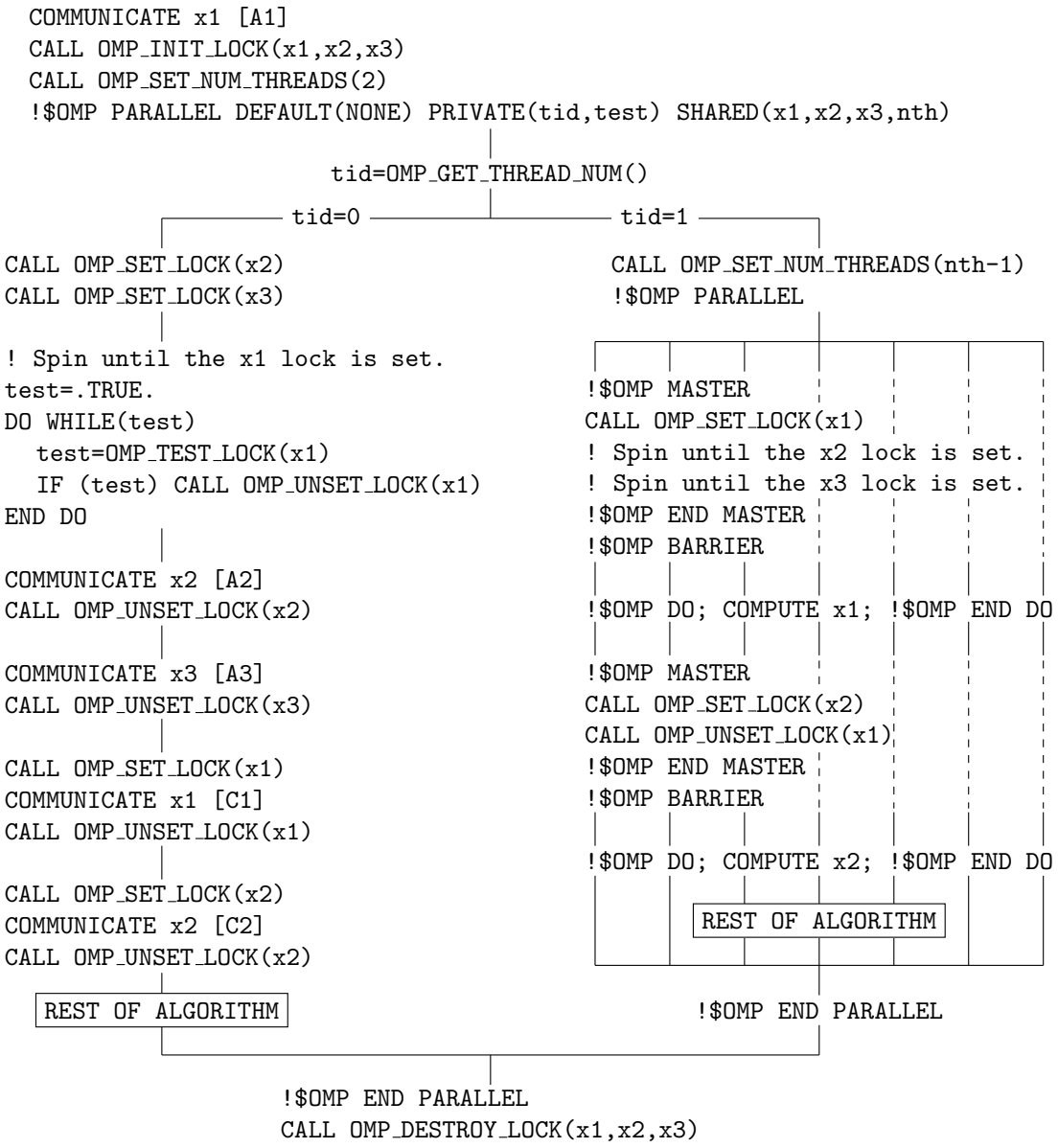

Figure 4.3: Outline of hybrid MPI-OpenMP implementation of CCD code in which one thread performs communication and all other threads perform computation. The first parallel region is always spawned with two threads, where `tid` stands for thread identifier. The total number of threads available (including the communication thread) is `nth`. The dashed lines during the `!$OMP MASTER` region illustrate other computation threads being idle while locks are exchanged. The general pattern repeats until all steps in table 4.1 are complete for all coordinate directions. The descriptions for each communication call, e.g., `[C1]`, follow those given in the caption of table 4.3.

neither thread group advances to its first task until it is known that all locks have been properly set. To accomplish this, each thread group checks that the other thread group has set its lock(s) with the help of the `OMP_TEST_LOCK` function. The actual code used is illustrated in the left of figure 4.3 for the communication thread, and the master thread of the computation thread team performs the same operations to check that the locks for the x_2 and x_3 coordinate directions were properly set by the communication thread. All computations are performed in the large (nested) OpenMP parallel region, which has sufficient flexibility to divide the workload among the computation threads with standard work-sharing directives. The code then progresses through the complete cycle of operations outlined in table 4.3, with the exception that blocking communication is used, hence no `MPI_WAIT` calls are necessary. After each computation operation is complete the master computation thread exchanges locks with the communication thread. A subsequent `!$OMP BARRIER` in the nested OpenMP region synchronizes the computational thread team before they begin the next set of computations.

As with any OpenMP algorithm, there is a risk of losing computational efficiency if there is a load imbalance among the threads. In the algorithm described in this section, there are two sources of load imbalance. The first is quite obvious: since N_θ, M_θ as well as the total thread count per MPI process are all even numbers (and powers of 2), when one thread is removed from the computation section of the code the workload can no longer be divided equally among the threads. When using this routine on BW XE6 nodes one MPI process with 8 OpenMP threads is assigned to each NUMA domain, thus leaving only 7 out of 8 threads per MPI process to execute the nested parallel region. When the number of iterations in a loop is, say, 512, a loop using `!$OMP DO SCHEDULE(STATIC)` results in the first 6 threads being responsible for 74 units of work, while the last thread is responsible for 68, which amounts to a modest imbalance of the order 10%.

A second source of load imbalance encountered is specific to the Blue Waters architecture. The BW XE6 nodes are comprised of two AMD 6276 Interlagos processors (Bode *et al.*, 2013), and each processor contains two NUMA domains comprised of four floating point cores. There are two integer cores on the processor per floating point core, thus giving the eight integer cores per NUMA domain to which the MPI process (master thread) and computational threads are assigned. When the master thread is split off to perform only communication, the other integer core sharing a floating point core with this thread is automatically provided with exclusive access to the computational resources equivalent to two integer cores. This means one of the threads in the OpenMP parallel regions will be consistently — tests show as much as 40% — faster than the others. As a result, the default `STATIC` scheduling of OpenMP loops, which assigns approximately equal workloads to each thread, does not provide the best performance attainable.

To alleviate both the software and hardware driven load imbalances above it is profitable to use `GUIDED` instead of `STATIC` scheduling. This allows units of work to be assigned in chunks to the threads as the threads request them, where under `GUIDED` scheduling the chunk size starts high and is gradually reduced to a minimum value specified by the user. In this manner threads which can compute faster will be allocated more units of work as the loop iterations are computed. A slight improvement in performance can be expected, subject to the caveat that more thread synchronization is required when using `GUIDED` scheduling, because many loops can no longer be appended with the `NOWAIT` clause. Various OpenMP loop schedules were tested to see if the load imbalances could be reduced while resulting in faster overall execution of the CCD routines. Eventually `!$OMP DO SCHEDULE(GUIDED,1)`, i.e., the minimum chunk size being one, was used, because it provided a few percent speedup in the CCD routine and is a flexible loop schedule.

4.2.4 Improvements in performance obtained from different strategies

This section is concluded by presenting and comparing timings obtained from a total of five different implementations of the CCD routines for computing first and second derivatives in all three coordinate directions. For convenience, the following two-letter acronyms are used for the various implementations:

BS	*Blocking* communication, *single threaded* (baseline version)
OS	*Overlap* communication and computation, *single threaded*
BM	*Blocking* communication, *multithreaded*
OM	*Overlap* communication and computation, *multithreaded*
ON	*Overlap* communication and computation with *nested multithreading*

Although implementations BS and OS have been compared already in table 4.4, for a complete discussion data from all implementations is presented in both graphical (figure 4.4) and tabular (table 4.5) forms. For BM and OM tests were conducted with 2, 4 and 8 threads per MPI process, and the data shows the best timings among the three choices of thread count. Weak scaling is inferred relative to the case of $N_\theta = 1024$ along the horizontal dashed lines in figure 4.4, while noting that the computational operations count is proportional to N_θ^3. Strong scaling for each N_θ is inferred based on the smallest number of PEs tested. It is clear from figure 4.4 that at smaller problem sizes (up to 2048^3) all implementations show near-perfect scalability, and single threaded, i.e., pure MPI, approaches suffice. However, at large problem sizes and PEs (to which table 4.5 is restricted) the picture is quite different.

As expected, the baseline implementation (BS) where operations are performed sequentially with no attempt at overlap is the least efficient. However, even without any overlap of communication and computation, a hybrid MPI-OpenMP approach (implementation BM, the open non-square blue symbols) with a smaller number of MPI processes scales better than pure MPI. This improvement can be explained by

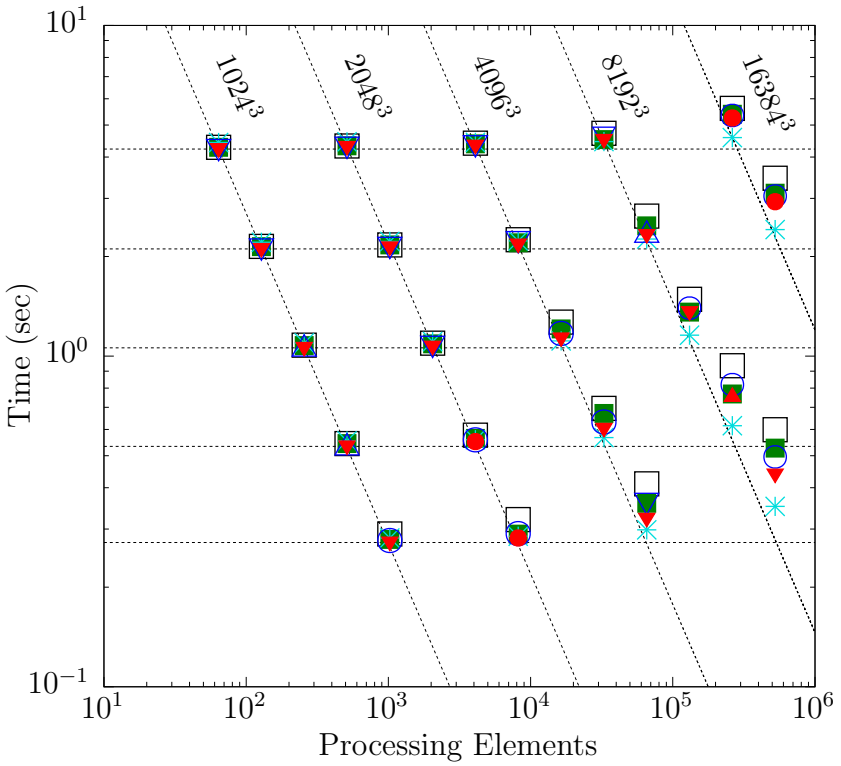

Figure 4.4: Timings for different implementations of the CCD routine that calculate derivatives in all three coordinate directions, versus the number of processing elements (defined as the product of the number of MPI processes and the number of OpenMP threads per MPI process). Some symbols are partially hidden as they overlap with one another. Open symbols denote implementations where derivatives are taken independently with no overlap of communication and computation: black squares (□) for single threaded version, and blue symbols for best performing (with respect to thread count) multithreaded version with △, ▽, and ○ for 2, 4, and 8 OpenMP threads, respectively. Filled symbols denote implementations which overlap communication and computation: green squares (■) for single threaded version, and red symbols for best performing (with respect to thread count) multithreaded version with ▲, ▼, and ● for 2, 4, and 8 OpenMP threads, respectively. Cyan star (✳) for overlapping and nested-parallelism implementation in which a dedicated thread per NUMA domain performs communication and no computation.

Table 4.5: Overall CCD timings in seconds, with strong and weak scaling results for (top) 4096^3 and (bottom) 8192^3 grids. Weak scaling is relative to timings at 1024^3. Each column is for a given number of processing elements (PEs, with $K \equiv 1024$), defined as the number of MPI processes multiplied by the number of OpenMP threads per MPI process. Each horizontal block is for a different implementation with the two-letter codes defined at the beginning of this section. The symbols given in parenthesis match those in figure 4.4 for each implementation. Subroutine ON uses one MPI process per NUMA domain (4 per BW XE6 node) with 8 OpenMP threads per MPI process. For BM and OM the timing from the best MPI-OpenMP configuration for a given problem size is reported (see symbol shapes in figure 4.4 for the number of threads).

PEs	$4K$	$8K$	$16K$	$32K$	$64K$
BS (□)	4.403	2.249	1.268	0.695	0.411
Weak (%)	97.2	95.5	85.2	78.2	70.6
Strong (%)	—	97.9	86.8	79.2	67.0
OS (■)	4.386	2.202	1.209	0.671	0.359
Weak (%)	97.6	97.4	89.0	81.1	77.8
Strong (%)	—	99.6	90.7	81.7	76.4
BM (△,▽,○)	4.349	2.231	1.171	0.632	0.361
Weak (%)	97.4	95.0	91.1	84.9	76.8
Strong (%)	—	97.5	92.9	86.0	75.4
OM (▲,▼,●)	4.331	2.176	1.132	0.603	0.322
Weak (%)	97.6	97.0	93.6	88.5	84.7
Strong (%)	—	99.5	95.7	89.8	84.0
ON (✳)	4.429	2.230	1.112	0.567	0.298
Weak (%)	99.9	99.1	98.6	97.7	94.5
Strong (%)	—	99.3	99.6	97.6	92.9

PEs	$32K$	$64K$	$128K$	$256K$	$512K$
BS (□)	4.717	2.651	1.484	0.936	0.599
Weak (%)	90.7	81.0	72.7	58.1	48.4
Strong (%)	—	89.0	79.4	63.0	49.2
OS (■)	4.518	2.477	1.359	0.768	0.527
Weak (%)	94.7	86.5	79.2	70.8	53.0
Strong (%)	—	91.2	83.1	73.5	53.6
BM (△,▽,○)	4.641	2.358	1.397	0.820	0.496
Weak (%)	91.3	89.9	76.4	65.4	55.9
Strong (%)	—	98.4	83.1	70.7	58.5
OM (▲,▼,●)	4.510	2.322	1.358	0.757	0.438
Weak (%)	93.7	90.9	78.0	70.5	62.3
Strong (%)	—	97.1	83.0	74.4	64.3
ON (✳)	4.454	2.259	1.157	0.616	0.351
Weak (%)	99.3	97.8	94.7	89.9	80.3
Strong (%)	—	98.6	96.2	90.3	79.3

noting that when the number of MPI processes is reduced the overall volume of data required to fill ghost layers as well as the complexity of the `MPI_ALLTOALL` calls are reduced. Implementations OS and OM (closed green squares and closed non-square red symbols, respectively) which are designed to overlap communication and computation also produce measurable if not dramatic performance gains. For example, for the case $N_\theta = 8192$ and $262,144$ PEs the overall timing drops from 0.820 to 0.757 seconds when using OM in favor of BM. It is clear that the more elaborate approach of employing a dedicated communication thread with nested OpenMP parallelism for computations described in §4.2.3 (designated ON, with cyan star symbols) gives the best timing, beating the BS time of 0.936 seconds for 8192^3 on $262,144$ PEs with a time of 0.616 seconds — a 34% improvement. This approach maintains strong scaling of effectively 80% over an sixteen-fold increase of core count and 80% weak scaling over an eight-fold increase in problem size ($N_\theta = 1024$ versus $N_\theta = 8192$).

While the primary production problem sizes of interest are 4096^3 and 8192^3, some timings for a 16384^3 problem are also included in figure 4.4 using as many as $524,288$ PEs. Since tests at this scale are expensive, they are limited to the cases of 1 and 8 OpenMP threads, and have used a fewer number of process layout combinations than tests at smaller problem sizes, e.g., for 1 OpenMP thread and $262,144$ PEs only $128 \times 64 \times 32$ and $64 \times 64 \times 64$ process layouts were tested. It is encouraging to note that implementation ON performs well for this problem size with 97% and 92% weak-scaling relative to 1024^3 timings at $262,144$ and $524,288$ PEs, respectively. In addition, the strong scaling when increasing from $262,144$ to $524,288$ PEs is 95%.

In addition to timings and scalability data, measurements are taken to assess the floating point and memory performance of the code. Since Blue Waters is a Cray machine these types of data are obtained by instrumenting the CCD routines with low-level Fortran routines provided by the CrayPat application programming interface (API) (CRAY, 2015). While implementation ON performs best, because of

Table 4.6: Floating point and memory usage statistics for OS version of the CCD routine for problems $N_\theta = 1024$, $N_\theta = 2048$, and $N_\theta = 4096$ run on 2, 16, and 128 BW XE6 nodes, respectively. Each BW XE6 node has a maximum FLOP rate of 313 GFLOP/s, which is used to calculate the percentage of peak floating point utilization. The read memory bandwidth ("Read BW") is normalized by the read portion of the memory bandwidth attained by the STREAM triad benchmark (38.8 GB/s). "L1 Hit" gives the fraction of memory accesses that were found in the L1 cache.

N_θ	% Peak FLOP/s	FLOP/N_θ^3	% Read BW	L1 Hit (%)
1024	5.90	153.6	77.8	97.9
2048	5.86	153.7	77.2	98.0
4096	5.81	153.7	76.4	98.0

some issues with CrayPat reporting when using OpenMP, these floating point and memory metrics are reported for the (single threaded) OS implementation only. Using the CrayPat API, the entire CCD routine is enclosed in a "region" to be monitored. Although it is possible to measure loop-level performance, the overall performance of the CCD routine including communication and computation is of primary interest.

A summary of floating point and memory usage statistics is given in table 4.6 for problem sizes $N_\theta = 1024$, 2048, and 4096 using 2, 16, and 128 BW XE6 nodes, respectively. These problem sizes form a weak scaling study and are run on MPI process layouts $4 \times 4 \times 4$, $8 \times 8 \times 8$, and $16 \times 16 \times 16$, respectively. The floating point performance of the subroutine is evaluated by comparing the floating point operation (FLOP) rate to the theoretical maximum for the number of BW XE6 nodes used (each node having a maximum of 313 GFLOP/s). The CCD routine achieves close to 6% of the peak FLOP rate, which compares well with other large production codes run on BW (Bauer *et al.*, 2016). As the problem size increases this percentage is expected to drop due to an increase in communication overhead. This effect appears to be only slight but will be stronger for problem sizes larger than those listed in the table. A normalized FLOP count (FLOP/N_θ^3) for the CCD routine is also provided. Because the linear system corresponding to the CCD scheme is factorized once at the

beginning of the simulation, the normalized FLOP count is approximately constant.

The memory performance of the CCD subroutine is assessed by measuring the read bandwidth from main memory using L3 cache misses (Bauer *et al.*, 2016). Instead of comparing with the maximum theoretical memory bandwidth for each node (102.4 GB/s (Bode *et al.*, 2013)), comparisons are made with the the bandwidth attained by the STREAM TRIAD benchmark (McCalpin, 1995), which is similar to the LAPACK subroutine DAXPY. To take the reference measurement, the MPI Fortran version of STREAM is compiled with O3 optimization and run on one XE6 node using 32 MPI processes. Each MPI process uses 768 MB of memory, i.e., the entire program uses 24 GB of the 64 GB available memory on the node, and the test is run for 40 steps. The STREAM TRIAD benchmark reports a memory bandwidth of 58.2 GB/s. Since the TRIAD benchmark counts two loads and one store per iteration, the read bandwidth is obtained by multiplying the raw bandwidth figure by 2/3, giving a read bandwidth of 38.8 GB/s. To check the validity of this modification, an independent test was run with a CrayPat-instrumented STREAM executable, which showed that the read bandwidth measured from L3 cache misses is within 2% of the modified STREAM TRIAD bandwidth. As shown in table 4.6, the CCD routine attains approximately 75%–80% of the STREAM benchmark, suggesting that the code is memory-bandwidth bound. Finally, a L1 hit ratio of 98% also shows the cache is being used efficiently.

Compact finite difference schemes are an important class of numerical methods for solving multi-dimensional partial differential equations arising in many different fields of science. As parallel computers evolve from multi-core to many-core, the principles behind the "best" approach of overlapping via dedicated thread(s) for communication and nested OpenMP parallelism for computation should be of general interest. The next section focuses on how the new parallel CCD algorithm is coupled with the velocity field through a dual communicator approach.

4.3 Parallel implementation and performance of the high Sc code

In this section the FPS and CCD algorithms are combined to create a new code for DNS of turbulent mixing at high Sc. First a description of the dual-communicator implementation is provided, followed by a presentation of scaling data.

4.3.1 A dual-communicator algorithm for simulations at high Sc

The new DNS code is built upon two existing code bases: an FPS code for computing the velocity field governed by the Navier-Stokes equations, and a CCD code for computing the scalar field governed by an advection-diffusion equation. The MPI processes constituting the global communicator `MPI_COMM_WORLD` are divided into two groups forming two disjoint intra-communicators, called `FPS2D_COMM` and `CCD3D_COMM` for FPS and CCD, respectively. A vital aspect of the new code is the nature of information transfer between MPI processes belonging to the different communicators. While some of the data transfer can be handled through `MPI_COMM_WORLD`, an explicit inter-communicator named `PSCCD_COMM` (created with `MPI_INTERCOMM_CREATE`) is used when sending information like the time step size and signals that coordinate writing checkpoints or statistical output.

Figure 4.5 shows a schematic of the inter-communicator transfer required to couple the Navier-Stokes and scalar field codes. The task at hand is to send the entire velocity field from the velocity communicator to the scalar communicator, where it it used to form the advective terms necessary to advance the scalar field in time. The velocity field is on a coarser grid with a 2-D decomposition, where each MPI process holds a "pencil" formed at the intersection between two row and column sub-communicators of size M_1 and M_2 along the x_2 and x_3 directions, respectively. The scalar field is on a finer grid with a 3-D decomposition based on three sub-communicators `i_world`, `j_world` and `k_world` formed by a $P_1 \times P_2 \times P_3$ process

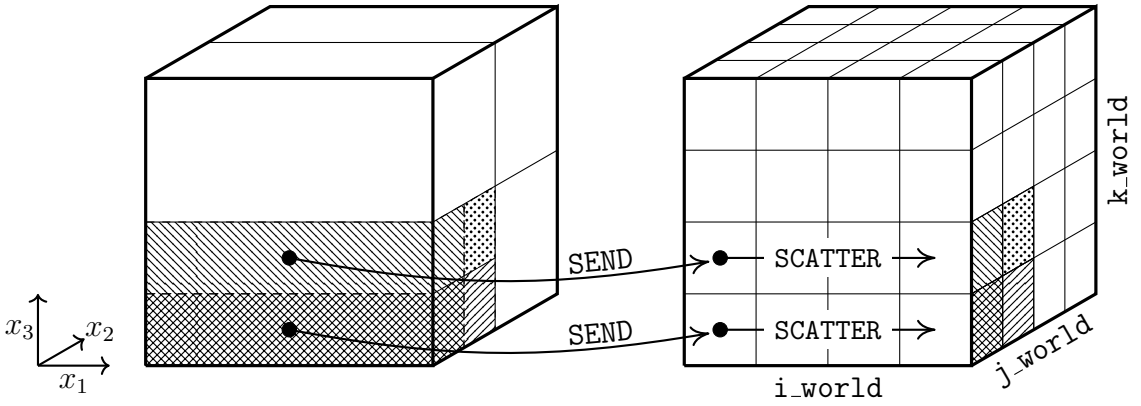

Figure 4.5: Illustration of the (left) physical-space orientation of the 2-D domain decomposition for the velocity field and the (right) orientation of the 3-D domain decomposition for the scalar field. In this example 4 processors are used for the velocity field and 64 processors are used for the scalar field. Patterns are shown on the two domains to illustrate how different portions of velocity field "pencils" are sent to the scalar field processors. Communication occurs by sending entire portions of the "pencils" from the velocity field communicator to the respective root process in the `i_world` scalar field communicator, which then scatters the data to the other processors.

layout. The FPS pencils are aligned in the same coordinate direction (x_1) as the `i_world` sub-communicator. The configuration seen in figure 4.5 corresponds to the case of $M_1 = M_2 = 2$ and $P_1 = P_2 = P_3 = 4$. The inter-communicator transfer required is one way, and is implemented using discrete sends and receives. In the example shown each of the $M_1 \times M_2 = 4$ pencils in the `FPS2D_COMM` communicator is split into $P_1 \times P_2/M_1 \times P_3/M_2 = 16$ small cubes in the `CCD3D_COMM` communicator, which can be accomplished by sending a total of 16 small messages (1 to each of 16 recipients). On the other hand, if $P_2 < M_1$ and $P_3 < M_2$, each sending `FPS2D_COMM` process will send only 1 message, but each `CCD3D_COMM` process will have to receive from multiple senders. The prospect of many small messages with either multiple recipients per sender or multiple senders per recipient is clearly not optimal, especially at large core counts.

To address the challenges above an alternative scheme is used in which each `FPS2D_COMM` process sends a sub-pencil to only the root process in the target `i_world`

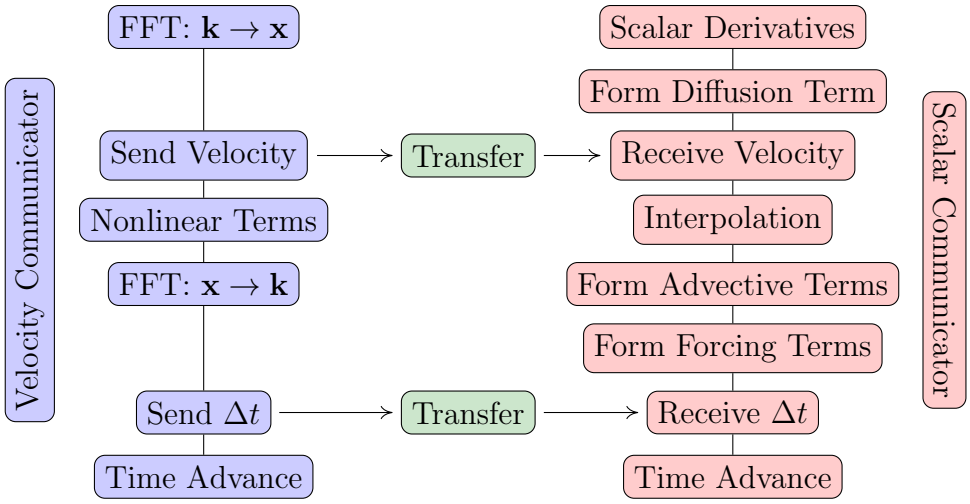

Figure 4.6: Time stepping procedure in the dual-communicator code for the first Runge-Kutta sub-stage. Velocity field solution on the left (blue) using FPS scheme, and scalar field solution on the right (red) using CCD scheme. The velocity field and time step are sent from the FPS communicator to the CCD communicator using inter-communicator transfers during each step of time integration.

communicator. The data is subsequently distributed to the other processes in the `i_world` communicator via `MPI_SCATTER`. This approach reduces the number of messages sent from `FPS2D_COMM` to `CCD3D_COMM` by a factor of P_1, and is efficient because communication in the x_1 direction is very robust, as discussed in §4.2.1. The number of messages can be minimized by matching M_1 with P_2, and M_2 with P_3. Some of the largest simulations use M_1 and M_2, as well as P_1, P_2 and P_3 all equal to 64. Finally, the messages in both the inter-communicator send and receive pairs and the scatter within the `i_world` communicators are not contiguous in memory. This is addressed by using MPI derived datatypes defined through `MPI_TYPE_CREATE_SUBARRAY` instead of explicit packing and unpacking.

