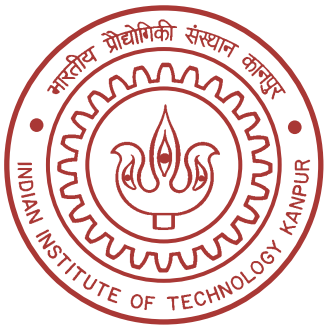
Spectral simulations of hydrodynamic and thermal turbulence for extreme resolutions

A Thesis Submitted
in partial fulfilment of the requirements
for the degree of
Doctor of Philosophy

by

Anando Gopal Chatterjee



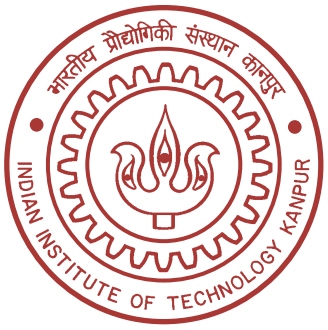
DEPARTMENT OF PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
March, 2018

Spectral simulations of thermal and hydrodynamic turbulence for extreme resolutions

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy

by

Anando Gopal Chatterjee



to the
DEPARTMENT OF PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
March, 2018

CERTIFICATE

It is certified that the work contained in the thesis titled "**Spectral simulations of thermal and hydrodynamic turbulence for extreme resolutions**", by **Anando Gopal Chatterjee**, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Prof. Mahendra K. Verma
(Thesis Supervisor)
Department of Physics
IIT Kanpur
Kanpur – 208016
India

Prof. Mainak Chaudhuri
(Thesis Co-supervisor)
Department of Computer Science
IIT Kanpur
Kanpur – 208016
India

April, 2018

Synopsis

Name	:	**Anando Gopal Chatterjee**
Roll No.	:	**12109870**
Degree for which submitted	:	**Doctor of Philosophy**
Department	:	**Physics**
Title of the Thesis	:	**Spectral simulations of thermal and hydrodynamic turbulence for extreme resolutions**
Thesis Supervisors	:	**Prof. Mahendra K. Verma** **Prof. Mainak Chaudhuri**
Month and year of submission	:	**March, 2018**

Turbulent flow is characterised by chaotic motion of fluid at different length and time scales. Its multi-scale features make it one of the most challenging problems in physics. Theoretical analysis of turbulence is very limited due to the complex nonlinearities present in the Navier -Stokes equation. Therefore, significant attempts have been made to understand turbulence through experiments and numerical simulations. Over time, very powerful supercomputers and software tools have emerged that have propelled the computational capabilities of turbulent flows to significant heights.

A turbulent flow contains a large number of interacting scales—from box size to dissipation scale. For Re $= 10^5$, the grid resolution required for resolution is 6000^3, which is a daunting task. Therefore, we need efficient flow solvers. Also, turbulent flows comes in various forms—-hydrodynamics, passive scalar, magnetohydrodynamics, thermal convection. Here, it is good to have a general purpose solver that can be used to solve these flows. I have contributed towards development of one such solver, Tarang.

We have developed a turbulent flow solver, Tarang, that is an object oriented program. It can solve for fluid, Rayleigh-Bénard Convection (RBC), Rayleigh-Taylor instability, Stratified flows, Magnetohydrodynamics, etc. Tarang can solve these systems with four boundary conditions. These are, i) periodic along the three directions (called

FFF basis), ii) wall along one and periodic along two directions (called SFF basis), iii) wall along two and periodic along one direction (called SSF basis), and iv) wall along three directions (called SSS basis). The main solver and many functions of Tarang are agnostic of the boundary condition. A large-resolution simulations involve input/output of large data sets. We have developed a library H5SI (HDF5 simple interface) that simplifies implementation of hdf5 library for various basis function.

A large fraction of computation time of a spectral solver is spent on Fast Fourier Transform (FFT). We have developed a pencil-based FFT library called FFTK (FFT Kanpur). This library scales well up to 196608 cores and grids up to 8192^3. Our study shows how MPI-all-to-all communication affects scaling. Also, FFTK works for FFF, SFF, SSF, and SSS basis.

We performed detailed scaling analysis of two solvers—hydrodynamic flows and thermal convection. We showed that these solvers scale quite well up to 196608 cores. The solvers exhibit strong and weak scaling.