In the dual-communicator algorithm, it is desirable for operations on the two communicators to be executed concurrently as much as possible. Figure 4.6 shows the essential operations for the first of four sub-stages in the classical RK4 time integration scheme. Subsequent sub-stages follow the same procedures except that the time step size Δt will already be known. At the beginning of each RK4 sub-

stage the velocity is in Fourier space and the scalar is in physical space. The FPS processes transform the velocity to physical space, while CCD processes compute the first and second derivatives of the scalar field needed for advection and diffusion terms, respectively. The velocity field is then transferred from the `FPS2D_COMM` communicator to the `CCD3D_COMM` communicator. Subsequently the FPS processes continue with forming nonlinear terms and transforming those to Fourier space. At the same time the CCD processes fill ghost layers for the velocity field under periodic boundary conditions and then interpolate the velocity to the finer grid to calculate the advective and mean-gradient terms. Tricubic interpolation, which provides sufficient accuracy, is used (Gotoh *et al.*, 2012). The interpolation does take significant resources but requires no communication and thus scales perfectly. The time step Δt is computed by the FPS processes considering the Courant number criterion for numerical stability based on the spacing of the finer scalar grid. When the CCD processes are ready to advance in time the value of Δt is made available to them by inter-communicator transfer. The time advance itself is implemented separately using the same RK4 scheme as for the velocity in Fourier space, but in physical space for the scalar field.

Several other opportunities for optimization merit further discussion. Since the FPS and CCD processes execute different instructions, even when the core counts deployed for each problem are proportional to the number of grid points there is no expectation that they will complete their respective time steps in the same wall clock time. On the other hand, since the coupling between the `FPS2D_COMM` and `CCD3D_COMM` communicators is one-way, the velocity field communicator is not required to be on the same RK4 sub-stage as the scalar field communicator. The code allows the velocity field integration (which will take less time by design) to get a few sub-stages ahead of the scalar field, which means that the velocity communicator must be able to send multiple velocity field realizations to the scalar communicator. Although the velocity field processors may be left idle for a faction of the time that is not a concern since the

overall cost of the simulation is still dominated by the larger scalar field problem. This asynchronous approach is implemented by allocating a number of buffer arrays for the velocity field and time step on the scalar processors, and then setting up distinct non-blocking `MPI_IRECV` calls for each sub-stage. The number of slots in the buffer arrays, i.e., the number of sub-stages by which the velocity field can get ahead in integration, is controlled by a user input to the code. In practice, to avoid creating a burden in memory usage, the buffer size is limited to 3 or 4. If the FPS processors get ahead by too many RK4 sub-stages they will wait for empty slots in the buffer arrays owned by the CCD processors to become available. The advantage of performing the time integration in an asynchronous fashion is that it allows the code to hide part of the communication cost of the data transfer between the communicators.

Depending on the parameters but especially at higher grid resolutions, turbulence simulations are often carried out over many thousands of time steps. Usually at approximately regular time intervals the code computes and outputs a number of statistical quantities, such as the turbulence kinetic energy and its spectrum. In the FPS code the energy spectrum is readily formed in wavenumber space by summing over the energies of Fourier modes which are already naturally available as part of the numerical solution. In the present code for scalar fields, since the numerical solution is based on CCD methods entirely in physical space, a 3-D Fourier transform must be taken specifically for the purpose of computing the spectrum (whose functional form at high Sc is of great interest (Batchelor, 1959; Donzis & Yeung, 2010)). While the scalar field is on a 3-D domain decomposition, the FPS code base already provides highly optimized routines for 3-D FFTs on a 2-D decomposition. To use those FFT routines the code performs a transpose of the scalar field from a 3-D to 2-D domain decomposition with a series of of `MPI_IRECV` and `MPI_SEND` calls within in the `CCD3D_COMM` communicator. This allows existing FFT routines, which after initialization only function for one grid resolution to be used at different resolutions by MPI

processes on the two distinct communicators (`FPS2D_COMM` and `CCD3D_COMM`). While output steps have a higher wall clock time, they are infrequent and are thus not a heavy addition to the overall simulation.

4.3.2 Parallel performance of dual-communicator DNS code

This subsection reports on the performance of the new dual-communicator DNS code. As done when analyzing the CCD routines, timings are collected with calls to `MPI_WTIME`. For each step the maximum time is taken among all MPI processes, and then the performance is assessed by taking the minimum over a number of time steps. Only regular time steps are considered, which do not include the additional costs of forming statistics or writing checkpoints. In addition to measuring the overall wall clock per time step, sub-timings for different parts of the code are also collected, including the CCD routines, performing interpolation, forming advection terms, and the time-advance component of each RK4 sub-stage. An emphasis is placed on performance measurements for the grid configuration $N_\theta/N_v = 8$, which is used for simulations with Schmidt number 64 or higher.

Because the problem size for the scalar field is larger than the velocity field by a factor proportional to $(N_\theta/N_v)^3$, the cost of the simulation is expected to be dominated by the scalar field computation. The dual-communicator nature of the algorithm also implies that the code will be most efficient if the velocity and scalar field core counts are chosen in proportion to the workload borne by each communicator. Otherwise, if M_v is too small the velocity field computation may potentially slow down the code, leaving the larger group of processes in `CCD3D_COMM` waiting for transfer of the velocity field. On the other hand, an overly large M_v would be wasteful.

Figure 4.7 shows the effects of the velocity-field core count M_v on the elapsed wall time per step of the dual-communicator DNS code for the problem size of $N_\theta = 4096$ and $N_v = 512$. The FPS routines are run in pure MPI mode with 32 MPI processes per

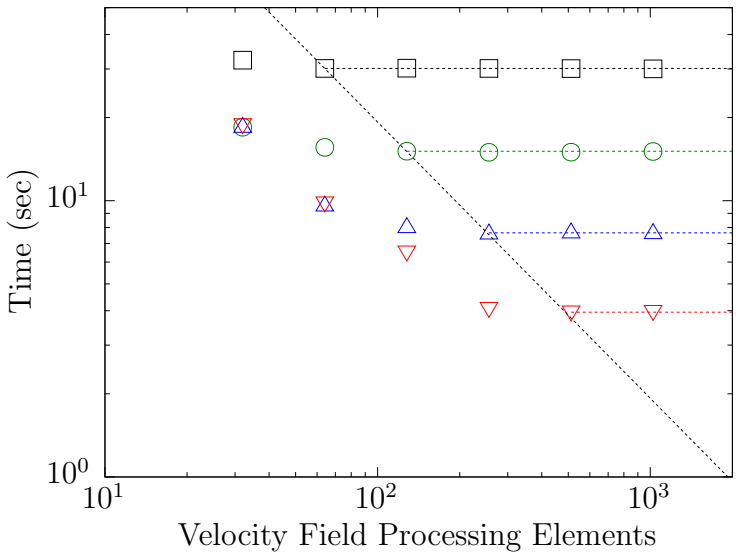

Figure 4.7: Wall clock time per step in seconds for dual-communicator code with $N_v = 512$ and $N_\theta = 4096$ as a function of the number of PEs used for the velocity field. The velocity field is computed using 32 MPI processes per BW XE6 node. Different symbols represent different node counts used for the scalar field: black squares (□) for 256 nodes, green circles (○) for 512 nodes, upward facing blue triangles (△) for 1024 nodes, and downward facing red triangles (▽) for 2048 nodes. The scalar field is computed using the ON version of the CCD routine described in §4.2.4.

XE6 node on Blue Waters, while the CCD routines for the scalar field were run using the implementation ON (see §4.2.3). Different symbols represent data where the scalar field communicator consists of 256, 512, 1024 and 2048 nodes, respectively. For each configuration the task is to determine the minimum number of MPI processes that must be used for the Navier-Stokes communicator so that the velocity field calculation does not slow down the entire code. This number is identified as the number of MPI processes beyond which the wall clock time per step no longer decreases. For example, when using 1024 nodes for the scalar field (corresponding to the upward facing blue triangles) 256 MPI processes (8 nodes) suffice for the velocity communicator.

Overall timings and scalability of the dual-communicator DNS code are shown in figure 4.8, where the scalar field problem size is varied from $N_\theta = 1024$ to $N_\theta = 8192$ and in each case the number of PEs is varied by a factor of eight. The velocity field

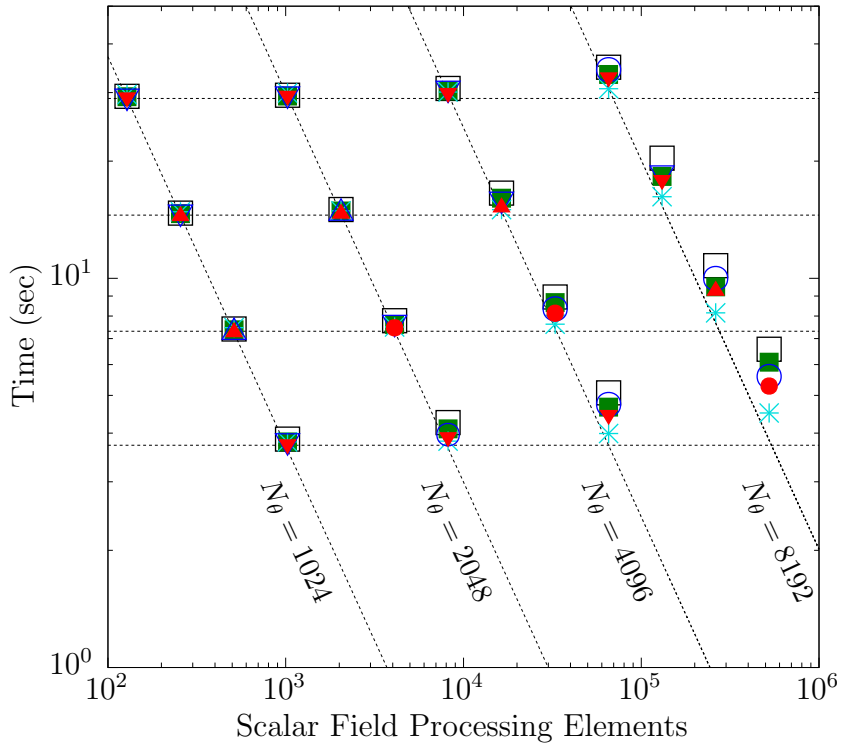

Figure 4.8: Overall simulation time per step as a function of the number of processing elements for the scalar field code. The symbols indicate which version of the CCD routine was employed during the run (see §4.2.4). Open symbols for routine in which derivatives are taken independently with no overlap of communication and computation: black squares (□) for single threaded version, and blue symbols for best performing (with respect to thread count) multithreaded version with △, ▽, and ○ for 2, 4, and 8 OpenMP threads, respectively. Filled symbols for routine which calculates all derivatives at once and overlaps communication and computation: green squares (■) for single threaded version, and red symbols for best performing (with respect to thread count) multithreaded version with ▲, ▼, and ● for 2, 4, and 8 OpenMP threads, respectively. Cyan star (∗) for hybrid MPI-OpenMP routine in which a dedicated thread per NUMA domain performs communication and no computation. Timings shown for best performing process layout in each case. The number of processing elements is the product of the number of MPI processes and the number of OpenMP threads per MPI process.

problem size is a factor of $N_\theta/N_v = 8$ smaller. Following the discussion for figure 4.7, the velocity field core count M_v is chosen so that it does not slow down the overall simulation and is not wasteful of resources on its own. Different symbols in the figure represent performance obtained with different implementations of the CCD routines as described in §4.2. Numbers for weak and strong scaling are presented in table 4.7 for the two largest production problem sizes, namely $N_\theta = 4096$ and $N_\theta = 8192$. Weak scaling is inferred relative to the case of $N_\theta = 1024$ along the horizontal dashed lines in figure 4.8, and strong scaling for each N_θ is inferred based on the smallest number of PEs tested. Because of additional memory usage in the DNS code, the initial number of PEs used is a factor of two larger than those used in the analysis of the CCD kernel in §4.2.4. In addition, due to the cost of the runs, tests at $N_\theta = 8192$ and $524,288$ PEs were conducted with fewer process layouts than tests at smaller PE counts, and the multithreaded routines were only tested with eight threads.

The picture conveyed by figure 4.8 has a number of similarities to that in figure 4.4, suggesting that the performance of the DNS code is heavily influenced by the performance of the CCD routines. This is not surprising, since at every RK4 sub-stage of time integration the CCD routines are called twice: once to compute the derivatives of the scalar field and a second time to compute the derivatives of the velocity-scalar products. At the two smaller problem sizes ($N_\theta = 1024$ and $N_\theta = 2048$), the strong and weak scalings are robust, and the timings are only mildly sensitive to the choice of the CCD routine. As the problem size is increased to $N_\theta = 4096$ and $N_\theta = 8192$ the scalability observed is somewhat better than that seen earlier in figure 4.4, while the sensitivity to specific CCD implementations decreases. This suggests that other operations executed by the processes in the scalar field communicator also make substantial contributions to the timings, and that those other operations scale better than the CCD routines. Nevertheless, the dual-communicator code is least efficient when using the most basic version of the CCD routine which is single threaded and does

Table 4.7: Overall simulation time per step in seconds, and strong and weak scaling results for (left table) $N_\theta = 4096$ and (right table) $N_\theta = 8192$ problem sizes. Each column is for a given number of processing elements for the scalar field code (PEs, with $K \equiv 1024$), defined as the number of MPI processes multiplied by the number of OpenMP threads per MPI process. Each horizontal block represents the version of the CCD routine used by the scalar field code, with abbreviations detailed at the beginning of §4.2.4 The symbols given in parenthesis match those in figure 4.8 for each subroutine. Subroutine ON uses one MPI process per NUMA domain (4 per BW XE6 node) with 8 OpenMP threads per MPI process. For BM and OM the timing is reported for the best MPI-OpenMP configuration for a given problem size (see symbol shapes in figure 4.8 for the number of threads).

PEs	$8K$	$16K$	$32K$	$64K$
BS (□)	30.7	16.5	8.95	5.08
Weak (%)	95.5	88.8	82.7	76.1
Strong (%)	—	92.9	85.8	75.7
OS (■)	30.3	16.1	8.66	4.67
Weak (%)	96.7	91.3	85.2	81.4
Strong (%)	—	94.3	87.4	81.1
BM (△,▽,○)	30.3	15.7	8.34	4.74
Weak (%)	95.5	92.7	88.2	79.5
Strong (%)	—	96.4	90.9	80.0
OM (▲,▼,●)	29.8	15.3	8.13	4.43
Weak (%)	97.4	95.0	89.9	84.1
Strong (%)	—	97.3	91.5	83.9
ON (⁎)	30.1	15.0	7.62	3.99
Weak (%)	98.6	98.8	97.9	95.2
Strong (%)	—	100	98.7	94.3

PEs	$64K$	$128K$	$256K$	$512K$
BS (□)	34.8	20.4	10.8	6.57
Weak (%)	84.2	72.2	68.8	58.7
Strong (%)	—	85.6	80.8	66.2
OS (■)	33.4	18.3	9.51	6.09
Weak (%)	87.6	80.3	77.6	62.4
Strong (%)	—	91.5	87.8	68.6
BM (△,▽,○)	34.4	18.4	9.99	5.59
Weak (%)	84.4	79.3	73.7	67.4
Strong (%)	—	93.4	86.0	76.8
OM (▲,▼,●)	32.6	17.8	9.34	5.29
Weak (%)	88.8	81.8	78.2	70.5
Strong (%)	—	91.8	87.3	77.2
ON (⁎)	30.7	16.2	8.15	4.51
Weak (%)	96.7	91.5	91.6	84.2
Strong (%)	—	94.7	94.1	85.1

not overlap communication and computation (version BS marked with open black squares). Consistent with the CCD performance data in §4.2.4, even with blocking communication, performance can be improved by reducing the number of MPI processes through hybrid programming. For both the single threaded and multithreaded versions, greater gains in performance are obtained when non-blocking communication is used. However, the best results are obtained when the OpenMP threads are split to perform separate communication and computational work (version ON marked with cyan stars).

To quantify the costs of various portions of the code, table 4.8 shows a detailed

Table 4.8: A breakdown of the cost of each time step into contributions from various operations, reported by MPI processes responsible for the scalar field. The timings reported are the average timings over all MPI processes and steps (excluding the first and last step). The scalar field code uses the ON version of the CCD routine. The numbers of PEs used for the scalar are $32,768$ and $262,144$, i.e., 1024 and 8192 BW XE6 nodes, respectively.

	$N_v = 512$, $N_\theta = 4096$		$N_v = 1024$, $N_\theta = 8192$	
Operation	Time (s)	Percent (%)	Time (s)	Percent (%)
Scalar Derivatives	2.30	29.7	2.59	30.5
Diffusion Term	0.28	3.6	0.28	3.3
Receive Velocity	0.07	1.0	0.20	2.4
Interpolation	2.12	27.3	2.14	25.2
Advection Derivatives	2.36	30.4	2.61	30.8
Advection Term	0.28	3.6	0.28	3.3
Time Advance	0.28	3.7	0.28	3.3
Sum and % of Total	7.70	99.4	8.40	98.9

breakdown of the wall clock time per step for problem sizes $N_\theta = 4096$ and $N_\theta = 8192$. The operations presented were summarized previously in figure 4.6. The focus is on the average instead of the minimum or maximum timings to ensure that the total percentage is close to 100%. The successive rows in the table correspond to the order in which different operations occur at every RK4 sub-stage, but only including those which account for at least 1% of the wall time per step.

Based on the timings reported in table 4.8, calculating the derivatives of the scalar field takes about 30% of the overall cost. The diffusion term is then calculated by simply adding the three second derivatives and multiplying them by the molecular diffusivity. At this point the velocity field must be transferred from the velocity field communicator to form the advection terms. The cost of the transfer is between 1%–3% of the overall cost of the simulation. The code then enters a routine in which tricubic interpolation is used to interpolate the velocity field onto the fine grid. The interpolated values at each point of the finer scalar grid are not stored but are used immediately to form the advection and mean-gradient forcing terms.

This interpolation accounts for 25% of the overall cost. The next step is to calculate the derivatives of the velocity-scalar products, which are required to complete the skew-symmetric form of the advection terms. This cost is approximately equal to the cost of calculating the scalar derivatives in the first operation, i.e., 30% of the overall cost. Although, strictly speaking, only the first (not the second) derivatives of the velocity-scalar products is required, it is important that this first derivative is computed using the same numerical scheme, i.e., the same CCD equations, as done for the derivatives of the scalar field itself. The skew-symmetric form of the advection terms is then finalized (Operation "Advection Term"). At this point the scalar field uses the time step from the velocity-field processors to advance the scalar field. The transfer of the time step itself (a single word) takes a trivial amount of time and is not reported in the table.

When comparing results for the two problem sizes in table 4.8 it is clear that as the problem size is increased from 4096^3 to 8192^3, operations that involve communication ("Scalar Derivatives", "Receive Velocity", and "Advection Derivative") show a non-trivial increase in absolute timings with an accompanied mild increase in percentage contribution. On the other hand, the interpolation shows near-perfect weak scaling, as expected. One possibility for further optimization is the use of accelerator-based heterogeneous computing, especially the use of Graphical Processing Units (GPUs) that allow substantial speedups in computation. Such an effort is reported next.

4.4 Acceleration of DNS code using OpenMP 4.5

This section provides the details of how GPUs are utilized to accelerate the DNS code. Since the total cost of a simulation is almost entirely dictated by the scalar field computation, only the operations carried out in the scalar field communicator are accelerated. In §4.4.1, an overview of how the DNS code is modified for heterogeneous computing environments is given, and an emphasis is placed on how operations are

scheduled between CPUs and GPUs to minimize data movement. In §4.4.2, the parallel implementation of the compact scheme is discussed, followed by a presentation of how OpenMP 4.5 is used to perform the acceleration in §4.4.3.

4.4.1 Memory management and schedule of operations

In both homogeneous and heterogeneous computing environments, a major factor influencing algorithm design is that of memory placement on the underlying hardware. While the focus in homogeneous computing is on NUMA latencies, in heterogeneous environments additional complications arise when memory spaces on the CPU and GPU are distinct. For example, while each CPU on the XK7 architecture has access to 32 GB of memory, each Kepler GPU provides only 6 GB of memory, which imposes a severe constraint on algorithm design. To minimize the number of nodes required for the simulations while still allowing the entire memory space to fit into the GPUs, the code has been refactored to reduce memory usage. The largest arrays dominating the memory usage are proportional to N_θ^3 in volume, and include the three arrays required for RK4 time integration, and six arrays for derivatives (first and second derivatives in each coordinate direction). A substantial reduction in memory usage compared with the CPU-only version of the code is achieved by modifying the RK4 routine so that every term represented on the right-hand side of (4.5) is immediately added to the partial Runge-Kutta updates, without using additional arrays to store them. This re-factoring reduces the number of Cray XK7 nodes needed for the main target problem size ($N_\theta = 8192$) from 16,384 for the scalar to 8192, which is also more economical in terms of resources charged.

While the reductions in memory reduced the number of nodes required for the simulations, the avoidance of additional arrays increased the complexity of the RK4 time integration procedure. In addition, now that the scalar field is computed entirely on the GPU, additional memory transfers of the incoming velocity field from the

Table 4.9: Summary of the operations during each sub-stage of RK4 time integration. The second column lists the device for each operation, with "All" meaning that the CPU, PCI bus, and GPU are used. Steps that occur within the same loop nest are appended with letters, e.g., 2A and 2B must occur in the same loop because otherwise 2B could not occur after the scalar field is updated during 2A. The code uses Fortran derived datatypes for the CCD scheme, and the `ds1`, `ds2`, and `ds3` arrays are 3-D arrays of the derived datatypes, which contain two elements each, e.g., `ds1%a` for the first element, and `ds1%b` for the second element. Upon return from the CCD subroutine, the first element contains the first derivative, and the second element contains the second derivative.

Step	Device	Operation Summary
1	All	Calculate and store scalar derivatives in x_1, x_2, x_3 in `ds1`, `ds2`, `ds3`
2A	GPU	RK4 diffusion term (get 2^{nd} derivatives from `ds1%b`, `ds2%b`, `ds3%b`)
2B	GPU	Store current sub-stage value of scalar in `ds3%b`
3	CPU	Receive velocity field and fill ghost layers with halo exchange
4	PCI	Transfer velocity field from CPU to GPU
5	GPU	Interpolate u_1 and u_2 velocities into `ds1%b` and `ds2%b` arrays
6A	GPU	RK4 non-conservative advective and forcing terms in x_1 and x_2
6B	GPU	Store θu_1 in `ds1%a` and θu_2 in `ds2%a` (get θ from `ds3%b`)
7	GPU	Interpolate u_3 velocity into `ds1%b` array
8A	GPU	Increment RK4 non-conservative advective and forcing terms in x_3
8B	GPU	Store θu_3 in `ds3%a` (get θ from `ds3%b`)
9	All	Calculate advective derivatives in x_1, x_2, x_3 in `ds1`, `ds2`, `ds3` arrays
10	GPU	RK4 conservative advective terms in all coordinate directions

velocity communicator to the GPU through the PCI bus are required. As a result, the time integration procedure in the accelerated code contains a few more steps compared to the algorithm shown in figure 4.6 for the CPU code. Table 4.9 shows an outline of the operations for each RK4 sub-stage of the scalar field in the new CPU-GPU code. As in the CPU version, the code begins each RK4 sub-stage by calling CCD routines with the scalar field and computes first and second derivatives in each direction (step 1). As part of the effort to reduce memory usage, the arrays used to store these derivatives are re-used for other purposes as much as possible. For example, after the diffusion term is added to the RK4 updates (step 2A) the memory for the second derivatives is freed, and one slot is used to hold the value of the scalar

field at the current sub-stage (step 2B). Next, after communication of the velocity field to the scalar communicator is complete (step 3), it is transferred to the GPU (step 4), where there is enough free memory to interpolate two velocity components (step 5). The code is now in a position to calculate other terms on the right-hand side of (4.5) which involve the first two velocity components (step 6A, 6B). After this is done, storage space is now available for operations involving the third velocity component (steps 7, 8A, 8B). Finally, the code collects information required for the skew-symmetric form of the advective terms (step 9) which completes the current RK4 sub-stage (step 10).

4.4.2 Asynchronous algorithm for the CCD scheme

While the application of the CCD scheme is the most expensive operation in a simulation, there are considerable opportunities to improve the scalability and performance of the CCD subroutines (and thereby the DNS code) by overlapping communication with computation as much as possible. Such strategies were explored in §4.2 for the CPU version of the code, and here they are extended to the algorithms used in heterogeneous computing environments with distinct memory spaces for the CPU and GPU. Both algorithms rely on the static 3-D domain decomposition made possible by the parallel algorithm used to solve the CCD equations (Nihei & Ishii, 2003). Under a 3-D domain decomposition, since a direct solution of the CCD system in any coordinate direction is not possible, a series of alternating communication and computation operations are used to assemble the solution to the entire CCD linear system. All communication operations are directional, taking place in sub-communicators of the 3-D domain decomposition containing the processors which share grid points along a common grid line (illustrated in figure 4.5).

A detailed schedule of the operations required to solve the CCD system in parallel for a single coordinate direction in heterogeneous computing environments is shown

Table 4.10: Summary of sequential operations required to apply the CCD scheme in a single coordinate direction in a heterogeneous computing environment with distinct CPU and GPU memory spaces. When using a 3-D domain decomposition, the communication calls in steps 3, 8, and 10 reside in the directional sub-communicator in the direction the derivatives are being taken.

Step	Device	Operation Summary
1	GPU	Pack ghost layer information into contiguous buffer
2	PCI	Update ghost layer information from GPU to CPU
3	CPU	Exchange ghost layers with `MPI_SENDRECV`
4	PCI	Update ghost layer information from CPU to GPU
5	GPU	Form right-hand side of linear system and obtain solution
6	GPU	Pack data required for reduced linear system
7	PCI	Update data for reduced linear system from GPU to CPU
8	CPU	Distribute data for reduced linear system with `MPI_ALLTOALL`
9	CPU	Unpack data and solve reduced linear system
10	CPU	Distribute data for final solution with `MPI_ALLTOALL`
11	PCI	Update data required for final solution from CPU to GPU
12	GPU	Finalize solution of CCD linear system

in table 4.10. At the start of this process, it is assumed that the signal to be differentiated resides in the GPU memory space. (In the DNS code, the scalar fluctuations and the advective terms are differentiated, as shown in steps 1 and 9 of table 4.9.) Because the CCD scheme has a finite stencil, ghost layers must be exchanged with neighboring processors, which requires packing ghost layer information into buffers on the GPU (step 1) and then updating the CPU with that data (step 2). After the ghost layers are exchanged (step 3), the GPU is updated (step 4), after which the first computations for the CCD scheme can proceed (step 5). As part of the parallel algorithm necessitated by the 3-D domain decomposition, the processes along a given grid line must coordinate to solve a reduced linear system. This requires packing some of the results computed in step 5 on the GPU (step 6), so the CPU can be updated (step 7) for the required communication (step 8). Importantly, the required computations for the reduced linear system are trivially small, so that leaving the computations on the CPU (step 9) is more economical than introducing additional

memory movements to solve the reduced linear system on the GPU. Once the reduced linear system is solved, the results are redistributed to the other processes (step 10), after which the GPU is updated (step 11) and the final solution to the CCD scheme is obtained (step 12).

In order to overlap communication with computation for the CCD scheme, the code exploits the fact that the operations listed in table 4.10 for each coordinate direction are independent of one another. The operations in table 4.10 for all coordinate directions can therefore be interleaved and the asynchronous capabilities of heterogeneous computing environments can be used to achieve the desired overlap. In the algorithm, this is achieved by launching all available GPU kernels for a given coordinate direction asynchronously, so that the master thread on the CPU can immediately proceed to a communication call for another coordinate direction. To illustrate how these concepts are used in the code, a detailed schedule of the operations for the asynchronous algorithm is shown in table 4.11. Because communication in the x_1 direction includes some intra-node communication and is therefore more robust than communication in x_2 or x_3, the algorithm begins by filling ghost layers in x_1, which requires packing a buffer (step 1) and updating the CPU with the ghost layer information (step 2). However, before proceeding with the `MPI_SENDRECV` call to exchange the ghost layers, the necessary operations to pack ghost layers and update the CPU for the x_2 and x_3 directions are launched asynchronously (steps 3-6), so they may occur concurrently with the ghost layer exchange (step 7). Once the ghost layers in x_1 are exchanged, a series of kernels are launched asynchronously (steps 8-11) to perform the initial computations in the x_1 direction. Because the kernels for the x_1 direction are launched asynchronously, the master thread can immediately proceed to verify that the x_3 ghost layer information is updated on the CPU (step 12) and then exchange the ghost layers with `MPI_SENDRECV` (step 13). The general pattern of launching kernels for one coordinate direction, while proceeding to synchronize and then communicate

Table 4.11: Operations for asynchronous CCD algorithm. Operations which do not have their execution deferred appear in the "Blocking" column, while operations that are launched asynchronously on the GPU appear in the "Asynchronous" column. Order of operations enforced with dependencies given in the "Depends" column.

Step	Depends	Blocking	Asynchronous
1		Pack Ghost x_1	
2		Update CPU x_1	
3			Pack Ghost x_3
4	3		Update CPU x_3
5			Pack Ghost x_2
6	5		Update CPU x_2
7		`SENDRECV` x_1	
8			Update GPU x_1
9	8		Compute x_1
10	9		Pack Data x_1
11	10		Update CPU x_1
12	4	Synchronize x_3	
13		`SENDRECV` x_3	
14			Update GPU x_3
15	14		Compute x_3
16	15		Pack Data x_3
17	16		Update CPU x_3
18	6	Synchronize x_2	
19		`SENDRECV` x_2	
20			Update GPU x_2
21	20		Compute x_2
22	21		Pack Data x_2
23	22		Update CPU x_2
24	11	Synchronize x_1	
25		`ALLTOALL` x_1	
26		Reduced sys. x_1	
27		`ALLTOALL` x_1	
28			Update GPU x_1
29	28		Final soln. x_1
30	17	Synchronize x_3	
31		`ALLTOALL` x_3	
32		Reduced sys. x_3	
33		`ALLTOALL` x_3	
34			Update GPU x_3
35	34		Final soln. x_3
36	23	Synchronize x_2	
37		`ALLTOALL` x_2	
38		Reduced sys. x_2	
39		`ALLTOALL` x_2	
40			Update GPU x_2
41	40		Final soln. x_2
42	29,35,41	Sync. x_1,x_2,x_3	

for another coordinate direction continues until all operations listed in table 4.10 are completed for all coordinate directions. While the exact amount of overlap that can be achieved depends on the cost of each communication and computation step listed in table 4.11, the current algorithm strives to ensure that the master threads on the CPUs are at attempting to communicate at all times, which is necessary to achieve the maximum benefit.

4.4.3 Implementation using OpenMP 4.5

Now that the accelerated algorithm for the DNS code (§4.4.1) and the asynchronous algorithm for the CCD scheme (§4.4.2) have been described, their implementations for production simulations on the Cray XK7 machine Titan are presented. To avoid extensive code rewrites, a major objective when porting the code to run on GPUs was to keep the original Fortran code base intact. By using a directive-based approach such as OpenMP or OpenACC, many computational kernels can be offloaded to the GPUs with relative ease, thus allowing one to focus on the higher-level code restructuring that is required to run on GPUs, e.g., controlling the data flow between the CPU and GPU memory spaces, and reorganizing the RK4 subroutine due to memory constraints. OpenMP is well-suited to address the needs of the DNS code, since in OpenMP 4.5 the essential clauses to enable asynchronous GPU algorithms were added to the device constructs, i.e., the `TARGET` tasks, which are needed to implement the asynchronous CCD algorithm presented in table 4.11. The Cray Compiler Environment (CCE) has supported many OpenMP 4.5 features since release 8.5, and in all of the work for the accelerated code CCE version 8.5.7 has been used.

Before any computational work can be done on the GPUs, the user must ensure that the memory space on the GPU is properly initialized with the required data. As described in the OpenMP 4.5 standard (OMP, 2015), the OpenMP model for accelerated computing is "host-centric," meaning that data resides on the host (i.e.,

the CPU) until it is explicitly mapped to the device (i.e., the GPU) by the user for the required computations. In OpenMP, the basic construct used to perform GPU computations is the `TARGET` construct, which can be appended with `MAP` clauses to specify memory movements or memory allocations which are required for a single computational kernel. Data movements can be minimized with the `TARGET DATA` construct, which allows the user to create a data environment on the GPU that is inherited by all `TARGET` constructs contained within the region generated by the `TARGET DATA` construct. While the arrays are mapped once to the GPU with `TARGET DATA`, at times the GPU must be explicitly updated with data from the CPU (e.g., the updated velocity field at each RK4 sub-stage), or (on occasion) the CPU must be updated with data from the GPU to perform certain statistical analyses (e.g., obtain the scalar spectrum by taking the FFT of the scalar field on the CPUs). Such explicit data movements are made with the `TARGET UPDATE` construct, where the clause `TO` indicates that data is moved from the CPU to the GPU, and the clause `FROM` indicates that data is moved from the GPU to the CPU.

The current usage of data movement constructs provided by OpenMP is illustrated in the pseudo-code in figure 4.9. The top frame outlines the execution of the main Fortran program, which begins by initializing all memory on the CPU required for the simulation. Once the code is initialized, the velocity field communicator proceeds to time advance the velocity field (a CPU-only calculation), while the scalar field processors must continue to initialize the GPU memory space for the scalar field computation. All arrays required for the scalar field computation are initialized once on the GPU with a `TARGET DATA` region surrounding the entire time-advance loop. During time stepping, if a checkpoint of the scalar field is to be written out, the explicit data movement is made with `TARGET UPDATE FROM` (line 14 in the top frame of figure 4.9), which ensures the required data is transferred from the GPU to the CPU before it is written out. The bottom frame of figure 4.9 also shows the general

```
PROGRAM DNS_CODE
! Code initialization.
IF (velocity_field_process) THEN
   ! Time advance the velocity field.
ELSE IF (scalar_field_process) THEN
   ! Move data to the GPU.
   !$OMP TARGET DATA MAP(TO:scalar_field,...)
   ! Time advance the scalar field.
   DO step=1,num_steps
      CALL RK4_TIME_ADVANCE
      !
      ! Periodically write checkpoint on CPU.
      IF (checkpointing_step) THEN
         !$OMP TARGET UPDATE FROM(scalar_field)
         CALL WRITE_CHECKPOINT(scalar_field)
      END IF
   END DO
   !$OMP END TARGET DATA
END IF
END PROGRAM DNS_CODE
```

```
SUBROUTINE RK4_TIME_ADVANCE
! Create data environment for subroutine.
!$OMP TARGET DATA MAP(TO:scalar_field,...)
DO runge_kutta_stage=1,4
   ! Perform computations, e.g., calculate
   ! derivatives of scalar field.
   CALL CCD_SUBROUTINE(scalar_field,ds1,...)
   !
   ! Receive velocity field and update GPU.
   CALL RECEIVE_VELOCITY_FIELD(u1,u2,u3)
   !$OMP TARGET UPDATE TO(u1,u2,u3)
   !
   ! Continue computations, e.g., form
   ! advective and forcing terms.
   CALL TRICUBIC_INTERPOLATION(u1,ds1)
   CALL TRICUBIC_INTERPOLATION(u2,ds2)
END DO
!$OMP END TARGET DATA
END SUBROUTINE RK4_TIME_ADVANCE
```

Figure 4.9: Pseudo-code illustration of how OpenMP `TARGET DATA` and `TARGET UPDATE` constructs are used to create data regions in the code and to explicitly move data, respectively. In the top frame, an outline of the main Fortran program file is given, where before time stepping begins all data is mapped to the GPU. The bottom frame includes some code snippets from the RK4 subroutine, with an emphasis on some of the subroutines called and the data movements made for the three velocity components, which are stored in the `u1`, `u2`, and `u3` arrays.

structure and data movements that occur in the RK4 subroutine. While it is known that the working arrays for the simulation are already mapped to the GPU in the main program file, each subroutine also includes a `TARGET DATA` region so the routines can function in a standalone manner, if ever necessary. In the RK4 subroutine, the important data movement illustrated in the pseudo-code (lines 10-11) involves obtaining the velocity field from the velocity field communicator, and then updating the GPU memory space with a `TARGET UPDATE TO` construct.

As mentioned previously, the most expensive operation in the code is the application of the CCD scheme. In the CCD subroutine, all work on the GPU, either in the form of actual computations or packing memory buffers, is made possible with the `TARGET` construct. The computational kernels which are launched, i.e., the `TARGET` tasks, are in the form of nested Fortran `DO` loops, which the Cray compiler auto-parallelizes for GPU threading when the construct includes the essential ingredients `TARGET TEAMS` and `DISTRIBUTE`. As others have noted (Lopez *et al.*, 2016), the necessary clauses that must be added to the `TARGET` construct to get good performance can be vendor-specific, and with CCE version 8.5.7 it appears that the compiler can well-parallelize a loop for the GPU with `TARGET TEAMS` and `DISTRIBUTE` as long as the loop nest is clearly vectorizable (often checked by viewing the compiler listings). In order to make the execution between the CPU and GPU asynchronous, the GPU kernels launched with the `TARGET` construct are appended with the task-oriented clauses added in OpenMP 4.5. To make a `TARGET` task deferred, i.e., "non-blocking," such that the CPU thread can progress on to subsequent tasks while the GPU kernel is running concurrently, the `NOWAIT` clause must be used. While the `NOWAIT` clause enables the CPU thread to move on with its execution, some of the other tasks the CPU will perform include launching GPU kernels which depend on the completion of previously-launched kernels. To ensure the proper order of execution on the GPU, the `DEPEND` clause must also be used, where `DEPEND(IN:var)`, `DEPEND(OUT:var)`, and

`DEPEND(INOUT:var)` can be used to control the sequence of execution based on data dependencies (e.g., a variable `var`).

To elucidate these concepts further, figure 4.10 provides pseudo-code which highlights how the OpenMP 4.5 `DEPEND` and `NOWAIT` clauses are utilized to implement an asynchronous algorithm for the CCD scheme. In this version of the subroutine, there is a single input signal, which is to be differentiated in all three coordinate directions. After the GPU data environment is initialized (lines 11-12), a kernel is launched to pack ghost layer information for the x_1 direction (lines 14-19), which is then followed up with a data movement from the GPU to the CPU (lines 21-22). Before proceeding with the MPI communication call to exchange ghost layers in the x_1 direction, the necessary kernels to pack and update the CPU with ghost layer information in x_2 and x_3 are launched. Focusing just on the operations for the x_3 direction, a kernel is launched asynchronously to pack the necessary buffer (lines 24-29) with an outward dependency on a dummy variable `sync_3`, which is used throughout the subroutine to order the execution of x_3-direction `TARGET` tasks. Because the subsequent task to update the CPU with the x_3-direction ghost layer information (lines 31-32) contains an inward dependency on `sync_3`, it cannot begin until the packing task is complete. With work on the GPU progressing for the x_3 and x_2 directions, the CPU proceeds to exchange ghost layers in the x_1 direction (lines 43-44). Once the ghost layers are exchanged, all available tasks for the x_1 direction are launched asynchronously (lines 46-64), with the proper order of operations enforced by using the `DEPEND` clause with the dummy variable `sync_1`.

With much of the initial work for the x_1 direction underway on the GPU, ghost layers in the x_3 direction can be exchanged, but only after making sure that the previously-enqueued asynchronous update (lines 31-32) is complete. This is accomplished with an empty `TARGET` task with a dependency on `sync_3` (lines 66-70), and once this synchronization task is complete, the ghost layers are exchanged (lines 72-

```
1  SUBROUTINE CCD_SUBROUTINE(signal,ds1,ds2,ds3)
2  ! Array with signal to differentiate.
3  REAL,DIMENSION(:,:,:),INTENT(INOUT) :: signal
4  ! Arrays to hold derivatives.
5  TYPE(VECTOR),DIMENSION(:,:,:),INTENT(OUT) :: ds1,ds2,ds3
6  ! Arrays to move data between CPU and GPU.
7  REAL,DIMENSION(:,:) :: buf_1,buf_2,buf_3
8  ! Dummy variables for task dependencies.
9  INTEGER :: sync_1,sync_2,sync_3
10
11 ! Map data to accelerator.
12 !$OMP TARGET DATA MAP(TO:signal,ds1,ds2,ds3,buf_1,...)
13 ---------------------------------------------------------------
14 ! Pack X1 ghost layers (step 1)
15 !$OMP TARGET TEAMS
16 !$OMP DISTRIBUTE
17 <Pack buf_1 with X1 ghost layer information in signal>
18 !$OMP END DISTRIBUTE
19 !$OMP END TARGET TEAMS
20
21 ! Update CPU with X1 ghost layer information (step 2)
22 !$OMP TARGET UPDATE FROM(buf_1)
23 ---------------------------------------------------------------
24 ! Asynchronously pack X3 ghost layers (step 3)
25 !$OMP TARGET TEAMS DEPEND(OUT:sync_3) NOWAIT
26 !$OMP DISTRIBUTE
27 <Pack buf_3 with X3 ghost info. in signal>
28 !$OMP END DISTRIBUTE
29 !$OMP END TARGET TEAMS
30
31 ! Asynchronously update CPU with X3 ghost info. (step 4)
32 !$OMP TARGET UPDATE FROM(buf_3) DEPEND(INOUT:sync_3) NOWAIT
33 ---------------------------------------------------------------
34 ! Asynchronously pack X2 ghost layers (step 5)
35 !$OMP TARGET TEAMS DEPEND(OUT:sync_2) NOWAIT
36 !$OMP DISTRIBUTE
37 <Pack buf_2 with X2 ghost layer information in signal>
38 !$OMP END DISTRIBUTE
39 !$OMP END TARGET TEAMS
40
41 ! Asynchronously update CPU with X2 ghost info. (step 6)
42 !$OMP TARGET UPDATE FROM(buf_2) DEPEND(INOUT:sync_2) NOWAIT
```

Figure 4.10: See caption on next page.

```
43 ! Exchange ghost layers in X1 (step 7)
44 CALL MPI_SENDRECV(buf_1,...)
45
46 ! Asynchronously update GPU with X1 ghost info. (step 8)
47 !$OMP TARGET UPDATE TO(buf_1) DEPEND(OUT:sync_1) NOWAIT
48
49 ! Asynchronously launch computations in X1 (step 9)
50 !$OMP TARGET TEAMS DEPEND(INOUT:sync_1) NOWAIT
51 !$OMP DISTRIBUTE
52 <Compute in X1 with results going into ds1>
53 !$OMP END DISTRIBUTE
54 !$OMP END TARGET TEAMS
55
56 ! Asynchronously pack data in X1 (step 10)
57 !$OMP TARGET TEAMS DEPEND(INOUT:sync_1) NOWAIT
58 !$OMP DISTRIBUTE
59 <Pack required X1 data in buf_1>
60 !$OMP END DISTRIBUTE
61 !$OMP END TARGET TEAMS
62
63 ! Asynchronously update CPU with X1 info. (step 11)
64 !$OMP TARGET UPDATE FROM(buf_1) DEPEND(INOUT:sync_1) NOWAIT
65 ---------------------------------------------------------------
66 ! Synchronize X3 ghost update (step 12)
67 !$OMP TARGET DEPEND(IN:sync_3)
68 ! This is an empty, included task, which cannot begin until
69 ! the X3 ghost update (step 4) is complete.
70 !$OMP END TARGET
71
72 ! Exchange ghost layers in X3 (step 13)
73 CALL MPI_SENDRECV(buf_3,...)
74 ---------------------------------------------------------------
75 ! Rest of algorithm (steps 14-41).
76 <Compute and communicate in X1, X2, and X3>
77 ---------------------------------------------------------------
78 ! Synchronize final X1, X2, and X3 computations (step 42).
79 !$OMP TASKWAIT
80
81 ! Close structured data region.
82 !$OMP END TARGET DATA
83
84 END SUBROUTINE CCD_SUBROUTINE
```

Figure 4.10: Pseudo-code illustration of initial steps in the CCD subroutine, with an emphasis how the OpenMP 4.5 clauses `DEPEND` and `NOWAIT` are used to make CPU and GPU execution asynchronous. Entire subroutine listing split over two pages, progressing from line 1 to line 84 consecutively. Horizontal lines correspond to those in table 4.11, and separate operations for different coordinate directions.

73). The current approach of performing all data movements asynchronously (i.e., launched with `NOWAIT`) and following them up with empty `TARGET` tasks is necessary to achieve the proper overlap with CCE version 8.5. One might, for example, simply attempt to update data on the CPU when it is needed with a `TARGET UPDATE FROM` construct, devoid of the `NOWAIT` clause. Currently it appears that with such an approach, the underlying code for the data movement between the CPU and GPU cannot be overlapped with computations on the GPU, since pinned memory is not used; these issues should be cleared up in CCE 8.6 (private communication). After progressing through the entire cycle of operations detailed in table 4.11, the final synchronization required for steps 29, 35, and 41 in table 4.11 is accomplished with `TASKWAIT`, after which the required derivatives are obtained.

4.4.4 Performance of accelerated DNS code

The performance of the GPU-accelerated DNS code is now reported. Tests are conducted on Titan, a 27 petaflop heterogeneous machine which derives most of its floating point performance from the Nvidia K20 Kepler GPUs attached to each compute node. When gathering performance data for accelerated codes, a common practice is to run the same tests with the original unaccelerated code (i.e., the CPU-only code) in order to assess how well the code was accelerated. Performance can be reported in terms of the speedup of the accelerated code relative to the original code, although a drawback of this approach is that the speedup metric is highly dependent on the particular node configuration, and does not clearly show how well either code utilizes the underlying hardware. If the speedup is, for example, reported relative to the serial version of the CPU-only code, a much larger speedup is certainly attained compared to the case that timings are reported using a parallel version of the CPU-only code which utilizes the full capabilities of the CPU portion of the node. At this time, low-level performance metrics have not been measured, so the focus will be on

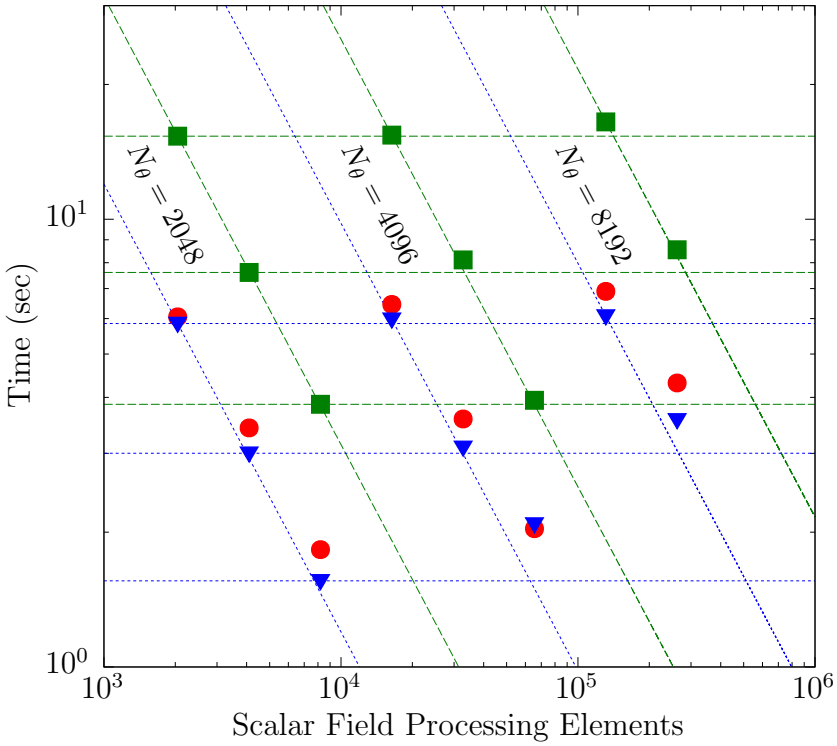

Figure 4.11: Overall simulation time per step for the GPU-accelerated code as a function of the number of processing elements for the scalar field communicator. The symbols indicate which version of the code was used for the timings: ■ for the CPU-only code, ● for the GPU code without asynchronous execution, and ▼ for the GPU code with asynchronous execution. Horizontal and sloped lines to mark strong and weak scaling datasets for the CPU-only code (green dashed lines) and the GPU code with asynchronous execution (blue dotted lines).

the strong and weak scaling of the accelerated code, and its speedup relative to the CPU-only code using the best performing (parallel) configurations for both codes for each problem.

The performance of the original code and the accelerated code on Titan using a grid ratio $N_\theta/N_v = 8$ for $N_\theta = 2048$, 4096, 8192 problems sizes is presented in graphical and tabular forms in figure 4.11 and table 4.12, respectively. Strong and weak scaling datasets for the CPU-only code and the accelerated code with asynchronous capabilities are marked in figure 4.11 with sloped and horizontal lines. Each strong scaling dataset begins at the minimum number of nodes (each node having 16 PEs)

Table 4.12: Overall simulation time per step in seconds, and strong and weak scaling results for (left table) $N_\theta = 2048$, (middle table) $N_\theta = 4096$, and (right table) $N_\theta = 8192$ problem sizes using grid ratios $N_\theta/N_v = 8$. Each column is for a given number of processing elements for the scalar field code (PEs, with $K \equiv 1024$), defined as the number of MPI processes multiplied by the number of OpenMP threads per MPI process. Each horizontal block is for a specific version of the code: top block for the CPU-only code, middle block for GPU code with asynchronous execution off (A-Off), and bottom block for GPU code with asynchronous execution on (A-On). Strong scaling data (abbreviated "Str.") given with respect to the minimum PE count used for a given problem size, and weak scaling data for each version of the code given with respect to $N_\theta = 2048$ timings for the same version of the code. For the GPU codes (middle and bottom blocks), the speedup relative to the CPU code is the ratio of the CPU to the GPU timings for the same PE count. Symbols correspond to those used in figure 4.11 for each code tested.

PEs	$2K$	$4K$	$8K$
CPU (■)	15.3	7.61	3.86
Str. (%)	—	100	99.3
A-Off (●)	6.05	3.42	1.83
Speedup	2.53	2.22	2.11
Str. (%)	—	88.5	82.8
A-On (▼)	5.85	3.00	1.56
Speedup	2.62	2.53	2.48
Str. (%)	—	97.4	93.9

PEs	$16K$	$32K$	$64K$
CPU (■)	15.4	8.11	3.94
Str. (%)	—	95.0	97.8
Weak (%)	99.5	93.8	97.9
A-Off (●)	6.45	3.58	2.04
Speedup	2.39	2.27	1.94
Str. (%)	—	90.1	79.2
Weak (%)	93.8	95.5	89.8
A-On (▼)	5.98	3.09	2.09
Speedup	2.58	2.62	1.89
Str. (%)	—	96.7	71.7
Weak (%)	97.8	97.0	74.7

PEs	$128K$	$256K$
CPU (■)	16.5	8.55
Str. (%)	—	96.5
Weak (%)	92.9	89.0
A-Off (●)	6.90	4.31
Speedup	2.39	1.98
Str. (%)	—	80.1
Weak (%)	87.7	79.4
A-On (▼)	6.08	3.56
Speedup	2.71	2.40
Str. (%)	—	85.3
Weak (%)	96.2	84.3

that provides enough memory to perform the scalar field computation entirely on the GPUs. Of particular interest is the weak scaling of this minimum-node configuration because it is the configuration used for production simulations. For $N_\theta = 2048$ using 2048 PEs, the asynchronous version of the accelerated code achieves a speedup of approximately 2.6, which significantly increases the amount of science (i.e., the number of simulation time steps) obtained from a simulation. For the relatively low node count used for $N_\theta = 2048$, the communication requirements are not so demanding that enabling the asynchronous capabilities in the CCD subroutine offers significant performance benefits. As the problem is weak-scaled out to $N_\theta = 4096$ and $N_\theta = 8192$, however, significant performance gains are possible by overlapping communication

and computation in the CCD subroutine. Specifically, for $N_\theta = 8192$ at $131,072$ PEs, the performance of the accelerated code can be improved from 6.90 sec/step to 6.08 sec/step by enabling asynchronous features, which represents more than a 10% improvement in absolute performance. The resulting improvement in weak scaling relative to the $N_\theta = 2048$ data is from 88% (asynchronous execution disabled) to 96% (asynchronous execution enabled). Evidently, maintaining good strong scaling with the accelerated code on Titan over a wide range of core counts is challenging. For the main production problem of $N_\theta = 8192$, the asynchronous code only achieves 85% strong scaling when increasing from $131,072$ PEs to $262,144$ PEs (the maximum number of PEs that can be used on Titan for this problem size). Strong scaling accelerated codes is expected to be a challenge, since only the computational portions of the code are accelerated, while the communication requirements and subroutines remain unchanged.