We performed direct numerical simulation (DNS) of turbulent RBC on a 4096^3 grid for a Rayleigh number (ratio of the buoyancy term to the diffusive term) $\mathrm{Ra} = 1.1 \times 10^{11}$ and Prandtl number (ratio of kinematic viscosity and thermal diffusivity) $\mathrm{Pr} = 1$. Our grid resolution of RBC simulation is the highest ever employed for such studies, and the Rayleigh number of 1.1×10^{11} is the largest for any spectral simulation. These runs were performed on 196608 cores of Cray XC40 of KAUST. Using the simulation results we showed that turbulent thermal convection exhibits $E_u(k) \sim k^{-5/3}$, thus we show that turbulent thermal convection has behaviour similar to hydrodynamic turbulence.

We also performed a hydrodynamic turbulence simulation on a 4096^3 grid for $\mathrm{Re} = 6.8 \times 10^4$, and computed the energy spectrum and flux. We show that the numerical results matches well with Pao's predictions that $E_u(k) \sim k^{-5/3} \exp(-k^{4/3})$ and the energy flux $\Pi_u(k) \sim \exp(-k^{4/3})$ for both inertial and dissipative ranges. We also show that the shell-to-shell energy transfer is local and forward in both inertial and dissipative ranges.

The outline of the thesis is as follows:

In Chapter 1 we introduce various topic covered in the thesis.

In Chapter 2 we introduce the pseudo-spectral method and the structure of the code Tarang . Here we describe the class and directory structure of Tarang. In addition, we discuss the H5SI library that is used for parallel I/O.

In Chapter 3 we present the design of our parallel FFT library, FFTK (FFT Kanpur). In FFTK, along with slab and pencil decomposition, all the required boundary conditions have been incorporated. This chapter contains discussion on scaling results of FFTK.

In Chapter 4, we describe the scaling results of two of the Tarang's solvers—hydrodynamics and thermal convection.

In Chapter 5 we describe the turbulence models of Pao [1] and Pope [2] and compare them with high resolution energy spectrum produced by our simulations.

In Chapter 6 we compare kinetic energy spectrum of RBC simulation with Pao's model. The numerical curve matches quite well with the model curves in the inertial range. We show that convective turbulence follows Kolmogorov's model of fluid turbulence. Using the steady-state data of our simulation, we compute the shell-to-shell energy transfers and show that they are local and forward, very similar to hydrodynamic turbulence.

In Chapter 7 we summarize the main results obtained in this thesis.

[1] PAO, Y.H. Structure of turbulent velocity and scalar fields at large wavenumbers. Phys. Fluids 8, 6 (1965), 1063
[2] POPE, S. B. Turbulent Flows. Cambridge University Press, Cambridge, 2000

Acknowledgements

I am delighted to express my sincere gratitude and warm appreciation to all the persons who were instrumental to the completion of my Ph.D. journey.

First of all, I would like to express my deepest gratitude to my thesis advisors Prof. Mahendra K. Verma and Prof. Mainak Chaudhuri for providing me an opportunity to work with them on one of the most important and rich problems in fluid dynamics, performing spectral simulations of thermal and hydrodynamic turbulence for extreme resolutions. I am highly indebted to Prof. Mahendra K. Verma for providing me state-of-the-art computational resources. His devotion for work always motivated and inspired me to work harder. I learned the basics of my research from his courses on Turbulence, Magnetohydrodynamics, Nonlinear Dynamics, Computational Physics, and High Performance Computing. His guidance and constant supervision strongly supported me in completing this endeavor. He helped me a lot in writing of the thesis.

I am highly grateful to my Peer Review Committee members, Prof. Sagar Chakraborty, Prof. Anand Jha, and Prof. Saikat Ghosh for valuable suggestions on my work. I am thankful to Prof. Ravi Samtaney for introducing me to some new concepts in physics and for providing access to the computational facilities of KAUST, Saudi Arabia. I acknowledge all the teachers who taught me during my M.Sc. and course work. I thank Dr. Supriyo Paul for the fruitful discussions and suggestions during our collaboration at the initial stage of my Ph.D. I thank Prof. Pankaj K. Mishra for meticulous discussions and suggestions during our collaboration.

I would further like to thank my friends and colleagues Dr. Rohit Kumar, Dr. Ambrish Pandey, Dr. Abhishek Kumar for all kinds of help and support during my entire stay at IIT Kanpur. We have always shared our knowledge with each other. I have benefited from the countless discussions during lab hours. I am thankful to Dr. Dinesh Nath, Biplab Dutta, Dr. Rakesh Yadav and Mani Chandra for all their support.

I would like to thank Akanksha Gupta, Manohar Sharma, Mohammad Anas, Roshan Samuel, Shashwat Bhattacharya and Shubhadeep Sadhukhan for helping me to revise the thesis. I am thankful to Roshan Bhaskaran and Manmohan Dewbanshi for their cooperation and help.