4.5 Summary

This chapter reports on the development and performance characteristics of a dual-communicator parallel algorithm for the direct numerical simulation (DNS) of turbulent mixing at high Schmidt number (Sc). At high Sc, the passive scalar requires grid resolution higher than that for the velocity field by a factor $\sqrt{Sc}$. To save computational resources, the velocity field can be computed on a coarser grid (Brethouwer *et al.*, 2003; Gotoh *et al.*, 2012) while a finer grid is necessary for the scalar field. The velocity field is computed according to the Navier-Stokes equations on a periodic domain of N_v^3 grid points using Fourier pseudo-spectral (FPS) methods, while the scalar field is computed according to an advection-diffusion equation on the same physical domain using a combined compact finite difference (CCD) scheme on a finer grid of N_θ^3 points. Parallel processes (MPI processes) running the code are divided into two disjoint communicators for the velocity and scalar fields, respectively. Since the scalar

is dynamically passive the coupling between the two communicators is one-way, and is realized through the inter-communicator transfer of the velocity field followed by interpolation onto the finer grid. Extensive benchmarking has been performed over a wide range of problem sizes using the Cray XE6 partition of Blue Waters at the University of Illinois, Urbana-Champaign, and the Cray XK7 machine Titan at Oak Ridge National Laboratory, TN.

An advantage that CCD schemes have over their FPS counterparts is their reduced communication requirements, which improves scalability to the large problems sizes necessary to study turbulent mixing at high Sc (Gotoh *et al.*, 2014). The CCD scheme computes first and second derivatives of a field to eighth-order accuracy (Mahesh, 1998) through a block tridiagonal system of equations (4.6). As explained by Nihei & Ishii (2003) and briefly summarized in Appendix D, this linear system can be solved on a static 3-D domain decomposition without the expensive memory transposes used by FPS schemes. Since the CCD scheme takes a large fraction (60% in table 4.8) of the overall cost of the simulation it is important to pursue optimization of the CCD routines rigorously. In addition to using shared-memory programming implemented with OpenMP to reduce the number of MPI processes and thus lower the volume of ghost-layer communication traffic, non-blocking MPI calls are also used to overlap communication and computation in all three coordinate directions (§4.2.2 and table 4.3). While such approaches did improve scalability, the biggest improvement on Blue Waters came from dedicating certain threads to perform only communication while others only compute. This thread splitting approach is implemented using nested OpenMP parallelism and utilizes OpenMP locks for synchronization.

At the largest production problem size of $N_\theta = 8192$ run on $262,144$ processing elements (PEs) on Blue Waters, dedicating one thread per NUMA domain for communication improves the performance of the CCD routines by 34% compared to the single threaded implementation using blocking communication. As seen in figure 4.4

and table 4.5, for this configuration strong scaling with respect to the fewest PEs tested and weak scaling with respect to 1024^3 are 90%. It can be seen in figure 4.4 that the best CCD implementation scales well to even $N_\theta = 16384$ on $524,288$ PEs. The CrayPat performance monitoring utility has been used to measure computational performance of the CCD routines for problem sizes up to 4096^3. The results suggest that the best CCD implementations can reach nearly 6% of theoretical peak FLOP rate. The CrayPat data also indicate that the memory bandwidth as well as cache subsystems are utilized effectively.

As analyzed in §4.3, the performance of the combined dual-communicator code depends on the size of the (disjoint) scalar and velocity communicators, the coupling between the communicators, and the CCD routines. The overall resource requirement is driven by the scalar field, and the number of PEs for the velocity field is chosen in proportion to its problem size, which is a factor $(N_\theta/N_v)^3$ smaller than the scalar problem. Focusing on high Schmidt number configurations, $N_\theta/N_v = 8$ is used in the performance evaluation of the dual-communicator code. As illustrated in figure 4.5, the velocity field is first sent to the root process of the scalar field sub-communicator in the first coordinate direction with discrete send and receive operations. After it has arrived, the velocity field is scattered to other processes in the same sub-communicator. Results in table 4.8 show that this transfer is being handled very efficiently, taking less than 3% of cost of the DNS at the largest production problem size. The nature of the one-way coupling between the velocity and scalar fields allows the transfer to be performed using non-blocking communication, which allows efficient overlap of this transfer with other operations in the scalar communicator. The timing data for the DNS code presented in figure 4.8 and table 4.7 show that the dual-communicator code is achieving good scalability on Blue Waters over a wide range of problem sizes and PEs of practical interest.

The relatively low communication requirements of the current code also make it a

prime candidate for acceleration in heterogeneous computing environments. In order to run on machines like Titan, the 27 petaflop heterogeneous machine with Nvidia GPU accelerators, the code is ported to run on GPUs using the latest OpenMP 4.5 capabilities of the Cray compiler. Because the computational cost of a simulation with high grid ratios, e.g., $N_\theta/N_v = 8$, is dominated by the scalar field computation, these efforts focus solely on accelerating the computation of the scalar field. Memory movement between the CPUs and GPUs is minimized by transferring the entire memory space required for the scalar field computation to the GPUs. With this approach, data movement between the CPUs and GPUs is limited to just the transfer of the velocity field and the data required for the evaluation of the CCD scheme. Motivated by the improved scalability found on Blue Waters when communication and computation were explicitly overlapped in the CCD subroutine, the latest capabilities of OpenMP 4.5 are used in a version of the CCD subroutine for heterogeneous computing environments which overlaps communication and computation by allowing the CPUs and GPUs to operate asynchronously. The routine utilizes the task-oriented clauses, e.g., `DEPEND` and `NOWAIT`, added to the accelerator (`TARGET`) constructs in OpenMP 4.5 to allow the CPUs to immediately proceed to a required communication call after asynchronously launching computational kernels. For a target production problem of $N_\theta = 8192$ on Titan using $131,072$ PEs (8192 Titan XK7 nodes), absolute performance is improved by over 10% when using the asynchronous CCD subroutine, resulting in a weak scaling of 96% relative to $N_\theta = 2048$ timings. The overall speedup compared to the CPU-only version of the code at this problem size is 2.7.

To summarize, the motivation for this chapter was to develop a new parallel code capable of efficiently simulating turbulent mixing at high Schmidt number. The target production problem size is 8192^3 (more than 0.5 trillion) grid points on large massively parallel systems such as Blue Waters and Titan. The key strategy is to exploit opportunities for overlapping between independent operations whenever the

physics of the passive scalar problem or other computations allow. The codes use shared-memory programming, in particular OpenMP's nested parallelism capabilities to optimize the CCD routines, and a dual-communicator formulation which ultimately leads to substantial enhancement in performance compared to recent work by Gotoh *et al.* (2012). The code was also successfully ported to run in heterogeneous computing environments using the latest techniques in OpenMP 4.5, resulting in a significant improvement in absolute performance compared to CPU-only execution.

CHAPTER V

TURBULENT MIXING AT HIGH SCHMIDT NUMBER

Turbulent mixing of high Schmidt number scalars is relevant to many flows of interest, including industrial mixing of liquids and the mixing of organic matter and salinity in the ocean. Conducting DNS of such flows is challenging due to the increased resolution requirements of the Batchelor scales, which have often limited the study of very high Schmidt number scalars to low Reynolds numbers (Donzis & Yeung, 2010). Because of limited data, there are many open questions for high Schmidt number passive scalars in high Reynolds numbers turbulence (Gotoh & Yeung, 2013). For example, the shape of the scalar spectrum at these extreme conditions is still an open question (Warhaft, 2000; Gotoh *et al.*, 2014). Also, the status of local isotropy and the saturation of small-scale intermittency in the scalar field, while previously examined at low Reynolds numbers for very high Schmidt numbers (Yeung *et al.*, 2002, 2004; Donzis & Yeung, 2010), need further examination at higher Reynolds numbers. With the numerical algorithms developed in Chapter IV and access to petascale computational resources, a great opportunity is now available to provide insight into these questions and more for high Schmidt number passive scalars.

This chapter presents a DNS database used to investigate the Schmidt number dependence of passive scalar statistics in forced $R_\lambda \approx 140$ isotropic turbulence. This Reynolds number is sufficiently high to support a narrow inertial range in the velocity field (Yeung & Zhou, 1997). A description of the DNS database is provided in §5.1, and includes information like the resolution of the velocity field and scalar field for each simulation. In §5.2 results are examined for a single Schmidt number simulated with multiple grid configurations, to check what impact the numerics have on the results. Single-point statistics of the scalar field, including moments of

scalar gradients and PDFs of the scalar gradients and scalar dissipation rate are presented in §5.3. Two-point statistics of the scalar field are investigated in §5.4 both in Fourier space with spectra and in physical space with structure functions. Scalar gradient fluctuations are studied in §5.5. Finally, §5.6 summarizes the results from this chapter.

5.1 Description of DNS database

To systematically investigate the influence of the Schmidt number on the statistics of a passive scalar, a DNS database is generated at a fixed Reynolds number spanning a wide range of Schmidt numbers. While past studies at very high Schmidt number have been limited to low Reynolds numbers (Donzis *et al.*, 2010), the algorithms developed in Chapter IV and access to petascale computational resources on Blue Waters and Titan have enabled simulations of turbulent mixing at high Schmidt numbers with a sufficiently high Reynolds number to support a narrow inertial range (Yeung & Zhou, 1997). Specifically, isotropic turbulence at $R_\lambda \approx 140$ is maintained in all simulations with large-scale forcing (Donzis & Yeung, 2010), and scalars with Schmidt numbers ranging from 4 to 512 achieve stationarity under forcing by a mean scalar gradient in the x_1 direction. Aside from conducting simulations for a given set of flow parameters, it is also important to ensure that the numerical resolutions employed are sufficiently high for the statistics of interest (Donzis & Yeung, 2010). To check the robustness of the results, multiple numerical configurations are used for moderate Sc runs, which is permissible given their inexpensive computational cost relative to the high Sc runs.

Table 5.1 provides a list of the simulations carried out in this study. All runs use the code described in Chapter IV, which employs N_v^3 grid points in the FPS calculation of the velocity field, and N_θ^3 grid points for the scalar field computation using the CCD scheme. The resolution estimates for the velocity and scalar fields for all runs use the result that a $N_v = 256$ grid provides an approximate resolution

Table 5.1: List of simulations in DNS database for turbulent mixing at high Sc. The nominal resolution for $R_\lambda \approx 140$ isotropic turbulence on a $N_v = 256$ grid is 1.4, which is used to provide resolution estimates for all other velocity fields and scalar fields.

Run	Sc	N_v	$k_{\max,v}\eta$	N_θ	$k_{\max,\theta}\eta_B$	CFL
1	4	512	2.8	1024	2.8	0.8
2	4	512	2.8	2048	5.6	0.8
3	8	256	1.4	1024	2.0	0.8
4	8	512	2.8	1024	2.0	0.8
5	8	512	2.8	1024	2.0	0.4
6	8	512	2.8	2048	4.0	0.8
7	16	256	1.4	1024	1.4	0.8
8	16	256	1.4	2048	2.8	0.8
9	16	512	2.8	1024	1.4	0.8
10	16	512	2.8	2048	2.8	0.8
11	32	512	2.8	2048	2.0	0.8
12	32	512	2.8	2048	2.0	0.4
13	32	1024	5.6	4096	4.0	0.8
14	64	512	2.8	2048	1.4	0.8
15	128	512	2.8	4096	2.0	0.8
16	512	1024	5.6	8192	2.0	0.8

of $k_{\max,v}\eta = 1.4$ for $R_\lambda \approx 140$ forced isotropic turbulence. The resolution of the velocity field on different grids is therefore easily calculated, but to determine the scalar field resolution, the different grids and the increased resolution requirements of the Batchelor scale must be taken into account with the relation

$$k_{\max,\theta}\eta_B = k_{\max,v}\eta \left(\frac{N_\theta}{N_v}\right) \frac{1}{\sqrt{Sc}} \,. \tag{5.1}$$

The majority of the runs employ good small-scale resolution of the scalar field, e.g., $k_{\max,\theta}\eta_B = 2.0$, and some runs achieve very high resolution with $k_{\max,\theta}\eta_B = 4$ or greater. Time stepping in the DNS code uses the classical RK4 scheme for both the velocity and scalar fields, with the time step controlled by the Courant-Friedrichs-Lewy (CFL) criterion based on the finer grid spacing used for the scalar field. It was found through experimentation that with RK4 a CFL of 0.8 provides stable

and accurate time integration. Overall, RK4 greatly increases the computational throughput of the simulations compared to RK2, which is limited to CFL numbers close to 0.1 for high Schmidt numbers (determined from numerical tests). To check the influence of the CFL number, a few simulations (Runs 5 and 12) with a reduced CFL number of 0.4 have been included.

5.2 Numerical resolution effects

Before investigating the Schmidt number dependence of passive scalar statistics, the robustness of the results is assessed from a purely numerical perspective. In this section various statistics are compared from Runs 7–10, which all simulate a $Sc = 16$ scalar, but with differing grid resolutions. Here the focus is solely on the numerical results, with descriptions of the statistics chosen, and their physical significance, deferred to later sections of this chapter. As can be seen in table 5.1, Runs 7 and 8 use the same moderate resolution for the velocity field ($k_{\max,v}\eta = 1.4$), but differ in their resolution of the passive scalar, with Run 7 maintaining $k_{\max,\theta}\eta_B = 1.4$ while Run 8 has a higher resolution of $k_{\max,\theta}\eta_B = 2.8$. Runs 9 and 10 then increase the resolution of the velocity field by a factor of two, and use the same grids for the passive scalar as Runs 7 and 8, respectively. The configurations tested span a range of grid ratios N_θ/N_v from 2 to 8, which is useful to assess the impact that the tricubic interpolation scheme has on the results.

Figure 5.1 begins with the time evolution of certain scalar statistics from the four $Sc = 16$ simulations. In frame (a), the scalar variance and mean scalar dissipation rate are seen to remain indistinguishable for all simulations until $t/\tau \approx 3$. After this time, simulations with different values of N_v show different evolutions for these statistics, and close observation reveals that the influence of the scalar grid for a given velocity grid do not become apparent until $t/\tau \approx 7$. Frame (b) shows the time evolution of the scalar derivative skewness in the direction of the mean scalar

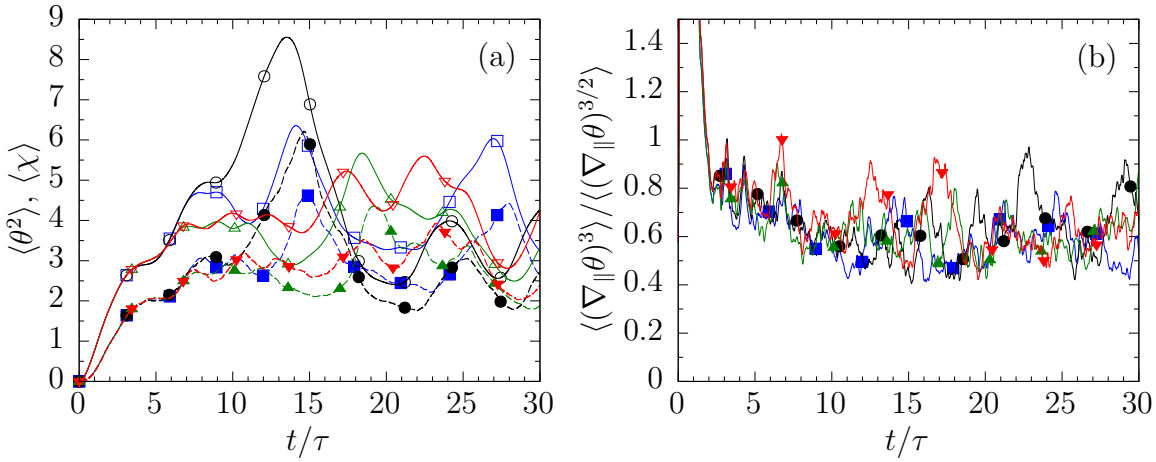

Figure 5.1: Development of (a) the scalar variance and dissipation rate and (b) the skewness of the fluctuating scalar gradient parallel to the mean scalar gradient for the $Sc = 16$ simulations. In (a), solid lines for scalar variance and dashed lines for scalar dissipation rate, with a single color and symbol shape corresponding to a given grid configuration: □ and ■ for Run 7, ○ and ● for Run 8, △ and ▲ for Run 9, and ▽ and ▼ for Run 10. In (b), colors and symbol shapes match those in (a) for the scalar dissipation rate.

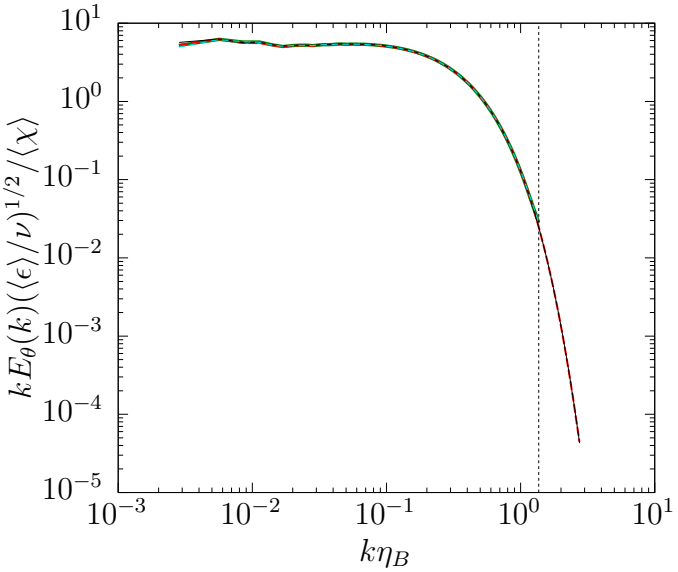

Figure 5.2: Time-averaged 3-D scalar spectrum under Batchelor scaling for the $Sc = 16$ simulations. Solid (thickest) green curve for Run 7, dashed red curve for Run 8, dashed cyan curve for Run 9, and solid (thinnest) black curve for Run 10. Cutoff wavenumber for Runs 7 and 9 (those with $N_\theta = 1024$) marked with vertical line.

gradient (a small-scale statistic). After an initial development period, the skewness also hovers around a nominal value in the stationary state. Close observation reveals that the runs with higher resolution scalar grids (Run 8 in black, and Run 10 in red) experience larger fluctuations in the skewness compared to the lower resolution cases, showing the expected result that small-scale statistics are sensitive to grid resolution. One should not, of course, expect the instantaneous values for the statistics shown in figure 5.1 to remain the same for all simulations, given the chaotic nature of turbulence (Pope, 2000); however, it is reasonable to expect that as long as the resolutions employed are high enough for a quantity of interest, statistical averages in the stationary state should be in good agreement across multiple configurations. To illustrate this, figure 5.2 presents the time-averaged 3-D scalar spectrum for Runs 7–10 under Batchelor scaling (the details of which will be discussed later). Except for very slight variations at the largest scales (lowest wavenumbers), all spectra agree excellently. For the higher resolution grids (Runs 8 and 10, with $N_\theta = 2048$), whose resolution extends beyond that of the coarser grids, the spectra agree perfectly far into the dissipation range. The excellent agreement of the scalar spectrum across many numerical configurations (N_θ/N_v ranging from 2 to 8) suggests that the numerical resolutions and scheme employed are well suited to investigate the scaling of the scalar spectrum at high Sc.

As alluded to earlier when discussing the scalar gradient skewness, small-scale statistics in DNS are sensitive to the numerical resolution employed. In particular, intermittent quantities such as the scalar dissipation rate (Sreenivasan & Antonia, 1997) require high resolution to calculate and sample properly (Donzis & Yeung, 2010). To investigate the impact of the numerical configurations on the sampling of small-scale statistics, figure 5.3 shows for all $Sc = 16$ simulations the time-averaged PDFs of normalized scalar gradients parallel and perpendicular to the mean scalar gradient in frames (a) and (b), respectively, and the PDF of the normalized scalar dis-

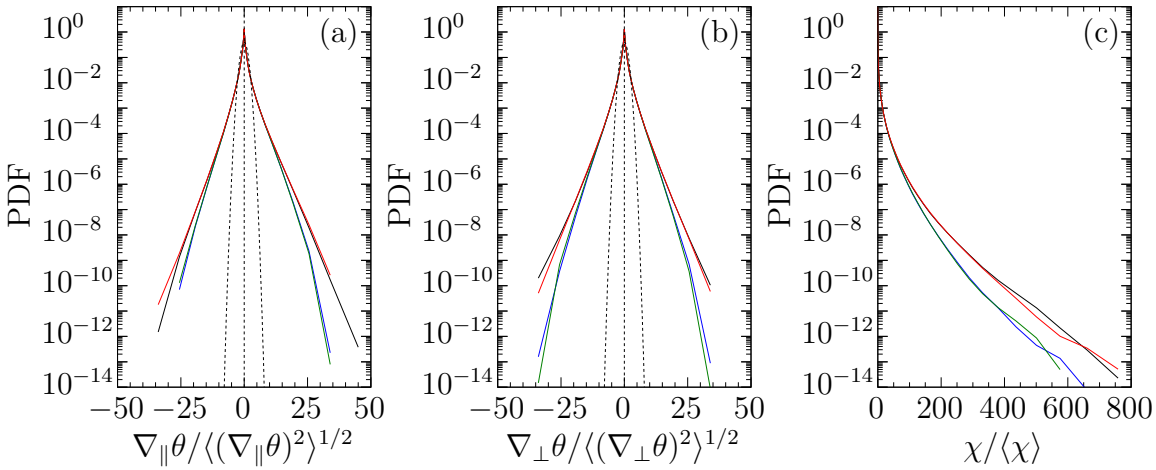

Figure 5.3: Time-averaged PDFs of (a) normalized fluctuating scalar gradients in the direction of the mean scalar gradient, (b) average of the normalized fluctuating scalar gradients in directions transverse to the mean scalar gradient, and (c) normalized scalar dissipation rate. Data for the $Sc = 16$ simulations, with blue for Run 7, black for Run 8, green for Run 9, and red for Run 10. Upper curves in each frame contain results from the higher resolution scalar grids (Runs 8 and 10).

sipation rate in frame (c). While the PDFs are sensitive to the scalar grid resolution (expected), they exhibit no discernible dependence on the velocity field resolution. This provides further support that the velocity field resolutions used in the current work are more than adequate for the statistics of interest. Because of the apparent sensitivity of such statistics to the scalar grid resolution, in later sections when examining the Schmidt number dependence of various quantities, results will mainly be compared for fixed small-scale resolutions, i.e., fixed values of $k_{\max,\theta}\eta_B$.

5.3 Single-point statistics

To begin analyzing the Schmidt number dependence of turbulent mixing, table 5.2 presents some single-point statistics obtained from all runs. In each sub-table of table 5.2, the second block contains the time-averaged turbulence kinetic energy and energy dissipation rate, along with the amount of time the simulation was run in the stationary state T, normalized by the large-eddy turnover time $\tau = \ell/u'$. Most simulations using $N_\theta = 2048$ or fewer grid points for the scalar field were run for much

Table 5.2: Simulation parameters and time-averaged statistics, split over two tables for (upper) Runs 1–8 and (lower) Runs 9–16. In each table, numerical configuration in the first block. Second block includes turbulence kinetic energy and energy dissipation rate, and the amount of time the simulation was run in the stationary state, with $\tau = \ell/u' \approx 0.8$. Third block gives time-averaged statistics for the scalar fields, including the mechanical-to-scalar timescale ratio r_θ and the velocity-scalar correlation coefficient $\rho_{u\theta}$. Last block for scalar derivative statistics, with the parallel subscript indicating the derivative is in the direction of the mean scalar gradient, the perpendicular subscript indicating an average over statistics in the transverse directions, and σ for standard deviation and μ_n for normalized central moment of order n.

Run	1	2	3	4	5	6	7	8
N_v	512	512	256	512	512	512	256	256
N_θ	1024	2048	1024	1024	1024	2048	1024	2048
CFL	0.8	0.8	0.8	0.8	0.4	0.8	0.8	0.8
Sc	4	4	8	8	8	8	16	16
K	3.17	3.21	3.18	3.19	3.17	3.15	3.18	3.17
$\langle\epsilon\rangle$	1.32	1.36	1.34	1.34	1.32	1.30	1.34	1.32
T/τ	105	26.6	95	85.6	66.3	45.3	97.9	43.9
$\langle\theta^2\rangle$	3.62	3.69	4.03	3.66	3.36	3.67	4.03	4.15
P	2.91	2.97	2.99	2.78	2.55	2.72	2.81	2.90
$\langle\chi\rangle$	2.91	2.98	3.01	2.80	2.57	2.76	2.83	2.90
r_θ	1.95	1.92	1.80	1.84	1.88	1.85	1.70	1.72
$\rho_{u\theta}$	-0.52	-0.52	-0.50	-0.50	-0.49	-0.49	-0.48	-0.48
$\sigma_\parallel^2/\sigma_\perp^2$	1.03	1.04	1.02	1.03	1.03	1.02	1.02	1.01
$\mu_{3\parallel}$	1.09	1.08	0.844	0.843	0.860	0.828	0.592	0.634
$\mu_{4\parallel}$	19.1	19.1	18.8	18.4	18.2	18.6	16.9	18.5
$\mu_{4\perp}$	17.3	17.2	17.3	16.9	16.8	17.4	16.0	17.4

Run	9	10	11	12	13	14	15	16
N_v	512	512	512	512	1024	512	512	1024
N_θ	1024	2048	2048	2048	4096	2048	4096	8192
CFL	0.8	0.8	0.8	0.4	0.8	0.8	0.8	0.8
Sc	16	16	32	32	32	64	128	512
K	3.20	3.15	3.16	3.14	3.18	3.18	3.13	3.21
$\langle\epsilon\rangle$	1.36	1.30	1.31	1.29	1.34	1.33	1.28	1.35
T/τ	84.3	56.8	43.9	18.6	10.6	53.4	22.6	7.95
$\langle\theta^2\rangle$	3.66	4.65	4.61	3.83	4.20	5.34	5.90	5.45
P	2.67	3.18	3.00	2.50	2.71	3.21	3.23	2.81
$\langle\chi\rangle$	2.62	3.14	2.97	2.46	2.80	3.20	3.32	2.75
r_θ	1.71	1.66	1.58	1.58	1.59	1.45	1.40	1.21
$\rho_{u\theta}$	-0.48	-0.49	-0.47	-0.45	-0.45	-0.46	-0.45	-0.42
$\sigma_\parallel^2/\sigma_\perp^2$	1.02	1.01	1.01	1.02	1.02	1.00	0.99	1.01
$\mu_{3\parallel}$	0.602	0.638	0.470	0.483	0.492	0.310	0.244	0.134
$\mu_{4\parallel}$	16.9	18.8	18.2	18.1	18.9	16.6	17.9	17.5
$\mu_{4\perp}$	16.0	17.6	17.3	17.3	18.0	16.1	17.5	17.2

longer than is typically required for statistical convergence (e.g. see Donzis & Yeung (2010), where simulations are run for $T/\tau \approx 10$), but are inexpensive enough that longer run times are permissible. The third block reports the time-average of statistics such as the scalar variance and production rate. While dimensional quantities such as the scalar variance show large fluctuations in the stationary state (e.g., see figure 5.1), non-dimensional parameters like the mechanical-to-scalar timescale ratio $r_\theta = (K/\langle\epsilon\rangle)/(\langle\theta^2\rangle/\langle\chi\rangle)$ show significantly reduced variation, and better agreement after averaging for a given set of flow parameters, i.e., at a fixed Schmidt number. Consistent with previous studies (Yeung *et al.*, 2002), both r_θ and the velocity-scalar correlation coefficient $\rho_{u\theta}$ decrease with increasing Sc, indicating the importance of incorporating Schmidt number dependencies into turbulence models.

The last block of the sub-tables reports statistics of scalar gradients parallel and perpendicular to the mean scalar gradient. Such derivative statistics characterize the small scales, and are useful when verifying the isotropy (or lack thereof) in the small scales of the scalar field. Beginning with Kolmogorov (1941), a driving force in turbulence theory has been the hypothesis that at high Reynolds numbers, the smallest scales in the flow are locally isotropic, and are therefore universal. There is a wide body of research supporting Kolmogorov's original hypotheses at the second and third order for the velocity field (Sreenivasan, 1995; Yeung & Zhou, 1997), but extensions of the hypotheses to passive scalars (Obukhov, 1949; Corrsin, 1951) have suffered from results contradicting the theory (Sreenivasan, 1991). If it were true, local isotropy for the passive scalar field would imply that even-order moments of scalar gradients parallel and perpendicular to the mean gradient would be equal, and that odd-order moments in the direction of the mean gradient would vanish. Table 5.2 shows that derivative variance ratios are mostly above one, the skewness of the gradients parallel to the mean gradient is non-zero, and that the flatness factors of parallel and perpendicular gradients are not equal. While statistics are not in support

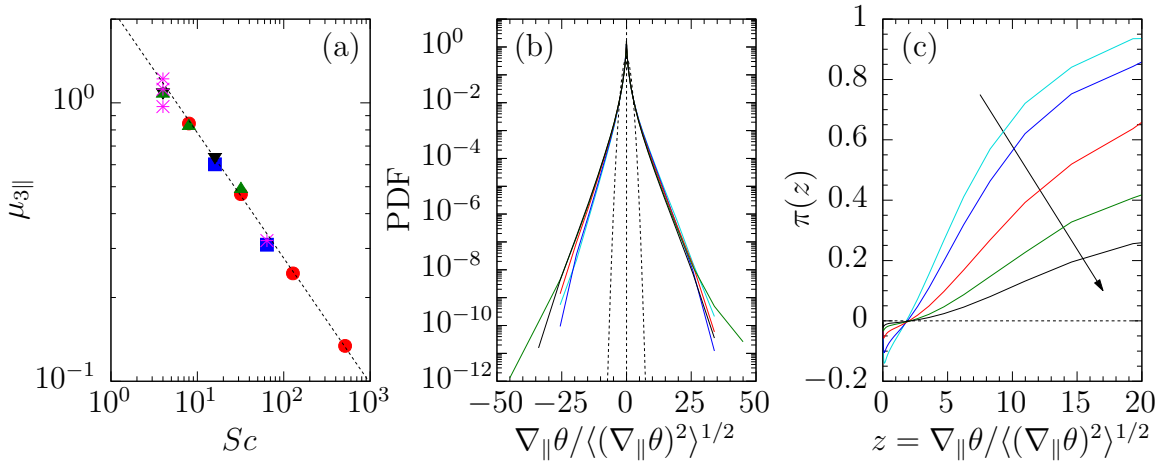

Figure 5.4: Schmidt number dependence of (a) the skewness of the parallel scalar gradients, (b) the PDF of the parallel scalar gradients, and (c) an asymmetry indicator for the PDF. In (a), non-star symbols for different grid resolutions in the new DNS database: ■ for $k_{\max,\theta}\eta_B = 1.4$ (Runs 9 and 14), ● for $k_{\max,\theta}\eta_B = 2.0$ (Runs 4, 11, 15 and 16), ▼ for $k_{\max,\theta}\eta_B = 2.8$ (Runs 1 and 10), and ▲ for high-resolution cases (Runs 2, 6, and 13). Magenta stars (✳) for $R_\lambda \approx 140$ data (at multiple resolutions) reported by Donzis & Yeung (2010). In left frame, sloped line proportional to $Sc^{-0.45}$. In (b) and (c), cyan curve for $Sc = 4$ (Run 2), blue curve for $Sc = 8$ (Run 5), red curve for $Sc = 32$ (Run 12), green curve for $Sc = 128$ (Run 15), and black curve for $Sc = 512$ (Run 16). In (b), reference Gaussian PDF shown with curved dotted line.

of isotropy, there is a trend towards local isotropy with increasing Schmidt number, as has been previously reported at lower Reynolds numbers (Yeung *et al.*, 2002).

It is useful to determine the rate by which the scalar field becomes more isotropic as the Schmidt number is increased. For this consider the skewness of the scalar gradients in the direction of the mean scalar gradient, which would be zero if local isotropy were to hold. Figure 5.4 shows the Schmidt number dependence of the skewness in frame (a). Results from Donzis & Yeung (2010) are also included, where Fourier pseudo-spectral methods were used for both the velocity and scalar fields for Schmidt numbers up to 64 in $R_\lambda \approx 140$ forced isotropic turbulence. Both datasets are suggestive of an approach toward local isotropy with a power-law dependence on Sc of the form $Sc^{-\alpha}$ with $\alpha \approx 0.45$. Mathematically, the skewness is related to the third moment of the scalar gradient PDF, which is shown in frame (b) for multiple simulations having nominally the same scalar field resolution. The positive

skewness is a result of the asymmetry of the PDF, which shows large positive scalar gradients being more likely than large negative gradients. The asymmetry in the PDF is perhaps not as evident as it is for lower Schmidt numbers (e.g., see data in Yeung *et al.* (2002)), so frame (c) includes an asymmetry indicator of the form (Schumacher & Sreenivasan, 2003)

$$\pi(z) = \frac{p(z) - p(-z)}{p(z) + p(-z)} \ , \tag{5.2}$$

where $p(z)$ denotes the value of the PDF for a particular value of $z = \nabla_{\|}\theta / \left\langle (\nabla_{\|}\theta)^2 \right\rangle^{1/2}$. The indicator function clearly shows that as Sc increases, the asymmetry in tails of the PDF weakens, suggesting that passive scalars become more isotropic at higher Sc.

The anisotropy in the small scales of the scalar field is the direct result of their apparent coupling to the large-scale, anisotropic mean scalar field. This coupling between the large and small scales arises when turbulent advection in the direction of the imposed mean scalar gradient brings together low and high values of the scalar (i.e., warm and cool fluid, in the case of temperature fluctuations), which results in the formation of steep scalar gradients in the direction of the mean gradient (Warhaft, 2000). The scalar fluctuations have a characteristic ramp-cliff structure (Sreenivasan *et al.*, 1979; Holzer & Siggia, 1994), where ramps correspond to regions of relatively constant values of the total scalar, and cliffs correspond to the steep gradients. To visualize the scalar structure, and to compare lower Schmidt number scalars with higher Schmidt number scalars, figure 5.5 shows surface plots of the total instantaneous scalar value

$$\Theta = \frac{\partial \langle \Theta \rangle}{\partial x_i} x_i + \theta \tag{5.3}$$

for $Sc = 4$ and $Sc = 32$. Scientific visualization in the manner presented in figure 5.5 is not very precise, but although the slices presented are taken randomly from the available data, they possess many of the expected features for passive scalars given the above discussions. The top image, corresponding to $Sc = 4$, is perhaps easier to look

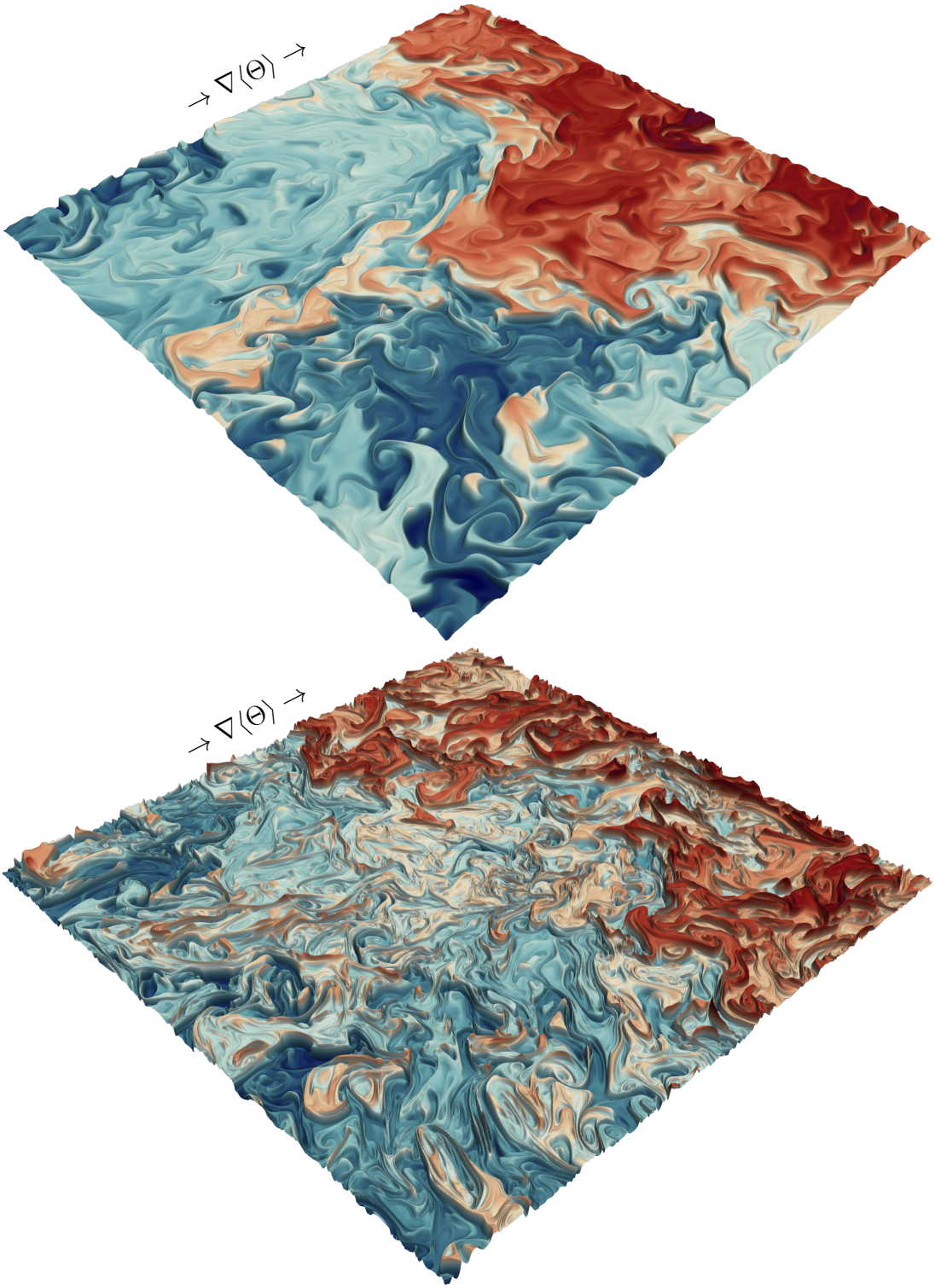

Figure 5.5: Surface plot of total scalar (mean plus fluctuation) for arbitrary slices taken from (top) a $Sc = 4$ simulation (Run 2) and (bottom) a $Sc = 32$ simulation (Run 11). Low (cooler) values for the scalar shown in blue, and higher (warmer) values shown in red. Direction of the mean gradient shown next to each rendering.

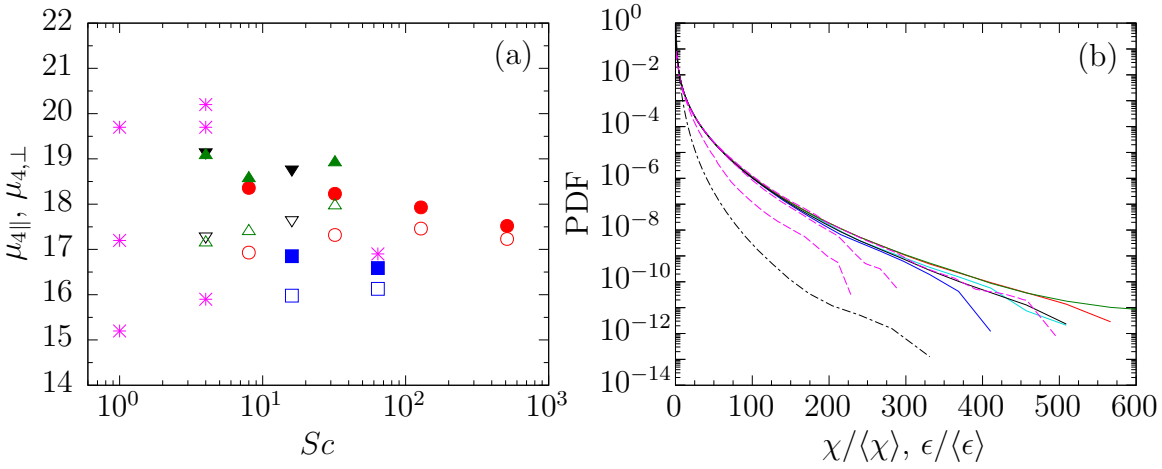

Figure 5.6: Plots of statistics focusing on scalar field intermittency. In (a), flatness factors of scalar gradients (closed symbols) parallel and (open symbols) perpendicular to the mean scalar gradient, along with data for $\mu_{4\|}$ from Donzis & Yeung (2010). Symbols in (a) are the same as those in frame (a) of figure 5.4 for the parallel gradients, and matching open symbols are for perpendicular gradients. In (b), PDFs of (solid curves) normalized scalar dissipation rate and (black dashed-dotted curve) normalized energy dissipation rate. For scalar dissipation rate, cyan curve for $Sc = 4$ (Run 2), blue curve for $Sc = 8$ (Run 5), red curve for $Sc = 32$ (Run 12), green curve for $Sc = 128$ (Run 15), and black curve for $Sc = 512$ (Run 16). Also in (b) are PDFs of normalized scalar dissipation rate from Donzis & Yeung (2010) in the form of dashed magenta curves, with increasing tails for $Sc = 1/8$, $Sc = 1$, and $Sc = 4$.

at, and in it there are clear signatures of the ramp-cliff structure in the direction of the mean scalar gradient. There are vast regions where the total scalar is relatively well-mixed (i.e., taking nearly the same value), with fairly steep gradients in the direction of the mean gradient separating the regions. The lower image, corresponding to $Sc = 32$ is more challenging to understand because the reduced molecular diffusivity gives rise to a broader range of scales in the scalar field. Scalar gradients must take even larger values in order to maintain the scalar dissipation rate (Donzis *et al.*, 2005) after the diffusivity is reduced, which is visually evident in the rapid changes in the scalar over short length scales.

In addition to being anisotropic at the small scales, another characteristic of passive scalars is that they are generally more intermittent than the velocity field (Sreenivasan & Antonia, 1997). Small-scale intermittency in the passive scalar field can be

examined through statistics such as the flatness factors of the scalar gradients and the scalar dissipation rate (not simply its mean value). It has previously been found that for a fixed Reynolds number, intermittency saturates with increasing Schmidt number (Yeung *et al.*, 2002). Some previous work even reached $Sc = \mathcal{O}(1000)$ (Yeung *et al.*, 2004), but at severely restricted Reynolds numbers (e.g., $R_\lambda \approx 8$). The new DNS data is used to further investigate the saturation of intermittency in the scalar field, beginning in figure 5.6 with frame (a) for the Schmidt number dependence of the flatness factors of the scalar gradients parallel and perpendicular to the mean gradient (using data from table 5.2). While the data might be suggestive of saturation, it appears that higher resolution is needed to make a more definitive conclusion. Focusing on the two simulations at $Sc = 32$ (which achieved $k_{\max,\theta}\eta_B = 2$ and 4, respectively) there is an approximate 5% increase in the flatness factors at the higher resolution. The trend that the flatness factors apparently decrease at higher Sc is likely not robust, and the picture might change with higher resolution datasets. It would be worthwhile to invest in higher resolution simulations at $Sc = 64$ and $Sc = 128$ (the latter requiring 8192^3 for a higher resolution run) to clear this up.

Continuing, the PDF of the normalized scalar dissipation rate is presented in frame (b) of figure 5.6. Data from the new DNS database is shown for configurations with nominally the same small-scale resolution, alongside high-resolution results from Donzis & Yeung (2010) for comparison purposes (their simulations being for $Sc = 1/8$, 1 and 4). Also included is the PDF of the normalized energy dissipation rate. It is clear that the scalar field is much more intermittent than the velocity field, e.g., scalar dissipation events that are 200 times the mean value are approximately two orders of magnitude more likely for the $Sc = 1/8$ scalar compared to the same events for the energy dissipation rate. Intermittency indeed increases with Sc, but beyond $Sc = 1$ changes in the PDF are minimal, and the larger variations in the extreme tails of the PDFs are likely due to limited sampling. The robust shape of the

PDFs with increasing Schmidt number is also suggestive of the potential saturation of intermittency in the scalar field.

5.4 Two-point statistics

Moving beyond single-point statistics, there are many two-point statistics containing information on the structure of the scalar field that are of fundamental interest. Two-point statistics are defined in physical space, the simplest example of which is perhaps the two-point single-time correlation of the scalar fluctuations

$$Q_\theta(\mathbf{r}) = \langle \theta(\mathbf{x})\theta(\mathbf{x}+\mathbf{r}) \rangle \ , \tag{5.4}$$

where $\mathbf{r}$ is the separation vector, and in homogeneous turbulence the statistic is independent of $\mathbf{x}$. In homogeneous turbulence it is common to take the Fourier transform of statistics like (5.4) and study their corresponding spectra. The analysis in this section begins with the spectral view, and a focus on the 3-D scalar spectrum $E_\theta(k)$, which is related to the two-point correlation, and whose integral gives the scalar variance:

$$\langle \theta^2 \rangle = \int_0^\infty E_\theta(k)\, dk \ . \tag{5.5}$$

The question that arises is: what form (shape) does the scalar spectrum take, and what is its dependence on the Reynolds number and Schmidt number?

The first extensions of Kolmogorov's theory for the velocity field to passive scalars were made by Obukhov (1949) and Corrsin (1951) and made quantitative predictions for the shape of the scalar spectrum. Dimensional reasoning in the so-called inertial-convective range — a range of scales which are sufficiently small compared to the large-scale motions, and sufficiently large such that dissipation of both the kinetic

energy and scalar variance is negligible — provides the result

$$E_\theta(k) = C_{OC}\langle\chi\rangle\langle\epsilon\rangle^{-1/3}k^{-5/3} \ , \tag{5.6}$$

where C_{OC} is the Obukhov-Corrsin constant for the 3-D spectrum. Obukhov-Corrsin scaling has been observed in a number of high Reynolds number experimental (Sreenivasan, 1996) and numerical (Yeung & Donzis, 2005) studies, with the Obukhov-Corrsin constant taking a value of approximately 0.4 for the 1-D spectra, corresponding to 0.67 for the 3-D spectra assuming the spectra can be related through local isotropy relations. While the current simulations are at a sufficiently high Reynolds number for a narrow inertial range to develop in the velocity field ($R_\lambda \approx 140$), they fall short of what would required for a clearly developed $k^{-5/3}$ scaling range (e.g., see the wide scaling range attained in the $R_\lambda \approx 700$ simulations of Yeung & Donzis (2005)). Because of this, attention is focused on the different scaling behavior that develops in the scalar spectrum as the Schmidt number is increased.

Batchelor (1959) extended the theory of passive scalars to include scalars with low molecular diffusivity. As mentioned previously, when the molecular diffusivity is low (corresponding to a high Schmidt number), the range of scales in the scalar field is broader than that of the velocity field. Specifically, the so-called viscous-convective range develops, which is a range of scales smaller than the Kolmogorov scale of the velocity field, but larger than the Batchelor scale of the scalar field. Assuming that the relevant timescale for mixing in the viscous-convective range is the Kolmogorov timescale $\tau_\eta = (\nu/\langle\epsilon\rangle)^{1/2}$, dimensional reasoning gives the Batchelor spectrum

$$E_\theta(k) = C_B\langle\chi\rangle(\nu/\langle\epsilon\rangle)^{1/2}k^{-1} \ , \tag{5.7}$$

where C_B is the Batchelor constant. While the spectrum given by (5.7) is for the viscous-convective range of wavenumbers ($1/\eta \ll k \ll 1/\eta_B$), Batchelor's theory

predicts more rapid decay in the viscous-diffusive range ($k \gg 1/\eta_B$) with the form

$$E_\theta(k) = C_B\langle\chi\rangle(\nu/\langle\epsilon\rangle)^{1/2}k^{-1}\exp(-C_B(k\eta_B)^2) \ . \tag{5.8}$$

Batchelor's theory does not take into account potential temporal variations in the strain rates affecting the scalar field; later Kraichnan (1974) developed a theory assuming very rapid fluctuations in the strain rates, and arrived at the result

$$E_\theta(k) = C_B\langle\chi\rangle(\nu/\langle\epsilon\rangle)^{1/2}k^{-1}(1+(6C_B)^{1/2}k\eta_B)\exp(-(6C_B)^{1/2}k\eta_B) \ , \tag{5.9}$$

which also predicts k^{-1} scaling in the viscous-convective range, but less rapid decay in the far diffusive range compared to Batchelor's result. It is important to note that k^{-1} scaling in the viscous-convective range is very robust, as it is predicted by both theories. It is therefore unfortunate that experimental support for such k^{-1} scaling is elusive, or sometimes negative (Miller & Dimotakis, 1996; Warhaft, 2000). Simulations have, on the other hand, for some time shown increasing support for Batchelor scaling at high Schmidt number (Donzis *et al.*, 2010).

Figure 5.7 presents the time-averaged spectra obtained from the DNS database. In frame (a), the raw 3-D spectra show an increasing trend towards Batchelor scaling with increasing Schmidt number. To investigate this more rigorously, frame (b) shows the same spectra under Batchelor scaling, and includes Kraichnan's result for the spectrum assuming a Batchelor constant $C_B = 5.7$ (Gotoh *et al.*, 2014). If Batchelor scaling were clearly observed, a plateau at the value of the Batchelor constant would emerge in the spectrum at higher Schmidt numbers. For the highest Schmidt number of 512, there might be an emergence of a brief scaling range, but higher Schmidt numbers yet are required to draw definitive conclusions. It seems clear, however, that if such a scaling range were to emerge at higher Schmidt numbers, the resulting Batchelor constant would be higher than that observed by Gotoh *et al.*

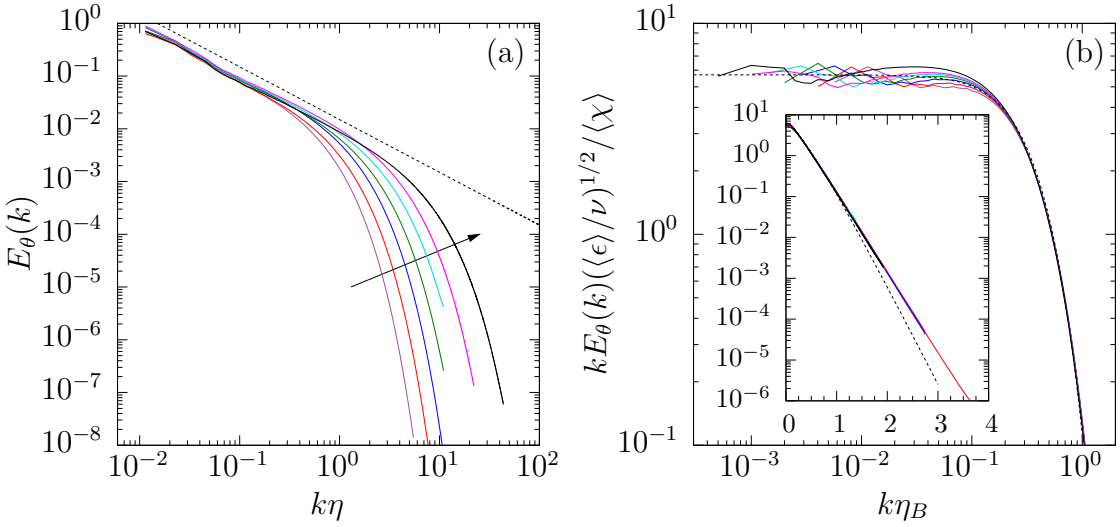

Figure 5.7: Time-averaged 3-D scalar spectrum obtained from DNS for multiple Schmidt numbers. In (a), raw 3-D spectra with arrow in the direction of increasing Sc; sloped dotted line proportional to k^{-1}. In (b), spectra presented under Batchelor scaling, with dotted black line using the Kraichnan (1974) result with a Batchelor constant of 5.7 (Gotoh *et al.*, 2014). The inset in (b) presents the same spectra in linear-log coordinates. In all plots, different colors for different Schmidt numbers: purple for $Sc = 4$, red for $Sc = 8$, blue for $Sc = 16$, green for $Sc = 32$, cyan for $Sc = 64$, magenta for $Sc = 128$, and black for $Sc = 512$.

(2014), who simulated scalars with Schmidt numbers up to 1000 in lower Reynolds number turbulence ($R_\lambda \approx 40$). The inset figure in frame (b) shows the same spectra in linear-log coordinates, which is useful to assess the functional form of the spectrum in the far diffusive range. Clearly, the DNS data show an exponential falloff at high wavenumbers, which is in agreement with Kraichnan's prediction for the scalar spectrum. Given that the somewhat high resolution employed for the $Sc = 512$ simulation might not be required to observe Batchelor scaling (e.g., see the agreement between the spectra for simulations at modest and high resolutions in figure 5.2), the simulation could potentially be continued with a reduced molecular diffusivity (perhaps to achieve $Sc = 1024$) to further explore Batchelor scaling.