I would also like to thank Mr. Arvind Mishra for all the computation related help, and all the staff members of the Department of Physics for their kind cooperation and

support.

I extend my thanks to IIT Kanpur for providing financial support to attend the international conferences.

I am grateful for the computational facilities offered by IIT Kanpur and CDAC Pune. Our numerical simulations were also performed on Shaheen-I and Shaheen-II at KAUST supercomputing laboratory under the project k97 and k1052. My work was supported by the research grants PLANEX/PHY/2015239 from Indian Space Research Organization, India, by the Department of Science and Technology, India (INT/RUS/RSF/P-03) and Russian Science Foundation Russia (RSF-16-41-02012) for the Indo-Russian project.

Finally, I would like to express my deep gratitude to my parents and family members for their continuous support and encouragement throughout the journey.

April, 2018 Anando Gopal Chatterjee

To my family and teachers...

List of Publications

Published/Accepted

1. **A. G. Chatterjee**, M. K. Verma, A. Kumar, R. Samtaney, B. Hadri, R. Khurram, "*Scaling of a Fast Fourier Transform and a pseudo-spectral fluid solver up to 196608 cores*", JPDC, 113, 77 (2018).

2. S. Sundar, M. K. Verma, A. Alexakis, and **A. G. Chatterjee**, "*Dynamic anisotropy in MHD turbulence induced by mean magnetic field*", Phys. Plasmas, 24, 022304 (2017).

3. A. Kumar, **A. G. Chatterjee**, and M. K. Verma, "*Energy spectrum of buoyancy-driven turbulence*", Phys. Rev. E, 90, 023016 (2014)

4. A. Pandey, A. Kumar, **A. G. Chatterjee**, and M. K. Verma, "*Dynamics of large-scale quantities in Rayleigh-Bénard convection*", Phys. Rev. E, 94, 053106 (2016).

5. A. Pandey, M. K. Verma, **A. G. Chatterjee**, and B. Dutta, "*Similarities between 2D and 3D convection for large Prandtl number*", Pramana-J. Phys., 87, 13 (2016).

6. M. K. Verma, A. Kumar, and **A. G. Chatterjee**, "*Energy spectrum and flux of buoyancy-driven turbulence*", Physics Focus, AAPPS Bulletin, 25, 45, 2015

7. M. K. Verma, **A. Chatterjee**, K. S. Reddy, R. K. Yadav, S. Paul, M. Chandra, and R. Samtaney, "*Benchmarking and scaling studies of pseudospectral code Tarang for turbulence simulations*", Pramana, 81, 617, 2013.

Conference Proceeding

1. M. K. Verma, A. Kumar, and **A. G. Chatterjee**, "*Energy Spectrum and Flux of Buoyancy-Driven Turbulence, In Proc. Advances in Computation, Modeling and Control of Transitional and Turbulent Flows*", Eds. T. K. Sengupta, S. Lele, K. R. Sreenivasan, and P. A. Davidson, p. 442, World Scientific (2016).

2. M. K. Verma, **A. Chatterjee**, and S. Reddy, "*Object-oriented Pseudo-spectral code Tarang for turbulence simulation*", In Proc. "ATIP/A*CRC Workshop on Accelerator Technologies for High-Performance Computing: Does Asia Lead the Way?", Singapore (2012).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Turbulent flow which is characterized by chaotic changes in pressure and flow velocity remains as one of the most challenging problems of physics. There are only a limited number of analytical results in turbulence. Therefore, researchers have attempted to understand turbulence using experiments and numerical simulations. In recent times, very powerful supercomputers and software tools have emerged that have propelled research in computational turbulence to unimaginable heights.

Some of the popular schemes to solve fluid flows are *finite difference*, *finite element*, *finite volume*, *spectral elements*, *pseudo-spectral*, *vortex method*, etc. The pseudo-spectral method [6, 9] is one of the most accurate schemes among them; it has been employed for studying physics of small-scale turbulence. This thesis is focussed on computational aspects, in particular spectral method, of turbulence.

We start our discussion by introducing spectral method for solving turbulent flows. In a pseudospectral code, approximately 70% to 80% of the total time is spent on the forward and inverse Fourier transforms. In next two sections we briefly explain the numerical schemes for FFT and the spectral solver Tarang.