Returning to the physical space, the exact relation derived by Yaglom (1949) for the mixed velocity-scalar structure function in the inertial-convective range is a

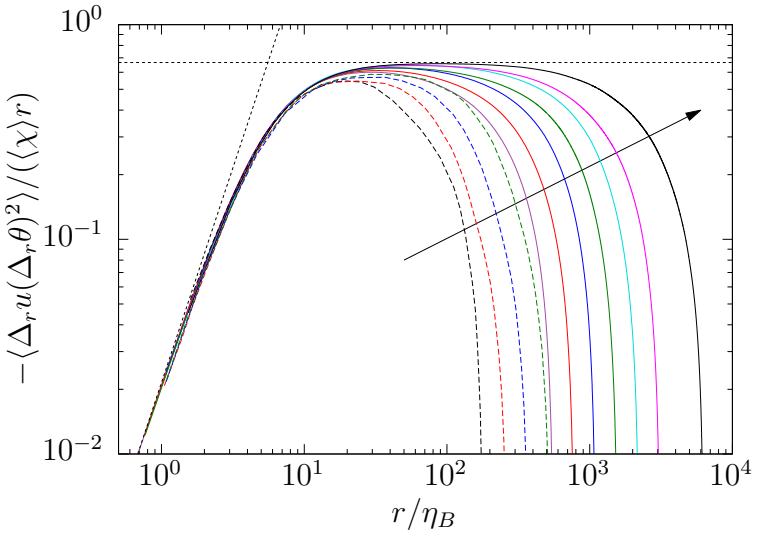

Figure 5.8: Mixed velocity-scalar structure function with scaling according to Yaglom's relation. Solid curves are from the new DNS database, with the same colors corresponding to the same Schmidt numbers detailed in the caption of figure 5.7. Dashed curves from figure 10 of Yeung *et al.* (2002), for Schmidt numbers 8, 16, 32, and 64 in $R_\lambda \approx 38$ forced isotropic turbulence. Horizontal dashed line at 2/3, and sloped dotted line proportional to r^2. Arrow drawn in direction of increasing Schmidt number for both (independent) datasets.

fundamental scaling result for the passive scalar field. For two points separated by a distance r, Yaglom's relation reads

$$\langle\Delta_r u(\Delta_r\theta)^2\rangle = -\frac{2}{3}\langle\chi\rangle r \ , \tag{5.10}$$

where $\Delta_r u = u(x+r) - u(x)$ is the velocity increment, and similarly $\Delta_r\theta$ is the scalar increment. This relation also holds in the viscous-convective range for high Schmidt number scalars (Yeung *et al.*, 2002; Gotoh & Yeung, 2013). While (5.10) shows directional dependencies in numerical simulations using mean-gradient forcing (Yeung *et al.*, 2002), theoretical arguments suggest that with direction averaging (strictly speaking, spherical averages) Yaglom's result still holds (Gotoh & Yeung, 2013). The results have therefore been directionally averaged, as is customary in DNS (Yeung & Donzis, 2005), to report the mixed velocity-scalar structure function in figure 5.8 with scaling according to Yaglom's relation. Figure 5.8 also includes data

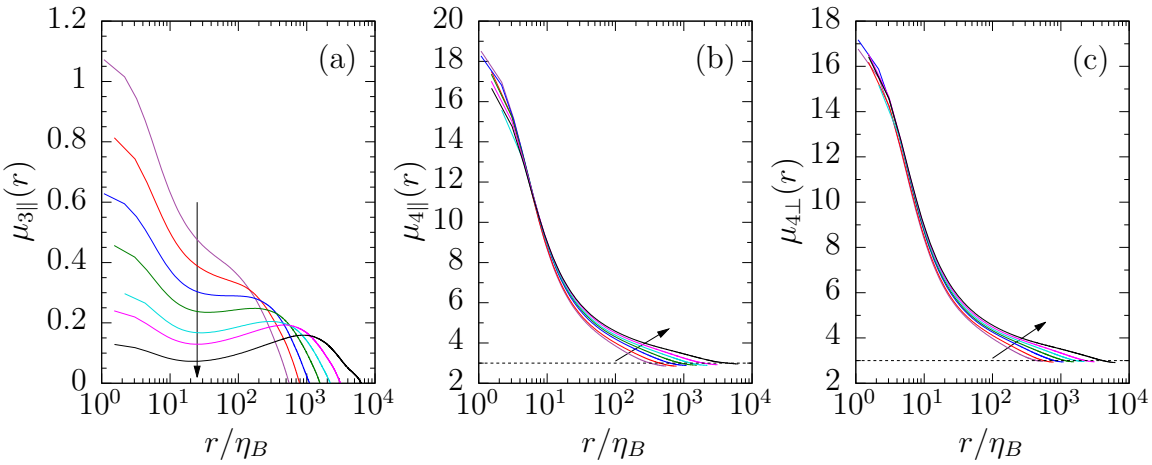

Figure 5.9: Structure function (a) skewness with separations in the mean gradient direction, (b) flatness with separations in the mean gradient direction, and (c) flatness with separations perpendicular to the mean gradient direction. Curve colors match description in caption of figure 5.7, with arrows in the directions of increasing Sc.

for $Sc = 8$, 16, 32, and 64 scalars in $R_\lambda \approx 38$ isotropic turbulence from Yeung *et al.* (2002). For small separations, a Taylor series expansion of the increments suggests that the normalized quantity should grow in proportion to r^2. The sloped dashed line is formed using a value of -0.5 for the mixed velocity-scalar skewness (Yeung *et al.*, 2002), and all data collapses very well (as expected) for small separations. As the Schmidt number increases, a clear plateau emerges at intermediate separations, with a peak value very close to Yaglom's theoretical result.

Structure functions of the scalar field alone are useful to assess isotropy and intermittency as a function of scale size. Consider the skewness structure function

$$\mu_{3\parallel}(r) = \frac{\left\langle \left(\Delta_\parallel \theta(r)\right)^3 \right\rangle}{\left\langle \left(\Delta_\parallel \theta(r)\right)^2 \right\rangle^{3/2}} , \tag{5.11}$$

where this time the separations $\Delta_\parallel \theta(r)$ are taken over a distance r in the direction of the imposed mean gradient. If the scalar field were isotropic at all scales, this quantity would be zero for all separations. Figure 5.9 shows the skewness structure function in frame (a), which is positive over all separations, similar to previous experimental

(Mydlarski & Warhaft, 1998) and numerical (Yeung *et al.*, 2002) work. Consistent with the approach toward local isotropy reported for the derivative skewness, there is a general approach toward isotropy over a wide range of scales as the Schmidt number is increased. Unlike previous work at lower Reynolds numbers and Schmidt numbers (Yeung *et al.*, 2002), the skewness structure function develops a local minimum in the range of 20–40 Batchelor scales. Further work is required to understand why the skewness then increases for larger separations; perhaps the behavior can be tied to the large-scale ramp-cliff structures that are present in the scalar field. Shown in frames (b) and (c) of figure 5.9 are the structure function flatness factors defined in directions parallel and perpendicular to the mean scalar gradient, respectively. As discussed by Warhaft (2000), classical Kolmogorov-Obukhov-Corrsin scaling predicts a constant value for these flatness factors as a function of r, which is clearly violated (see Yeung *et al.* (2002) for similar results). The Schmidt number dependence of the flatness factors is very weak, which again supports the hypothesis that scalar field intermittency saturates as the Schmidt number is increased.

5.5 Statistics of scalar gradients

Ultimately, passive scalar mixing is controlled by molecular diffusion at the finest scales present in the scalar field. Remarkably, the mean scalar dissipation rate

$$\langle \chi \rangle = 2D \left\langle \frac{\partial \theta}{\partial x_i} \frac{\partial \theta}{\partial x_i} \right\rangle \tag{5.12}$$

actually becomes independent of the molecular diffusivity at sufficiently high Reynolds numbers and Schmidt numbers (Donzis *et al.*, 2005; Shraiman & Siggia, 2000). When considering two scalars with different Schmidt numbers, it is clear that in order to maintain the same mean scalar dissipation rate the statistics of their respective scalar gradients must differ. It is therefore important to study the statistics of scalar gradi-

ents, and to understand the physical mechanisms behind their production and subsequent amplification to the required levels dictated by the scalar dissipation rate.

The starting point for the statistical analysis of scalar gradients is the evolution equation of the scalar gradient itself, obtained by taking the gradient of (4.4) to give

$$\frac{\partial \theta_i}{\partial t} + u_j \frac{\partial \theta_i}{\partial x_j} = -\theta_j s_{ji} - \langle \Theta_j \rangle s_{ji} - \frac{1}{2}\epsilon_{ijk}\theta_j \omega_k - \frac{1}{2}\epsilon_{ijk}\langle \Theta_j \rangle \omega_k + D\frac{\partial^2 \theta_i}{\partial x_j \partial x_j} \ , \qquad (5.13)$$

where $\theta_i \equiv \partial\theta/\partial x_i$, $\langle \Theta_i \rangle \equiv \partial\langle\Theta\rangle/\partial x_i$, s_{ij} is the fluctuating strain rate, ω_i is the vorticity, and ϵ_{ijk} is the permutation symbol. [For a detailed summary of many evolution equations relevant to the passive scalar gradients, please consult Brethouwer *et al.* (2003).] The equation for the scalar gradients variances is readily derived from (5.13), and is given by

$$\begin{aligned} \frac{\partial \langle \theta_i \theta_i \rangle}{\partial t} + \frac{\partial \langle u_j \theta_i \theta_i \rangle}{\partial x_j} &= -2\langle \theta_i \theta_j s_{ji} \rangle - 2\langle \Theta_j \rangle \langle \theta_i s_{ji} \rangle - \\ & \epsilon_{ijk}\langle \theta_i \theta_j \omega_k \rangle - \epsilon_{ijk}\langle \Theta_j \rangle \langle \theta_i \omega_k \rangle - 2D \left\langle \frac{\partial \theta_i}{\partial x_j} \frac{\partial \theta_i}{\partial x_j} \right\rangle \ , \end{aligned} \qquad (5.14)$$

where the second term on the left-hand side is zero in homogeneous turbulence. Of prime interest in (5.14) is the first term on the right-hand side, representing the nonlinear amplification of the scalar gradients by the fluctuating strain rate. This term has received considerable attention in the literature (Vedula *et al.*, 2001; Brethouwer *et al.*, 2003), and is most readily studied in the principal axes of the strain rate tensor. In this coordinate system, the strain rate tensor takes a diagonal form, and the strain is either extensional or compressive in each direction. The principal strain rates are ordered $\alpha \geq \beta \geq \gamma$, where $\alpha + \beta + \gamma = 0$ because of incompressibility, and $\alpha \geq 0$ and $\gamma \leq 0$. The intermediate eigenvalue is known to be mostly positive, and its PDF obtained from the current simulations (not shown) matches those previously reported well (Vedula *et al.*, 2001).

Of primary interest is the alignment of the scalar gradients with the principal

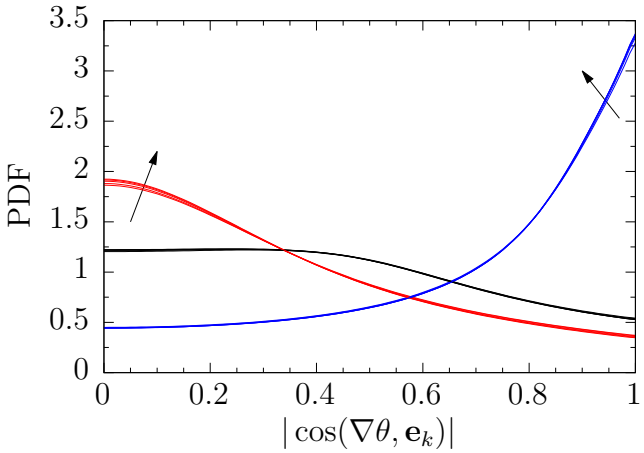

Figure 5.10: PDFs of direction cosines between scalar gradients and principal axes of the strain rate tensor. Starting from the left side of the figure, lower (blue) curves for most compressive direction ($\mathbf{e}_\gamma$), middle (black) curves for most extensional direction ($\mathbf{e}_\alpha$), and upper (red) curves intermediate direction ($\mathbf{e}_\beta$). Schmidt number increasing in the directions of the arrows, from $Sc = 4$ to $Sc = 512$.

Table 5.3: Alignment of scalar gradients with principal strain directions ($\mathbf{e}_\alpha$ most extensional, $\mathbf{e}_\beta$ intermediate, $\mathbf{e}_\gamma$ most compressive) for various Schmidt numbers.

Run	Sc	G_α	G_β	G_γ
1	4	0.1854	0.1273	0.6872
5	8	0.1864	0.1292	0.6843
12	32	0.1889	0.1311	0.6800
15	128	0.1935	0.1360	0.6705
16	512	0.1915	0.1326	0.6760

strain axes, as sustained alignment with the most compressive direction leads to rapid amplification of the scalar gradients. It is a matter of post-processing to calculate the principal strain directions, and to calculate the alignment of the scalar gradients with the local eigenframe. Figure 5.10 presents the direction cosines for different Schmidt numbers, and the data bears strong resemblance to that reported by Vedula *et al.* (2001). Specifically, the Schmidt number dependence is very weak (for a fixed R_λ), and the scalar gradient alignment with the most compressive eigendirection is strong.

Following Vedula *et al.* (2001), measures of the alignment of the form

$$G_k = \frac{\langle (\nabla\theta \cdot \mathbf{e}_k)^2 \rangle}{\langle |\nabla\theta|^2 \rangle} \tag{5.15}$$

are calculated, where $\mathbf{e}_k$ is a given eigendirection. The results are presented in table 5.3, where it is evident that the scalar gradients are preferentially aligned with $\mathbf{e}_\gamma$. However, the alignment appears to weaken slightly as the Schmidt number is increased. Vedula *et al.* (2001) reported an increasing alignment with the most compressive eigendirection for Schmidt numbers increasing to $Sc = 1$ from below. That, combined with the result in table 5.3 suggests that the alignment with the most compressive eigendirection is maximal around $Sc = 1$.

Thus far the analysis of scalar gradients has taken place in real (physical) space. There is also much that can be learned about the evolution of scalar gradients in Fourier space, where there is a notion of scale size. The dynamical equation for the scalar gradients in Fourier space is obtained by taking the Fourier transform of (5.13):

$$\begin{aligned} \frac{d\hat{\theta}_i(\mathbf{k})}{dt} &= -\hat{G}_i(\mathbf{k}) - \widehat{\theta_j s_{ji}}(\mathbf{k}) - \langle\Theta_j\rangle \hat{s}_{ji}(\mathbf{k}) - \\ &\frac{1}{2}\epsilon_{ijk}\widehat{\theta_j \omega_k}(\mathbf{k}) - \frac{1}{2}\epsilon_{ijk}\langle\Theta_j\rangle\hat{\omega}_k(\mathbf{k}) - Dk^2\hat{\theta}_i(\mathbf{k}) \ , \end{aligned} \tag{5.16}$$

where the advection term is $\hat{G}_i(\mathbf{k}) = \mathrm{i}k_j\widehat{u_j\theta_i}(\mathbf{k})$. The scalar gradient spectral covariance is then defined to be

$$E_{\theta,ij}(\mathbf{k}) = \langle \hat{\theta}_i^*(\mathbf{k})\hat{\theta}_j(\mathbf{k}) \rangle \ , \tag{5.17}$$

and its evolution equation is derived using (5.16). The result is

$$\frac{dE_{\theta,ij}(\mathbf{k})}{dt} = T_{\theta,ij}(\mathbf{k}) + s_{\theta,ij}(\mathbf{k}) + s_{\langle\Theta\rangle,ij}(\mathbf{k}) + \omega_{\theta,ij}(\mathbf{k}) + \omega_{\langle\Theta\rangle,ij}(\mathbf{k}) - D_{\theta,ij}(\mathbf{k}) \ , \tag{5.18}$$

where the various terms are

$$T_{\theta,ij}(\mathbf{k}) = -\langle\hat{\theta}_i^*(\mathbf{k})\hat{G}_j(\mathbf{k})\rangle - \langle\hat{\theta}_j^*(\mathbf{k})\hat{G}_i(\mathbf{k})\rangle^* \ , \tag{5.19}$$

$$s_{\theta,ij}(\mathbf{k}) = -\langle\hat{\theta}_i^*(\mathbf{k})\widehat{\theta_k s_{kj}}(\mathbf{k})\rangle - \langle\hat{\theta}_j^*(\mathbf{k})\widehat{\theta_k s_{ki}}(\mathbf{k})\rangle^* \ , \tag{5.20}$$

$$s_{\langle\Theta\rangle,ij}(\mathbf{k}) = -\langle\Theta_k\rangle\langle\hat{\theta}_i^*(\mathbf{k})\hat{s}_{kj}(\mathbf{k})\rangle - \langle\Theta_k\rangle\langle\hat{\theta}_j^*(\mathbf{k})\hat{s}_{ki}(\mathbf{k})\rangle^* \ , \tag{5.21}$$

$$\omega_{\theta,ij}(\mathbf{k}) = -\frac{1}{2}\epsilon_{jkl}\langle\hat{\theta}_i^*(\mathbf{k})\widehat{\theta_k\omega_l}(\mathbf{k})\rangle - \frac{1}{2}\epsilon_{ikl}\langle\hat{\theta}_j^*(\mathbf{k})\widehat{\theta_k\omega_l}(\mathbf{k})\rangle^* \ , \tag{5.22}$$

$$\omega_{\langle\Theta\rangle,ij}(\mathbf{k}) = -\frac{1}{2}\epsilon_{jkl}\langle\Theta_k\rangle\langle\hat{\theta}_i^*(\mathbf{k})\hat{\omega}_l(\mathbf{k})\rangle - \frac{1}{2}\epsilon_{ikl}\langle\Theta_k\rangle\langle\hat{\theta}_j^*(\mathbf{k})\hat{\omega}_l(\mathbf{k})\rangle^* \ , \tag{5.23}$$

$$D_{\theta,ij}(\mathbf{k}) = 2Dk^2E_{\theta,ij}(\mathbf{k}) \ , \tag{5.24}$$

and represent the physical mechanisms of advection ($T_{\theta,ij}(\mathbf{k})$), amplification of the scalar gradients by the fluctuating strain rate ($s_{\theta,ij}(\mathbf{k})$), production of scalar gradients through interactions of the fluctuating strain rate and the mean scalar gradient ($s_{\langle\Theta\rangle,ij}(\mathbf{k})$), rotation of the scalar gradients by the fluctuating vorticity ($\omega_{\theta,ij}(\mathbf{k})$), production of scalar gradients through interaction of the fluctuating vorticity and the mean scalar gradient ($\omega_{\langle\Theta\rangle,ij}(\mathbf{k})$), and dissipation of scalar gradients by molecular diffusion ($D_{\theta,ij}(\mathbf{k})$). Integrating the equation for $E_{\theta,ij}(\mathbf{k})$ over all of Fourier space and taking the trace recovers (5.14). It is important to note that the terms $T_{\theta,ij}(\mathbf{k})$, $s_{\theta,ij}(\mathbf{k})$, and $\omega_{\theta,ij}(\mathbf{k})$ are nonlinear, and thus involve interactions between many Fourier modes. This point will be revisited after presenting some preliminary results for this type of analysis.

The spectral budgets of the scalar gradients have been analyzed for the highest resolution datasets, focusing on Runs 2, 6, and 13 for $Sc = 4$, 8, and 32, respectively, with results presented in figure 5.11. The spectra are integrated in Fourier space over spherical shells, and are normalized by the Batchelor scale and the trace of the dissipation rate tensor of the scalar gradients (last term in 5.14). There appears to be much going on at high wavenumbers near the Batchelor scales, as one would expect. For reference, the peak of the dissipation spectrum of the scalar variance

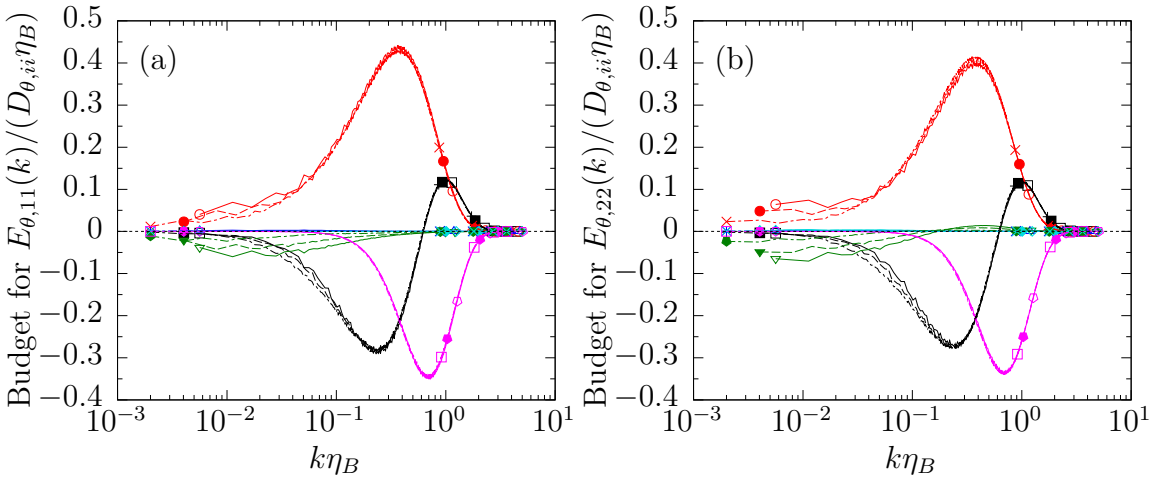

Figure 5.11: Spectral budget of scalar gradient covariance tensor components: (a) for gradients in the direction of the mean scalar gradient, and (b) for transverse gradients. Upper (red) curve for $s_{\theta,\alpha\alpha}(k)$, lowest (magenta) curve for $D_{\theta,\alpha\alpha}(k)$, curve that transitions from negative to positive in both frames (black) for $T_{\theta,\alpha\alpha}(k)$, and last discernible (green) curve for $\omega_{\theta,\alpha\alpha}(k)$. Terms involving mean gradient cannot be distinguished in the plot. Different symbols and line types for different Schmidt numbers, with Schmidt number increasing for curves extending to lower $k\eta_B$.

is around $0.2 \lesssim k\eta_B \lesssim 0.3$, and the peak of the dissipation spectrum for the scalar gradients is observed to be pushed out to even higher wavenumbers (expected). The nonlinear amplification spectrum $s_{\theta,\alpha\alpha}(k)$ is seen to dominate the budget over a wide range of wavenumbers, and is balanced by the advection and vorticity terms at lower wavenumbers. There does not, however, appear to be a strong dependence on the Schmidt number at high wavenumbers, suggesting that the scalar gradient budget takes a universal form at very high Schmidt numbers. As the Schmidt number increases, the various spectra cover a wider range of wavenumbers (extending to the left), and decrease in overall magnitude in the low wavenumber range.

While the analysis presented in figure 5.11 does provide some insight into what processes influence the evolution of the scalar gradients, there is still much that can be done to gain a better understanding. The shortcoming of the current analysis is that the nonlinear interactions, which involve various scales acting together, are not fully described by figure 5.11. Take, for example, the nonlinear amplification spectrum,

which can be written in the following manner after expanding the convolution sum

$$s_{\theta,ij}(\mathbf{k}) = -\left\langle \hat{\theta}_i^*(\mathbf{k}) \sum_{\mathbf{p}} \hat{\theta}_k(\mathbf{k}-\mathbf{p})\hat{s}_{kj}(\mathbf{p}) \right\rangle - \left\langle \hat{\theta}_j^*(\mathbf{k}) \sum_{\mathbf{p}} \hat{\theta}_k(\mathbf{k}-\mathbf{p})\hat{s}_{ki}(\mathbf{p}) \right\rangle^* . \quad (5.25)$$

The production of $E_{\theta,ij}(\mathbf{k})$ at a mode $\mathbf{k}$ is seen to depend on triads of wavenumbers: $\mathbf{k}$, $\mathbf{p}$, and $\mathbf{k}-\mathbf{p}$, and it would be very useful to identify what types of interactions are more important than others. Such analysis can be carried out through selective filtering of the wavenumbers used in the convolution sum, an approach which proved very successful in classical spectral transfer analysis for the velocity field (Domaradzki & Rogallo, 1990). In the high Schmidt number limit, specifically, it is likely the case that the strain rate fluctuations that dominate the evolution of the scalar gradients through this nonlinear mechanism occur in the wavenumber range of the energy dissipation range. Further elucidating the dynamics of scalar gradients in this manner might help improve understanding of mixing in general, and could also be useful in modeling.

5.6 Summary

This chapter presents results from a new DNS database intended to study turbulent mixing at high Schmidt numbers. Using the parallel algorithms developed in Chapter IV, petascale computational resources have enabled the first known turbulent mixing simulations with 8192^3 grid points for the scalar field. The largest simulations attain $Sc = 512$ for the passive scalar, which is comparable to salinity mixing in the ocean, while also maintaining a high enough Reynolds number ($R_\lambda \approx 140$) to observe a short inertial range the velocity field.

Thus far the DNS database has been used to investigate some fundamental questions regarding the Schmidt number dependence of certain passive scalar statistics. Important among these is the violation of local isotropy in the scalar field, and the hypothesis that local isotropy might be attained in the limit of high Schmidt num-

ber. A trend towards local isotropy is observed in the simulations, which is evident in frame (a) of figure 5.4 showing the skewness of the scalar gradient parallel to the mean gradient (which should be zero if isotropy holds), which exhibits a power-law dependence on the Schmidt number close to $Sc^{-0.45}$. The data appears very robust, as results from past DNS using different numerical methods for the scalar field (Donzis & Yeung, 2010) show very similar results at moderate Schmidt numbers. It has also been suggested in the literature that small-scale intermittency in the scalar field might saturate at high Schmidt number. To check this hypothesis with the new DNS data, the flatness factors of scalar gradients and the scalar dissipation rate PDF are presented in figure 5.6. While the PDF of the scalar dissipation rate is highly suggestive of saturation, higher resolutions yet are needed to confirm the expectation that the flatness factors of the scalar gradients might saturate at high Schmidt number.

Also of interest is the structure of the scalar field, and in particular, what shape the passive scalar spectrum attains at high Schmidt number. Both Batchelor and Kraichnan predicted that in the viscous-convective range the scalar spectrum would exhibit k^{-1} scaling, although they differ in their predictions for the far-diffusive range. The time-averaged spectra presented in figure 5.7 are very suggestive of Batchelor scaling, and agree with the exponential decay in the far-diffusive range predicted by Kraichnan. The development of a broad scaling range in the scalar field with increasing Schmidt number is also investigated in physical space with Yaglom's relation, where the result presented in figure 5.8 shows good agreement with the theoretical result over almost one decade of separations at $Sc = 512$.

Because mixing ultimately occurs at the small scales, statistics of scalar gradients have also been analyzed. With the setting being the principal axes of the strain rate tensor, there is good agreement with the expectation that scalar gradients are preferentially aligned with the direction of the most compressive principal strain rate. There is a slight weakening of the alignment at higher Schmidt numbers, which com-

bined with the results of Vedula *et al.* (2001) suggest that alignment is maximal near unity Schmidt number. Historically, much analysis of scalar gradients has taken place in physical space (Vedula *et al.*, 2001; Brethouwer *et al.*, 2003), where statistics such as the nonlinear amplification of the scalar gradient variances due to the fluctuating strain rates are calculated. To further elucidate scalar gradient evolution, the governing equation for the scalar gradient spectral covariance is derived, where physical processes such as nonlinear amplification are calculated on a scale-dependent basis. Preliminary results show nonlinear amplification dominating the evolution of the scalar gradients over a wide range of scales, with overall weak Schmidt-number dependencies in the shapes of the various balance terms for the spectrum. Further work is required to understand the locality of the interactions in the nonlinear terms, which could be useful to improve understanding of scalar gradient amplification in the viscous-convective range of scales at high Schmidt numbers.

CHAPTER VI

CONCLUSIONS

Turbulence remains a very challenging subject that is relevant to many science communities ranging from applied engineering to astrophysics (Sreenivasan, 1999). The unsolved nature of turbulence — owing to its strong nonlinearities — forces one to consider one particular flow of interest at a time, and to gradually build a deeper understanding and appreciation for the complexities of turbulence over time. Such an effort is undertaken in this thesis, where an emphasis is placed on the problems of turbulence subjected to irrotational mean strain and turbulent mixing of passive scalars. Turbulent flows are simulated using the method of direct numerical simulation (DNS), whereby the exact governing equations for the fluid motion are solved without modeling. This approach is very challenging from a computational perspective because turbulent flows possess a wide range of scales, from the large energy containing motions to the small dissipative scales, all of which must be adequately resolved in DNS. To handle the large computational requirements of DNS, the thesis emphasizes developing algorithms which enable simulations at unprecedented scales on many of today's high performance computing (HPC) architectures. This chapter summarizes these and other major efforts from this thesis, and looks forward to other problems and techniques which can further improve our understanding.

6.1 Summary of results and contributions

The research reported in this thesis has contributed to both the fundamental understanding of turbulence and turbulent mixing, and has demonstrated how HPC can be used efficiently in the pursuit of such knowledge. The following subsections provide a summary of the major results and contributions from each chapter of this thesis.

6.1.1 Turbulence under axisymmetric contraction

Simulations were conducted to study the evolution of turbulence subjected to irrotational mean strain in the form of axisymmetric contraction. The simulations used deforming computational domains and a strain rate which models the spatially-varying strain rate profile in the experimental wind tunnel facility of Ayyalasomayajula & Warhaft (2006) (AW henceforth). The experimental strain rate was modeled by extending the numerical method of Rogallo (1981) to evaluate the computational grid metrics and viscous integrating factors as functions of spatial location in a numerical wind tunnel, rather than of time. Key results presented include the axisymmetric energy spectrum and the budgets for the longitudinal and transverse 1-D spectra — quantities which are extremely difficult to measure experimentally. The simulations successfully confirmed the findings of AW that the "double-peak" spectral form that emerges in the transverse compensated 1-D spectrum following the contraction is a distinct result of high Reynolds number. Analysis of the spectral budgets showed that it is a result of very fast spectral transfer from low wavenumber to high wavenumbers.

6.1.2 Turbulent mixing under axisymmetric contraction

Building on the success of the numerical modeling of the AW wind tunnel, simulations were also conducted to study passive scalar mixing under axisymmetric contraction. The DNS used a numerical configuration similar to the wind tunnel experiments of Gylfason & Warhaft (2009) (GW henceforth; their experiments were conducted in the same wind tunnel as AW), where temperature fluctuations in air were studied by introducing a mean transverse temperature gradient into the flow. A key effort in the DNS was to broaden the scope of the investigation by also including a passive scalar with a streamwise mean gradient. The DNS showed that the rapid distortion theory (RDT) predictions of GW for the scalar spectrum and scalar gradients hold very well as the strain rate in the numerical wind tunnel is increased. The simula-

tions elucidated important differences that emerge during the contraction for scalars with mean gradients in different directions. Specifically, for the scalar with a mean streamwise gradient, an imbalance developed between the production and dissipation rates of scalar variance, such that the scalar variance was destroyed following the contraction. Similar to GW, during relaxation the scalar spectrum rapidly increased at high wavenumbers, which is similar to the findings for the velocity field.

6.1.3 Algorithms for petascale simulations of turbulent mixing at high Schmidt number

A major goal of this thesis was to develop numerical algorithms capable of simulating high Schmidt number turbulent mixing in moderately high Reynolds number turbulence (high enough for inertial-range scaling), requiring up to 8192^3 grid points for the passive scalar. To simulate such conditions efficiently, the dual-grid dual-scheme approach of Gotoh *et al.* (2012) was adopted, which uses high-order combined compact finite differences (CCD) to compute the passive scalar on a fine grid which resolves the Batchelor scale, and Fourier pseudo-spectral methods to compute the velocity field on a coarse grid which resolves the Kolmogorov scale. A major effort was to extend the ideas of Gotoh *et al.* (2012) to incorporate the physics of the passive scalar directly in the design of the new code. For simulations at high Schmidt number, this was accomplished by implementing a dual-communicator approach in which the velocity and scalar fields are computed using disjoint groups of processes, each matched to their respective problem size. In addition, the one-way coupling of the two fields enabled the design of a communication strategy in which the transfer of the velocity field from the velocity communicator to the scalar communicator is overlapped with computations in the larger scalar communicator. This work also addressed the challenges of achieving good scalability out to the large problems sizes required for high Schmidt number scalars. Scalability was improved by overlapping communication with com-

putation, where for the CPU-only version of the code run on the XE6 partition of Blue Waters at the University of Illinois, Urbana-Champaign, considerable improvement was attained by dedicating certain threads to perform communication while others compute concurrently. The code was also ported to run on the 27 petaflop heterogeneous GPU-accelerated machine Titan housed at Oak Ridge National Laboratory, TN, using the latest OpenMP 4.5 capabilities of the Cray compiler. Here too, scalability was improved by overlapping communication with computation, with such overlap being achieved through asynchronous execution between the CPU and GPU.

6.1.4 Turbulent mixing at high Schmidt number

Simulations using the algorithm described above were conducted on Blue Waters and Titan to generate a new DNS database for mixing in $R_\lambda \approx 140$ forced isotropic turbulence, with Schmidt numbers ranging from 4 to 512. The particular choice of $R_\lambda \approx 140$ was a good starting point for the simulations, given that some data up to $Sc = 64$ is available in the literature (Donzis & Yeung, 2010). Thus far the database has provided further support for the hypothesis that the scalar satisfies local isotropy and experiences a saturation of intermittency in the limit of high Schmidt number. The form of the scalar spectrum at high Schmidt number, which is a longstanding open question (Gotoh & Yeung, 2013), was also investigated. The DNS data support a trend towards Batchelor (k^{-1}) scaling in the viscous-convective range as the Schmidt number is increased, and an exponential decay in the far-diffusive range. Two-point statistics in physical space were also examined, where it was found that the third-order velocity-scalar structure function shows a broad scaling range at high Schmidt number, in good agreement with Yaglom's relation. Local isotropy and saturation of intermittency as functions of scale-size were assessed with scalar structure functions, and show similar trends with changes in the Schmidt number as the single-point statistics. New behavior, however, was observed as the skewness structure function

developed a local minimum for separations in the range of 20–30 Batchelor scales at higher Schmidt numbers. Scalar gradients were also studied, which show strong alignment with the most compressive principal strain direction, although the alignment weakens slightly as the Schmidt number is increased. Finally, the spectral budget for the scalar gradient covariance is derived and analyzed. The budget provides further insight into the evolution of the scalar gradients by showing how physical processes, e.g., nonlinear amplification by the strain rates, occur as functions of scale size.

6.2 Future considerations

Looking forward, DNS and HPC will continue to provide deep insight into the fundamental nature of turbulent flows. This thesis scratches the surface of many interesting problems, including anisotropic turbulence generated by mean strain, turbulent mixing, and parallel algorithms for large-scale simulations. There are many ways by which the current research can be extended to further improve our understanding of turbulence and our capabilities in HPC, as discussed below.

Physical-space significance of the double-peak spectral form

Following the application of axisymmetric contraction at sufficiently high Reynolds numbers, both the experiments of AW and the simulations reported in this thesis show the emergence of a double-peak structure in the transverse 1-D compensated spectrum. While the result is robust, and the physical mechanisms behind its formation have been elucidated, it is not clear how the double-peak relates to the flow structure in physical space. Flow visualization and analysis of how the double-peak structure affects the two-point correlation might be useful to understand this.

Effect of contraction ratio in strained simulations

The simulations of turbulence under axisymmetric contraction used a 4:1 area ratio to match the AW wind tunnel. One may then ask, what is the effect of the contraction

ratio on the spectral evolution? Clearly, the behavior is known for the limiting cases of no contraction (i.e., decaying isotropic turbulence) and the 4:1 area ratio contraction used by AW; however, additional simulations should be carried out for 2:1, 3:1, etc. area ratios to see when double-peak structure emerges for a given Reynolds number.

Grid refinement strategies for high contraction ratio simulations

The simulations of strained turbulence conducted for this thesis used a fixed number of grid points for all three phases of the simulations (i.e., the pre-simulation, the application of strain, and the post-contraction relaxation). The domain during the pre-simulation is short in the direction of extensional strain (see figure 2.1), and as a result has very fine grid spacing — much finer than might be necessary to resolve the small-scale fluctuations in that direction. This excessively fine grid spacing restricts the allowable time step during the pre-simulation, thus increasing the cost of that phase of the simulation. Instead of using a fixed number of grid points, an alternative strategy is to coarsen the grid in the direction of extensional strain during the pre-simulation, and to refine the grid during the contraction as the resolution worsens. This will increase the pre-simulation time step size, and can be accomplished by providing additional Fourier modes (when needed) in the extensional direction with zero initial value.

Reynolds number dependencies for other straining configurations

The simulations in this thesis for strained turbulence focused on the single configuration of turbulence under axisymmetric contraction, which AW showed evolves in a Reynolds-number dependent manner. A natural follow-up question is: what high Reynolds number behaviors are potentially awaiting discovery for other irrotational strains, e.g., axisymmetric expansion and plane strain?

Simulations with multiple strain rates applied in succession

While an emphasis was placed on a single axisymmetric contraction, there are many

engineering devices in which strains are applied in sequence, e.g., axisymmetric expansion follows axisymmetric contraction in a converging-diverging nozzle. The DNS code is more than capable of performing such simulations (after providing the straining profiles), which might prove useful for modeling purposes.

Transition from two-dimensional to three-dimensional turbulence

It was shown that during the application of sufficiently strong strain the turbulence approaches a limiting state of two-dimensional isotropic turbulence. This motivates a simple, but complicated problem: how would two-dimensional isotropic turbulence transition (through nonlinear interactions) to three-dimensional turbulence? To investigate this one can initialize a two-dimensional isotropic field, and superimpose three-dimensional perturbations on the flow field. At the time of this writing, it looks like there is some active research on this problem (Biferale *et al.*, 2017), which we are interested in pursuing.

Model assessment for passive scalars under strain

For passive scalars under axisymmetric contraction, it would be useful to make an assessment of commonly used turbulence models, and see if they can predict the evolution of the scalar flux during and after the contraction for scalars with mean gradients in different directions. Further comparisons with the scalar gradient models developed by Gylfason & Warhaft (2009) are also needed.

Additional insights from high Schmidt number DNS database

For high Schmidt number turbulent mixing, it is important that we gain a deeper understanding of some of the numerical results presented in this thesis. The most important of these is perhaps: why does the skewness of the scalar gradient parallel to the mean gradient decrease with a power-law dependence on the Schmidt number, and what determines the power-law exponent? Similar approaches towards local isotropy with increasing Schmidt number have been reported in the past (Yeung

et al., 2002). The decrease in the skewness may be to be related to the rolling up of thin scalar dissipation sheets by the flow (Brethouwer *et al.*, 2003). To gain a better understanding of the scalar gradients, and how they depend on Schmidt number, further analysis of the DNS database must be done.

Schmidt number dependencies in decaying scalar fields

Additional insights into scalar evolution can be gained from a somewhat simplified numerical configuration compared to the case of continuous forcing by a mean scalar gradient. Consider initializing the scalar fluctuations to be the negative of the velocity field in a given coordinate direction. Such an initial condition corresponds to generating scalar fluctuations with a uniform mean gradient activated for just an instant, and then turning the mean gradient off. The scalar gradients are initially positively skewed (due to the negative skewness of the velocity gradients), and their evolution will quickly differ for scalars at different Schmidt numbers. Although the scalar will be decaying, understanding how the gradients evolve in this simplified configuration might help explain the power-law dependence the skewness has on Sc for simulations in which the mean gradient is active.

Differential diffusion at high Schmidt number

There are other problems involving high Schmidt number scalars which have not been addressed in this work. One among these is differential diffusion, where the emphasis is shifted from studying the statistics of a single passive scalar to observing how scalars which are initially correlated become de-correlated as a result of diffusive processes. Differential diffusion is important in many applications which require scalars to be present in certain proportions at the same spatial location, e.g., in combustion applications the reactants must be brought together at the molecular level to react. While differential diffusion has been studied for modest Schmidt numbers (Yeung, 1998), little is known for the case of both scalars having high Schmidt numbers, further

insight into which would be very useful for applications involving liquid mixing.

High Schmidt number active scalars

It is also common that the scalar transported by the turbulent flow is active, meaning that it couples back to the momentum equation in the form of a forcing term. In incompressible flows a common example is a buoyancy force exerted on the fluid arising from fluctuations in the fluid density. Important examples of weakly-diffusive active scalars include temperature and salinity fluctuations in the ocean, which have a Prandtl number of 7 and a Schmidt number of approximately 700, respectively. It would be very useful to the utilize current DNS code to study the effects of one or more weakly-diffusive active scalars, as such simulations could provide some insight into important geophysical flows, e.g., the ocean.

High Schmidt number scalars under strain

It would also be very useful to extend the current work to consider passive scalars at high Schmidt number under the application of mean strain. Such studies would be relevant to mixing in water flowing through pipes of variable cross-section, and would serve as extensions to both the strained scalar work in Chapter III and the high Schmidt number work in Chapter V. To perform such simulations, the CCD code would need to be modified to solve for the passive scalar in the deforming (Rogallo) coordinate system, which should not be a very difficult task.

Lagrangian perspectives in turbulence

One limitation of the current work is that it only considers turbulence from the Eulerian (laboratory) reference frame. In the Lagrangian description of turbulence, the reference frame is switched to that of infinitesimal fluid particles advected by the turbulent flow. Many physical phenomena, e.g., turbulent dispersion of a contaminant, are more naturally described in a Lagrangian frame; however, as detailed by Yeung (2002), not only are Lagrangian studies somewhat rare in the literature, but there is a

strong need to consider more complex flows than isotropic turbulence. Following the work in this thesis on turbulence under axisymmetric contraction, Lagrangian studies should be conducted in this "simple" anisotropic flow (i.e., simple compared to inhomogeneous flows such as flows involving walls). Furthermore, Brethouwer *et al.* (2003) showed that considerable insight into the physics of high Schmidt number passive scalar mixing could be attained by considering the Lagrangian perspective. Lagrangian data obtained from mixing simulations at high Schmidt number would therefore be very useful.

Algorithms for high Reynolds mixing of moderate Schmidt number scalars

While the hybrid code developed in this thesis has been demonstrated to simulate very high Schmidt number scalars efficiently, the dual-communicator nature of the code can become a hindrance for moderate Schmidt number simulations. A current problem of interest is passive scalar mixing of a scalar with, say $Sc = 8$ or $Sc = 16$, but with a much higher Reynolds number than used in this thesis, e.g., $R_\lambda \approx 400$. Such a simulation would be useful to study the effects of small-scale velocity intermittency on the scalar field. For this purpose, the velocity field resolution should be kept very high, perhaps with $N_v = 4096$, while the scalar can be computed with $N_\theta = 8192$. The dual-communicator nature of the code is likely to be not optimal at this scale because the velocity field and scalar field communicators would be very close in size. To enable such simulations, the current FPS code can be augmented with the CCD scheme in a single-communicator format. Such an extension should be very straightforward, and would significantly expand the scope of problems that can be simulated.

Exploring the use of one-sided communication

Large-scale DNS of turbulence also demands that one continues to improve their understanding of HPC, so they can write codes that scale well and achieve high performance on current and emerging computing architectures. A starting point for

extending the current work is to consider different communication paradigms available through MPI. For example, it has previously been shown (not in this thesis) that for the FPS code a type of one-sided communication (specifically, co-array Fortran, or CAF) can be used to improve performance of certain collective communication calls. Unfortunately, it appears that CAF is severely lacking compiler support, so the one-sided communication supported by MPI-3 should be considered. One example from the hybrid code that should be tested with one-sided communication is the transfer of the velocity field from the velocity communicator to the scalar communicator. This communication is currently implemented with matching `MPI_SEND` and `MPI_IRECV` calls, but could be implemented with a one-sided `MPI_PUT` call from the velocity communicator to the scalar communicator, which might improve performance.

Task-based programming models for CFD code design

There is also potential room to improve the use of shared-memory programming in the codes developed for this work. One interesting paradigm to consider is that of task-based programming, which contrasts the fine-grained (i.e., loop-level) parallelism approach used in most of the subroutines developed in this work. Specifically, with OpenMP, units of work (i.e., actual computations or communication calls) can be encapsulated in the form of tasks, which are then executed by OpenMP threads. While task-based programming was in fact used for the GPU algorithms, for GPU execution it is a requirement, and the full power of OpenMP's task-based programming model was not explored. Its not entirely clear if performance can be improved with such an approach, but the combination of one-sided communication and task-based programming might be a viable option for many-core type machines. (Here it seems like one-sided communication might be a requirement, so as to avoid excessive synchronization or deadlocks between threads performing tasks in different orders.)

APPENDIX A

A NUMERICAL STUDY OF TURBULENCE UNDER TEMPORALLY EVOLVING AXISYMMETRIC CONTRACTION AND SUBSEQUENT RELAXATION

M. P. Clay and P. K. Yeung. 2016 *Journal of Fluid Mechanics* **805**, 460–493.

Abstract

Direct numerical simulations using up to 4096^3 grid points on a deforming domain have been used to study the response of initially isotropic turbulence to a period of spatially-uniform axisymmetric contraction (with one extensional and two equally compressive directions) and subsequent relaxation. A time-dependent strain rate is formulated to closely correspond to the downstream evolution in the wind tunnel experiments of Ayyalasomayajula & Warhaft (*J. Fluid Mech.* **566**, 273-307, 2006), with a smoothly-varying 4:1 contraction ratio. The application of strain leads to anisotropy in both the large scales and the small scales, in a manner where nonlinear effects not considered in rapid-distortion theory play an important role. Upon termination of strain the small scales quickly return to isotropy while a residual level of anisotropy appears to persist at the large scales. The simulations are shown to reproduce many key findings from experiments, including distinctive changes in the form of the one-dimensional spectra in the extensional direction that arise at sufficiently high Reynolds number, during both the straining and relaxation periods. Scale-dependent measures of anisotropy are presented in terms of one-dimensional spectra and axisymmetric versions of the energy spectrum. To explain the observed changes in spectral shapes, various terms in the spectral evolution equation representing rapid pressure-strain, slow pressure-strain, production, nonlinear transfer, and viscous dissipation are computed, showing that nonlinear effects take a dominant role when a wide range

of scales exists. In particular, the "double-peak" spectral form observed in experiments at high Reynolds number is found to be a consequence of the small scales relaxing towards isotropy much faster than the large scales. A comparison of results obtained from computational domains of varying sizes and grid resolutions show that the numerical findings are robust.

APPENDIX B

A NUMERICAL STUDY OF TURBULENCE UNDER TIME-DEPENDENT AXISYMMETRIC CONTRACTION AND SUBSEQUENT RELAXATION

M. P. Clay, P. K. Yeung and Z. Warhaft. Nov. 2015 *68th Annual Meeting of the Division of Fluid Dynamics of The American Physical Society*, Boston, MA.

Abstract

Turbulence subjected to axisymmetric strain is a fundamental problem which is common in engineering equipment with variable cross-section, but is not yet fully understood. We have performed direct numerical simulations on a deforming domain with grids up to 1024^3 and a time-dependent strain history designed to mimic spatial gradients in wind-tunnel experiments (Ayyalasomayajula & Warhaft *J. Fluid Mech.* 566, 273-307 (2006)). Isotropic turbulence with a specified energy spectrum is allowed to decay and then passed through a numerical conduit of 4:1 contraction ratio. The Reynolds stress tensor, velocity gradient variances, and longitudinal and transverse one-dimensional (1-D) spectra are studied during both the contraction and subsequent relaxation. Contraction leads to amplification of energy in the compressed directions and departures from local isotropy. When the strain is removed local isotropy returns quickly while the energy decays with a power law exponent smaller than for decaying isotropic turbulence. The evolution of 1-D spectra including changes in shape is consistent with experiments, but a large solution domain is important.

APPENDIX C

A DUAL COMMUNICATOR AND DUAL GRID-RESOLUTION ALGORITHM FOR PETASCALE SIMULATIONS OF TURBULENT MIXING AT HIGH SCHMIDT NUMBER

M. P. Clay, D. Buaria, T. Gotoh and P. K. Yeung. 2017 *Computer Physics Communications* **219**, 313–328.

Abstract

A new dual-communicator algorithm with very favorable performance characteristics has been developed for direct numerical simulation (DNS) of turbulent mixing of a passive scalar governed by an advection-diffusion equation. We focus on the regime of high Schmidt number (Sc), where because of low molecular diffusivity the grid-resolution requirements for the scalar field are stricter than those for the velocity field by a factor $\sqrt{Sc}$. Computational throughput is improved by simulating the velocity field on a coarse grid of N_v^3 points with a Fourier pseudo-spectral (FPS) method, while the passive scalar is simulated on a fine grid of N_θ^3 points with a combined compact finite difference (CCD) scheme which computes first and second derivatives at eighth-order accuracy. A static three-dimensional domain decomposition and a parallel solution algorithm for the CCD scheme are used to avoid the heavy communication cost of memory transposes. A kernel is used to evaluate several approaches to optimize the performance of the CCD routines, which account for 60% of the overall simulation cost. On the petascale supercomputer Blue Waters at the University of Illinois, Urbana-Champaign, scalability is improved substantially with a hybrid MPI-OpenMP approach in which a dedicated thread per NUMA domain overlaps communication calls with computational tasks performed by a separate team of threads spawned using OpenMP nested parallelism. At a target production problem

size of 8192^3 (0.5 trillion) grid points on $262,144$ cores, CCD timings are reduced by 34% compared to a pure-MPI implementation. Timings for 16384^3 (4 trillion) grid points on $524,288$ cores encouragingly maintain scalability greater than 90%, although the wall clock time is too high for production runs at this size. Performance monitoring with CrayPat blue for problem sizes up to 4096^3 shows that the CCD routines can achieve nearly 6% of the peak flop rate. The new DNS code is built upon two existing FPS and CCD codes. With the grid ratio $N_\theta/N_v = 8$, the disparity in the computational requirements for the velocity and scalar problems is addressed by splitting the global communicator `MPI_COMM_WORLD` into disjoint communicators for the velocity and scalar fields, respectively. Inter-communicator transfer of the velocity field from the velocity communicator to the scalar communicator is handled with discrete send and non-blocking receive calls, which are overlapped with other operations on the scalar communicator. For production simulations at $N_\theta = 8192$ and $N_v = 1024$ on $262,144$ cores for the scalar field, the DNS code achieves 94% strong scaling relative to $65,536$ cores and 92% weak scaling relative to $N_\theta = 1024$ and $N_v = 128$ on 512 cores.

APPENDIX D

PARALLEL ALGORITHM USED TO SOLVE PERIODIC BLOCK TRIDIAGONAL SYSTEM OF EQUATIONS UNDER A STATIC THREE-DIMENSIONAL DOMAIN DECOMPOSITION

The material presented here is borrowed from the Appendix in Clay *et al.* (2017).

Some of the most critical computational operations in this work involve forming and solving the system of equations represented by (4.6) in Section 4.1.2. For the sake of completeness, this appendix presents the essential details for the parallel algorithm used to solve the CCD linear system. The parallel algorithm was originally developed by Nihei & Ishii (2003) as an extension to the algorithm by Mattor *et al.* (1995) for tridiagonal matrices. A key feature of the algorithm is that the solution is obtained with data distributed across multiple parallel processes without using transposes. The notation and presentation of Nihei & Ishii (2003) is closely followed to facilitate comparisons when referencing their work. It should be noted that the word "vector" here takes its common meaning in computer science.

The eighth-order CCD scheme of Mahesh (1998) on a periodic grid line of N points can be written as

$$\begin{bmatrix} B & C & O & \cdots & A \\ A & B & C & & \vdots \\ O & \ddots & \ddots & \ddots & O \\ \vdots & & A & B & C \\ C & \cdots & O & A & B \end{bmatrix} \begin{bmatrix} \vec{f}_1 \\ \vec{f}_2 \\ \vdots \\ \vec{f}_{N-1} \\ \vec{f}_N \end{bmatrix} = \begin{bmatrix} \vec{g}_1 \\ \vec{g}_2 \\ \vdots \\ \vec{g}_{N-1} \\ \vec{g}_N \end{bmatrix}, \tag{4.1}$$

where the 2×2 block elements forming the matrix are

$$A = \begin{bmatrix} 51 & 9h \\ -138 & -18h \end{bmatrix} \ , \ B = \begin{bmatrix} 108 & 0 \\ 0 & 108h \end{bmatrix} \ , \ C = \begin{bmatrix} 51 & -9h \\ 138 & -18h \end{bmatrix} \ , \qquad (4.2)$$

where h is the grid spacing and O is the 2×2 zero matrix. The linear system relates the vectors $\vec{f}_i$ containing the first and second derivatives at the grid point x_i to the functional values stored in the vectors $\vec{g}_i$, both given as

$$\vec{f}_i = \begin{bmatrix} f_i' \\ f_i'' \end{bmatrix} \ , \ \vec{g}_i = \frac{1}{h} \begin{bmatrix} 107(f_{i+1} - f_{i-1}) - (f_{i+2} - f_{i-2}) \\ -(f_{i+2} + f_{i-2}) + 352(f_{i+1} + f_{i-1}) - 702 f_i \end{bmatrix} . \qquad (4.3)$$

Modifications are made for $\vec{g}_i$ at the extreme points to satisfy periodic boundary conditions. For example, at $i = 1$ and $i = N$ one obtains

$$\vec{g}_1 = \frac{1}{h} \begin{bmatrix} 107(f_2 - f_N) - (f_3 - f_{N-1}) \\ -(f_3 + f_{N-1}) + 352(f_2 + f_N) - 702 f_1 \end{bmatrix} , \qquad (4.4)$$

$$\vec{g}_N = \frac{1}{h} \begin{bmatrix} 107(f_1 - f_{N-1}) - (f_2 - f_{N-2}) \\ -(f_2 + f_{N-2}) + 352(f_1 + f_{N-1}) - 702 f_N \end{bmatrix} . \qquad (4.5)$$

The task is to solve (4.1) in parallel using P processors (MPI processes) numbered $p = 0, 1, \cdots, P-1$. In the implementation P is the size of the sub-communicator in a given direction, e.g., $P = P_1$ in the $P_1 \times P_2 \times P_3$ 3-D process layout discussed earlier when derivatives are taken in the x_1 direction. When the grid line of N points is distributed among the P processors the p^{th} process is responsible for the $M = N/P$ points $x_{pM+1}, \cdots, x_{(p+1)M}$. The $N \times N$ linear system of 2×2 block elements is then partitioned among the processors such that the p^{th} process solves a $M \times M$ linear

system comprised of 2×2 block elements, as

$$\begin{bmatrix} B & C & O & \cdots & O \\ A & B & C & & \vdots \\ O & \ddots & \ddots & \ddots & O \\ \vdots & & A & B & C \\ O & \cdots & O & A & B \end{bmatrix} \begin{bmatrix} \vec{X}_{p,1}^{(0)} \\ \vec{X}_{p,2}^{(0)} \\ \vdots \\ \vec{X}_{p,M-1}^{(0)} \\ \vec{X}_{p,M}^{(0)} \end{bmatrix} = \begin{bmatrix} \vec{g}_{pM+1} \\ \vec{g}_{pM+2} \\ \vdots \\ \vec{g}_{(p+1)M-1} \\ \vec{g}_{(p+1)M} \end{bmatrix} , \tag{4.6}$$

where this system is written as $L_p \vec{X}_p^{(0)} = \vec{D}_p$ following the notation of Nihei & Ishii (2003). Note that solving $L_p \vec{X}_p^{(0)} = \vec{D}_p$ corresponds to Operation B in Table 4.1. The LU factorization of L_p is precomputed and stored once at the beginning of program execution, and is later used when solving (4.6). The solution from this linear system is lacking additional coupling terms present in (4.1). This is remedied by defining $X_p^{(1)}$ and $X_p^{(2)}$

$$L_p X_p^{(1)} = \begin{bmatrix} A \\ O \\ \vdots \\ O \end{bmatrix} , \; L_p X_p^{(2)} = \begin{bmatrix} O \\ \vdots \\ O \\ C \end{bmatrix} , \tag{4.7}$$

such that the final solution for the p^{th} process given by

$$\vec{X}_p = \begin{bmatrix} \vec{f}_{pM+1} \\ \vec{f}_{pM+2} \\ \vdots \\ \vec{f}_{(p+1)M-1} \\ \vec{f}_{(p+1)M} \end{bmatrix} \tag{4.8}$$

can be written as

$$\vec{X}_p = \vec{X}_p^{(0)} - X_p^{(1)} \vec{\xi}_p^{(1)} - X_p^{(2)} \vec{\xi}_p^{(2)} \; . \tag{4.9}$$

The vectors $\vec{\xi}_p^{(1)}$ and $\vec{\xi}_p^{(2)}$ couple the processors and are determined by the $2P \times 2P$ reduced linear system comprised of 2×2 block elements given by

$$\begin{bmatrix} X_{1,1}^{(1)} & X_{1,1}^{(2)} & O & \cdots & \cdots & O & I \\ X_{1,M}^{(1)} & X_{1,M}^{(2)} & I & O & & & O \\ O & I & X_{2,1}^{(1)} & X_{2,1}^{(2)} & O & & \vdots \\ \vdots & O & X_{2,M}^{(1)} & X_{2,M}^{(2)} & I & O & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & O \\ O & & & O & I & X_{p,1}^{(1)} & X_{p,1}^{(2)} \\ I & O & \cdots & \cdots & O & X_{p,M}^{(1)} & X_{p,M}^{(2)} \end{bmatrix} \begin{bmatrix} \vec{\xi}_1^{(1)} \\ \vec{\xi}_1^{(2)} \\ \vec{\xi}_2^{(1)} \\ \vec{\xi}_2^{(2)} \\ \vdots \\ \vec{\xi}_p^{(1)} \\ \vec{\xi}_p^{(2)} \end{bmatrix} = \begin{bmatrix} \vec{X}_{1,1}^{(0)} \\ \vec{X}_{1,M}^{(0)} \\ \vec{X}_{2,1}^{(0)} \\ \vec{X}_{2,M}^{(0)} \\ \vdots \\ \vec{X}_{p,1}^{(0)} \\ \vec{X}_{p,M}^{(0)} \end{bmatrix}, \quad (4.10)$$

where I is the 2×2 identity matrix. Since no single processor has the complete information to form the right hand side of (4.10) initially, inter-processor communication must be used to gather $\vec{X}_{p,1}^{(0)}$ and $\vec{X}_{p,M}^{(0)}$ from all processors participating in the parallel solution on a single processor that will solve the reduced system. Once the reduced system is solved, the vectors $\vec{\xi}_p^{(1)}$ and $\vec{\xi}_p^{(2)}$ are redistributed to the other processors so their respective portions of the solution to (4.1) can be finalized with (4.9).

When applying the CCD scheme in multiple dimensions, the reduced linear systems that couple a group of processes can be efficiently solved in a load-balanced manner. For derivatives in the x_1 direction on a grid of size $N_1 \times N_2 \times N_3$ under a $P_1 \times P_2 \times P_3$ process layout, each processor owns a subdomain of size $N_1/P_1 \times N_2/P_2 \times N_3/P_3$ and participates in the solution of $N_2 N_3/(P_2 P_3)$ linear systems. The P_1 processors aligned in the x_1 direction can divide the work for the reduced linear systems such that each processor solves the same number of reduced linear systems (albeit for different "lines" of data in the x_1 direction). The work is divided by appropriate packing of $\vec{X}_{p,1}^{(0)}$ and $\vec{X}_{p,M}^{(0)}$ followed by a `MPI_ALLTOALL` such that the destination tasks will have all of the necessary data to form the right hand side of (4.10) for each reduced linear system it is solving. The solution vectors $\vec{\xi}_p^{(1)}$ and $\vec{\xi}_p^{(2)}$ are then packed and sent to the

appropriate task with another `MPI_ALLTOALL`. These calls to `MPI_ALLTOALL` and the solution of the reduced linear system appear as Operations C, D, and E in Table 4.1. The final solution obtained by (4.9) after the $\vec{\xi}_p^{(1)}$ and $\vec{\xi}_p^{(2)}$ vectors have been received is Operation F in Table 4.1.

APPENDIX E

IMPROVING SCALABILITY AND ACCELERATING PETASCALE TURBULENCE SIMULATIONS USING OPENMP

M. P. Clay, D. Buaria and P. K. Yeung. Sept. 2017 (accepted) *OpenMP Developers Conference*, Stony Brook, NY.

Abstract

In this talk we will present our recently devised parallel implementations of a computational fluid dynamics code used to study turbulent mixing. Using advanced features of OpenMP, including the latest in OpenMP 4.5, we focus on overlapping communication with computation while maximizing the number of threads on the host device (CPU) or target device (GPU) for work-sharing. We have developed distinct strategies for such overlap in both homogeneous and heterogeneous computing environments. For simulations on the homogeneous Cray XE6 partition of Blue Waters at the University of Illinois, Urbana-Champaign, we have improved scalability using up to $524,288$ cores for our largest production problem size of 8192^3 (0.5 trillion) grid points by using OpenMP locks and nested parallelism to dedicate certain threads to perform only communication, while other threads compute concurrently. Substantial further gains in performance have been obtained by accelerating the code using OpenMP 4.5 on the heterogeneous Cray XK7 system Titan, housed at Oak Ridge National Laboratory, TN. Data movement between the CPUs and GPUs is minimized by transferring the entire memory space for the simulation to the GPUs, where the majority of the computations are performed, and only transferring data between the CPUs and GPUs as needed for communication. We use the latest tasking capabilities added to the `TARGET` constructs in OpenMP 4.5 (e.g., the `DEPEND` and `NOWAIT` clauses) to overlap computation on the GPUs with (i) communication on the CPUs and (ii)

data movement between the CPUs and GPUs. The combination of threading on the CPUs and the asynchronous algorithm further improves performance by 20 percent. The overall GPU to CPU speedup achieved is 2.5X.

BIBLIOGRAPHY

Ayyalasomayajula, S. & Warhaft, Z. 2006 Nonlinear interactions in strained axisymmetric high-Reynolds-number turbulence. *J. Fluid Mech.* **566**, 273–307.

Batchelor, G. K. 1946 The theory of axisymmetric turbulence. *Proc. Roy. Soc. Lond. A* **186**, 480–502.

Batchelor, G. K. 1953 *The Theory of Homogeneous Turbulence*. Cambridge University Press.

Batchelor, G. K. 1959 Small-scale variation of convected quantities like temperature in turbulent fluid. Part 1. General discussion and the case of small conductivity. *J. Fluid Mech.* **5**, 113–133.

Batchelor, G. K. & Proudman, I. 1954 The effect of rapid distortion of a fluid in turbulent motion. *Quart. J. Mech. Appl. Math.* **7**, 83–103.

Bauer, G. H., Brandt, J., Gentile, A., Kot, A. & Showerman, M. 2016 Dynamic model specific register (MSR) data collection as a system service. In *Proceedings of the Cray Users Group Conference*. London, England.

Biferale, L., Buzzicotti, M. & Linkmann, M. 2017 From two-dimensional to three-dimensional turbulence through two-dimensional three-component flows. *Phys. Fluids* **29**, 111101.

Bilger, R. W. 2004 Some aspects of scalar dissipation. *Flow. Turbul. Combust.* **72**, 93–114.

Bode, B., Butler, M., Dunning, T., Hoefler, T., Kramer, W., Gropp, W. & Hwu, W. 2013 The Blue Waters super-system for super-science. In *Contemporary High Performance Computing: From Petascale toward Exascale* (ed. J. S. Vetter). Chapman and Hall.

Brethouwer, G., Hunt, J. C. R. & Nieuwstadt, F. T. M. 2003 Microstructure and Lagrangian statistics of the scalar field with a mean gradient in isotropic turbulence. *J. Fluid Mech.* **474**, 193–225.

Brown, M. L., Parsheh, M. & Aidun, C. K. 2006 Turbulent flow in a converging channel: effect of contraction and return to isotropy. *J. Fluid Mech.* **560**, 437–448.

Budwig, R., Tavoularis, S. & Corrsin, S. 1985 Temperature fluctuations and heat flux in grid-generated isotropic turbulence with streamwise and transverse mean-temperature gradients. *J. Fluid Mech.* **153**, 441–460.

Canuto, C., Hussaini, M. Y., Quarteroni, A. & Zang, T. A. 2006 *Spectral Methods: Fundamentals in Single Domains*. Springer.

Chasnov, J. R. 1995 The decay of axisymmetric homogeneous turbulence. *Phys. Fluids* **7**, 600–605.

Chen, J., Meneveau, C. & Katz, J. 2006 Scale interactions of turbulence subjected to a straining-relaxation-destraining cycle. *J. Fluid Mech.* **562**, 123–150.

Choi, K.-S. & Lumley, J. L. 2001 The return to isotropy of homogeneous turbulence. *J. Fluid Mech.* **436**, 59–84.

Clay, M. P., Buaria, D., Gotoh, T. & Yeung, P. K. 2017 A dual communicator and dual grid-resolution algorithm for petascale simulations of turbulent mixing at high schmidt number. *Comput. Phys. Commun.* **219**, 313–328.

Clay, M. P. & Yeung, P. K. 2016 A numerical study of turbulence under temporally evolving axisymmetric contraction and subsequent relaxation. *J. Fluid Mech.* **805**, 460–493.

Corrsin, S. 1951 On the spectrum of isotropic temperature fluctuations in isotropic turbulence. *J. Appl. Phys.* **22**, 469–73.

Corrsin, S. 1952 Heat transfer in isotropic turbulence. *J. Appl. Phys.* **23**, 113–118.

Davidson, P. A., Okamoto, N. & Kaneda, Y. 2012 On freely decaying, anisotropic, axisymmetric Saffman turbulence. *J. Fluid Mech.* **706**, 150–172.

Domaradzki, J. A. & Rogallo, R. S. 1990 Local energy transfer and nonlocal interactions in homogeneous, isotropic turbulence. *Phys. Fluids A* **2**, 413–426.

Donzis, D. A. & Sreenivasan, K. R. 2010 The bottleneck effect and the kolmogorov constant in isotropic turbulence. *J. Fluid Mech.* **657**, 171–188.

Donzis, D. A., Sreenivasan, K. R. & Yeung, P. K. 2005 Scalar dissipation rate and dissipative anomaly in isotropic turbulence. *J. Fluid Mech.* **532**, 199–216.

Donzis, D. A., Sreenivasan, K. R. & Yeung, P. K. 2010 The Batchelor spectrum for mixing of passive scalars in isotropic turbulence. *Flow. Turbul. Combust.* **85**, 549–566.

Donzis, D. A. & Yeung, P. K. 2010 Resolution effects and scaling in numerical simulations of passive scalar mixing in turbulence. *Phys. D* **239**, 1278–1287.

Donzis, D. A., Yeung, P. K. & Sreenivasan, K. R. 2008 Dissipation and enstrophy in isotropic turbulence: resolution effects and scaling in direct numerical simulations. *Phys. Fluids* **20**, 045108.

Durran, D. R. 2010 *Numerical Methods for Fluid Dynamics with Applications to Geophysics*, 2nd edn. Springer.

Enos, J., Bauer, G., Brunner, R., Islam, S., Fiedler, R., Steed, M. & Jackson, D. 2014 Topology-aware job scheduling strategies for torus networks. In *Proceedings of the Cray Users Group Conference*. Lugano, Switzerland.

Eswaran, V. & Pope, S. B. 1988 An examination of forcing in direct numerical simulations of turbulence. *Comp. Fluids* **16**, 257–278.

Frigo, M. & Johnson, S. G. 2005 The design and implementation of FFTW3. *Proceedings of the IEEE* **93**, 216–231.

Fujii, Y., Azumi, T., Nishio, N., Kato, S. & Edahiro, M. 2013 Data transfer matters for GPU computing. In *International Conference on Parallel and Distributed Systems*, pp. 275–282. Seoul.

Gence, J. N. & Mathieu, J. 1979 On the application of successive plane strains to grid-generated turbulence. *J. Fluid Mech.* **93**, 501–513.

George, W. K. & Hussein, H. J. 1991 Locally axisymmetric turbulence. *J. Fluid Mech.* **233**, 1–23.

Gibson, M. M. & Dakos, T. 1993 Production of temperature fluctuations in grid turbulence: Wiskind's experiment revisited. *Exp. Fluids* **16**, 146–154.

Godeferd, F. S. & Staquet, C. 2003 Statistical modelling and direct numerical simulations of decaying stably stratified turbulence. Part 2. Large-scale and small-scale anisotropy. *J. Fluid Mech.* **486**, 115–159.

Gotoh, T., Hatanaka, S. & Miura, H. 2012 Spectral compact difference hybrid computation of passive scalar in isotropic turbulence. *J. Comput. Phys.* **231**, 7398–7414.

Gotoh, T., Watanabe, T. & Miura, H. 2014 Spectrum of passive scalar at very high schmidt number in turbulence. *Plasma and Fusion Research* **9**, 3401019.

Gotoh, T., Watanabe, Y., Shiga, Y., Nakano, T. & Suzuki, E. 2007 Statistical properties of four-dimensional turbulence. *Phys. Rev. E* **75**, 016310.

Gotoh, T. & Yeung, P. K. 2013 Passive scalar transport in turbulence: a computational perpsective. In *Ten Chapters in Turbulence* (ed. P. A. Davidson, Y. Kaneda & K. R. Sreenivasan). Cambridge University Press.

Gualtieri, P. & Meneveau, C. 2010 Direct numerical simulations of turbulence subjected to a straining and destraining cycle. *Phys. Fluids* **22**, 065104.

Gylfason, A. & Warhaft, Z. 2009 Effects of axisymmetric strain on a passive scalar field: modelling and experiment. *J. Fluid Mech.* **628**, 339–356.

Hager, G., Schubert, G., Schoenemeyer, T. & Wellein, G. 2011 Prospects for truly asynchronous communication with pure MPI and hybrid MPI/OpenMP on current supercomputing platforms. In *Proceedings of the Cray Users Group Conference*. Fairbanks, Alaska.

Hoeflinger, J. P. & de Supinski, B. R. 2005 The OpenMP Memory Model. First International Workshop on OpenMP, Eugene, OR, USA.

Holzer, M. & Siggia, E. D. 1994 Turbulent mixing of a passive scalar. *Phys. Fluids* **6**, 1820–1837.

Ishihara, T., Gotoh, T. & Kaneda, Y. 2009 Study of high-Reynolds number isotropic turbulence by direct numerical simulation. *Annu. Rev. Fluid Mech.* **41**, 165–180.

Ishihara, T., Morishita, K., Yokokawa, M., Uno, A. & Kaneda, Y. 2016 Energy spectrum in high-resolution direct numerical simulations of turbulence. *Phys. Rev. Fluids* **1**, 082403(R).

Iyer, K. P. & Yeung, P. K. 2014 Structure functions and applicability of yaglom's relation in passive-scalar turbulent mixing at low schmidt numbers with uniform mean gradient. *Phys. Fluids* **26**, 085107.

Jagannathan, S. & Donzis, D. A. 2012 Massively parallel direct numerical simulations of forced compressible turbulence: a hybrid MPI/OpenMP approach. In *Proceedings of the* 1^{st} *Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*. Chicago, IL, USA.

Jiménez, J., Wray, A. A., Saffman, P. G. & Rogallo, R. S. 1993 The structure of intense vorticity in isotropic turbulence. *J. Fluid Mech.* **255**, 65–90.

Keckler, S. W., Dally, W. J., Khailany, B., Garland, M. & Glasco, D. 2011 GPUs and the future of parallel computing. *IEEE Micro* **31**, 7–17.

Kolmogorov, A. N. 1941 The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Dokl. Akad. Nuak. SSSR* **30**, 299–303.

Kraichnan, R. H. 1968 Small-scale structure of a scalar field convected by turbulence. *Phys. Fluids* **11**, 945–953.

Kraichnan, R. H. 1974 Convection of a passive scalar by a quasi-uniform random straining field. *J. Fluid Mech.* **64**, 737–762.

Krogstad, P.-Å. & Davidson, P. A. 2010 Is grid turbulence Saffman turbulence? *J. Fluid Mech.* **642**, 373–394.

Lee, C.-M., Gylfason, Á., Perlekar, P. & Toschi, F. 2015 Inertial particle acceleration in strained turbulence. *J. Fluid Mech.* **785**, 31–53.

Lee, M. J. & Reynolds, W. C. 1985 Numerical experiments on the structure of homogeneous turbulence. PhD thesis, Dept. of Mechanical Engineering, Stanford University, report TF-24.

Lele, S. K. 1992 Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.* **103**, 16–42.

Clark di Leoni, P., Cobelli, P. J., Mininni, P. D. & Dmitruk, P. 2014 Quantification of the strength of inertial waves in a rotating turbulent flow. *Phys. Fluids* **26**, 035106.

Liu, S., Katz, J. & Meneveau, C. 1999 Evolution and modelling of subgrid scales during rapid straining of turbulence. *J. Fluid Mech.* **387**, 281–320.

Lopez, M. G., Larrea, V. V., Joubert, W., Hernandez, O., Haidar, A., Tomov, S. & Dongarra, J. 2016 Towards achieving performance portability using directives for accelerators. In *Third Workshop on Accelerator Programming Using Directives*. Newark, DE.

Lumley, J. L. & Newman, G. R. 1977 The return to isotropy of homogeneous turbulence. *J. Fluid Mech.* **82**, 161–178.

Mahesh, K. 1998 A family of high order finite difference schemes with good spectral resolution. *J. Comput. Phys.* **145**, 332–358.

Mattor, N., Williams, T. J. & Hewett, D. W. 1995 Algorithm for solving tridiagonal matrix problems in parallel. *Parallel Computing* **21**, 1769–1782.

McCalpin, J. D. 1995 Memory bandwidth and machine balance in current high performance computers. *IEEE computer society technical committee on computer architecture (TCCA) newsletter* pp. 19–25.

Miller, P. L. & Dimotakis, P. E. 1996 Measurements of scalar power spectra in high Schmidt number turbulent jets. *J. Fluid Mech.* **308**, 129–146.

Mills, R. R. & Corrsin, S. 1959 Effect of contraction on turbulence and temperature fluctuations generated by a warm grid. *Tech. Rep.* 5-5-59W. NASA.

Mininni, P. D. 2011 Scale interactions in magnetohydrodynamic turbulence. *Annu. Rev. Fluid Mech.* **43**, 377–397.

Mininni, P. D., Rosenberg, D. & Pouquet, A. 2012 Isotropization at small scales of rotating helically driven turbulence. *J. Fluid Mech.* **699**, 263–279.

Moin, P. & Mahesh, K. 1998 Direct numerical simulation: a tool in turbulence research. *Annu. Rev. Fluid Mech.* **30**, 539–578.

Mydlarski, L. & Warhaft, Z. 1998 Passive scalar statistics in high-Péclet-number grid turbulence. *J. Fluid Mech.* **358**, 135–175.

Nihei, T. & Ishii, K. 2003 Parallelization of a highly accurate finite difference scheme for fluid flow calculations. *Theoretical and Applied Mechanics Japan* **52**, 71–81.

Obukhov, A. M. 1949 Structure of the temperature field in turbulent flows. *Izv. Akad. Nauk. SSSR, Geogr. Geofiz.* **13**, 58–69.

Orszag, S. A. & Patterson, G. S. 1972 Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.* **28**, 76–79.

Overholt, M. R. & Pope, S. B. 1996 Direct numerical simulation of a passive scalar with imposed mean gradient in isotropic turbulence. *Phys. Fluids* **8**, 3128–3148.

Pearson, J. R. A. 1959 The effect of uniform distortion on weak homogeneous turbulence. *J. Fluid Mech.* **5**, 274–288.

Peters, N. 2000 *Turbulent Combustion*. Cambridge University Press.

Piessens, R., de Doncker-Kapenga, E., Überhuber, C. W. & Kahaner, D. K. 1983 *QUADPACK: A Subroutine Package for Automatic Integration*. Springer.

Pope, S. B. 2000 *Turbulent Flows*. Cambridge University Press.

Rabenseifner, R., Hager, G. & Jost, G. 2009 Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In *Proceedings of the* 17^{th} *Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. Weimar, Germany.

Rabenseifner, R. & Wellein, G. 2003 Communication and optimization aspects of parallel programming models on hybrid architectures. *Int. J. High Perf. Comput. Applic.* **17**, 49–62.

Reynolds, A. J. & Tucker, H. J. 1975 The distortion of turbulence by general uniform irrotational strain. *J. Fluid Mech.* **68**, 673–693.

Rogallo, R. S. 1981 Numerical experiments in homogeneous turbulence. NASA Technical Memorandum 81315. NASA Ames Research Center.

Rogers, M. M. & Moin, P. 1987 The structure of the vorticity field in homogeneous turbulent flows. *J. Fluid Mech.* **176**, 33–66.

Saffman, P. G. 1967 The large-scale structure of homogeneous turbulence. *J. Fluid Mech.* **27**, 581–593.

Sagaut, P. & Cambon, C. 2008 *Homogeneous Turbulence Dynamics*. Cambridge University Press.

Sarkar, S. & Speziale, C. G. 1990 A simple nonlinear model for the return to isotropy in turbulence. *Phys. Fluids A* **2**, 84–93.

Savill, A. M. 1987 Recent developments in rapid-distortion theory. *Annu. Rev. Fluid Mech.* **19**, 531–575.

Schumacher, J. & Sreenivasan, K. R. 2003 Geometric Features of the Mixing of Passive Scalars at High Schmidt Numbers. *Phys. Rev. Lett.* **91**, 174501.

Shraiman, B. I. & Siggia, E. D. 2000 Scalar turbulence. *Nature* **405**, 639–646.

Sirivat, A. & Warhaft, Z. 1983 The effect of a passive cross-stream temperature gradient on the evolution of temperature variance and heat flux in grid turbulence. *J. Fluid Mech.* **128**, 323–346.

Sjögren, T. & Johansson, A. V. 1998 Measurement and modelling of homogeneous axisymmetric turbulence. *J. Fluid Mech.* **374**, 59–90.

Speziale, C. G. 1991 Analytical methods for the development of Reynolds-stress closures in turbulence. *Annu. Rev. Fluid Mech.* **23**, 107–157.

Sreenivasan, K. R. 1991 On local isotropy of passive scalars in turbulent shear flows. *Proc. Roy. Soc. Lond. A* **434**, 165–182.

Sreenivasan, K. R. 1995 On the universality of the Kolmogorov constant. *Phys. Fluids* **7**, 2778–2784.

Sreenivasan, K. R. 1996 The passive scalar spectrum and the Obukhov-Corrsin constant. *Phys. Fluids* **8**, 189–196.

Sreenivasan, K. R. 1999 Fluid turbulence. *Rev. Mod. Phys.* **71**, S383–S395.

Sreenivasan, K. R. & Antonia, R. A. 1997 The phenomenology of small-scale turbulence. *Annu. Rev. Fluid Mech.* **29**, 435–472.

Sreenivasan, K. R., Antonia, R. A. & Britz, D. 1979 Local isotropy and large structures in a heated turbulent jet. *J. Fluid Mech.* **94**, 745–775.

Sullivan, P. J. 1976 Dispersion of a line source in grid turbulence. *Phys. Fluids* **19**, 159–160.

Tavoularis, S., Bennett, J. C. & Corrsin, S. 1978 Velocity-derivative skewness in small Reynolds number, nearly isotropic turbulence. *J. Fluid Mech.* **88**, 63–69.

Townsend, A. A. 1976 *The structure of turbulent shear flow*, 2nd edn. Cambridge University Press.

Uberoi, M. 1956 Effect of wind-tunnel contraction on free-stream turbulence. *J. Aero. Sci.* **23**, 754–764.

Vedula, P., Yeung, P. K. & Fox, R. O. 2001 Dynamics of scalar dissipation in isotropic turbulence: a numerical and modelling study. *J. Fluid Mech.* **433**, 29–60.

Warhaft, Z. 1980 An experimental study of the effect of uniform strain on thermal fluctuations in grid-generated turbulence. *J. Fluid Mech.* **99**, 545–573.

Warhaft, Z. 2000 Passive scalars in turbulent flows. *Annu. Rev. Fluid Mech.* **32**, 203–240.

Warhaft, Z. 2009 Why we need experiments at high Reynolds numbers. *Fluid Dyn. Res.* **41**, 021401.

Yaglom, A. M. 1949 On the local structure of a temperature field in a turbulent flow. *Dokl. Akad. Nuak SSSR* **69**, 743–746.

Yakhot, V. & Sreenivasan, K. R. 2005 Anomalous scaling of structure functions and dynamic constraints on turbulence simulations. *J. Stat. Phys.* **121**, 823–841.

Yeung, P. K. 1998 Correlations and conditional statistics in differential diffusion: scalars with uniform mean gradients. *Phys. Fluids* **10**, 2621–2635.

Yeung, P. K. 2002 Lagrangian investigations of turbulence. *Annu. Rev. Fluid Mech.* **34**, 115–142.

Yeung, P. K., Brasseur, J. G. & Wang, Q. 1995 Dynamics of direct large-small scale couplings in coherently forced turbulence: concurrent physical- and Fourier-space views. *J. Fluid Mech.* **283**, 43–95.

Yeung, P. K. & Donzis, D. A. 2005 High-Reynolds-number simulation of turbulent mixing. *Phys. Fluids* **17**, 081703.

Yeung, P. K., Xu, S., Donzis, D. A. & Sreenivasan, K. R. 2004 Simulations of three-dimensional turbulent mixing for Schmidt numbers of the order 1000. *Flow. Turbul. Combust.* **72**, 333–347.

Yeung, P. K., Xu, S. & Sreenivasan, K. R. 2002 Schmidt number effects on turbulent transport with uniform mean scalar gradient. *Phys. Fluids* **14**, 4178.

Yeung, P. K., Zhai, X. M. & Sreenivasan, K. R. 2015 Extreme events in computational turbulence. *Proc. Nat. Acad. Sci.* **112**, 12633–12638.

Yeung, P. K. & Zhou, Y. 1997 Universality of the Kolmogorov constant in numerical simulations of turbulence. *Phys. Rev. E* **56**, 1746–1752.

Yokokawa, M., Itakura, K., Uno, A., Ishihara, T. & Kaneda, Y. 2002 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator. In *Proceedings of the Supercomputing Conference*. Baltimore.

Zusi, C. J. & Perot, J. B. 2013 Simulation and modeling of turbulence subjected to a period of uniform plane strain. *Phys. Fluids* **25**, 110819.

Zusi, C. J. & Perot, J. B. 2014 Simulation and modeling of turbulence subjected to a period of axisymmetric contraction or expansion. *Phys. Fluids* **26**, 115103.

2015 *OpenMP application programming interface*. OpenMP Architecture Review Board, 14–17, `www.openmp.org`.

2015 *Performance Measurement and Analysis Tools*. Cray Inc., S-2376-63, 24–27.