1.1 Fast Fourier Transform

The inverse Fourier transform is defined as

$$f(x,y,z) = \sum_{k_x,k_y,k_z} f(k_x,k_y,k_z)\phi_{k_x}(x)\phi_{k_y}(y)\phi_{k_z}(z), \tag{1.1}$$

where $f(k_x, k_y, k_z)$ is the Fourier transform of $f(x, y, z)$. Here we compute $f(x, y, z)$ from $f(k_x, k_y, k_z)$. The functions $\phi_k(s)$ are the basis functions that appear in the following forms:

$$\text{Fourier}: \quad \phi_k(s) = \exp(iks), \tag{1.2a}$$

$$\text{Sine}: \quad \phi_k(s) = 2\sin(ks), \tag{1.2b}$$

$$\text{Cosine}: \quad \phi_k(s) = 2\cos(ks), \tag{1.2c}$$

where k could be k_x, k_y, or k_z, s could be x, y, or z, and $i = \sqrt{-1}$. We use Fourier basis function for the periodic boundary condition, and employ the sine and cosine basis functions for the free-slip boundary condition. The FFTK library can perform the above transformations in different combinations. , for the periodic boundary condition along the three directions, we employ Fourier basis function

$$\exp(ik_x x + ik_y y + ik_z z). \tag{1.3}$$

But for the free-slip boundary condition at all the three walls, u_z, the z-component of the velocity is expanded using the basis function

$$8\cos(k_x x)\cos(k_y y)\sin(k_z z). \tag{1.4}$$

Similar schemes are used for u_x and u_y. We term the above two basis functions as `FFF` (Fourier Fourier Fourier) and `SSS` (Sin/Cos Sin/Cos Sin/Cos) respectively. In a similar fashion, we define other basis functions—`SFF` for one free-slip wall direction and two periodic directions, `SSF` (Sin/Cos Sin/Cos Fourier)for two free-slip wall directions and one periodic direction, and `ChFF` (Chebyshev Fourier Fourier)for one no-slip wall direction and two periodic directions. Note that the inverse of sine (or cosine) basis function is sine (or cosine) itself, but $\exp(ikx)$ and $\exp(-ikx)$ are mutual inverses of each other.

In Sec. 3.1 we will describe the FFT implementation in our library.

1.2 Spectral Solver

As described in the introduction, the pseudospectral scheme is one of the most accurate methods to solve partial differential equations. The incompressible fluid equation is

described using the celebrated Navier Stokes equation, which is

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}, \tag{1.5}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{1.6}$$

where $\mathbf{u}$ is the velocity field, p is the pressure field, and ν is the kinematic viscosity. For simplicity we take the density of the fluid, ρ, to be unity. An estimate of the ration of the nonlinear term and the viscous term is Reynolds number, which is

$$\mathrm{Re} = \frac{UL}{\nu}, \tag{1.7}$$

where U and L represent the large-scale velocity and length scales respectively.

We rewrite the above equations in Fourier space:

$$(\partial_t + \nu k^2)u_j(\mathbf{k}) = -ik_l(u_l u_j)(\mathbf{k}) - ik_j p(\mathbf{k}), \tag{1.8}$$

$$k_j u_j(\mathbf{k}) = 0, \tag{1.9}$$

where $i = \sqrt{-1}$. The above equations are time advanced using standard methods, e.g., Runge Kutta scheme. The $u_l u_j$ term of Eq. (1.8) is a convolution in spectral space that is very expensive to compute. In spectral method, the Navier-Stokes equation is solved in Fourier-Space, except for the nonlinear term, which is calculated in real space and transformed back to Fourier Space [6, 9].

A spectral transform is general, and it can involve basis functions from Fourier series, sines and cosines, Chebyshev polynomials, spherical harmonics, or a combination of these functions. The FFTK library uses Fourier, sines, and cosine functions only. We plan to incorporate Chebyshev polynomials and spherical harmonics in future. In this thesis, we solve Eqs. (1.8,1.9) in a periodic box (`FFF` basis) using Tarang. Therefore we illustrate the implementation and usage of `FFF` basis, and then describe scaling analysis of FFT, and the fluid and convection solvers.

We have employed Tarang to simulate fluid and magnetohydrodynamic flows, liquid metal flows, Rayleigh-Bénard convection (RBC), rotating convection, and rotating flows. In the following we will illustrate one of the above modules, the RBC solver,

whose governing equations are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla\sigma + \alpha g\theta\hat{z} + \nu\nabla^2\mathbf{u}, \tag{1.10}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = \frac{\Delta}{d}u_z + \kappa\nabla^2\theta, \tag{1.11}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{1.12}$$

where θ is the temperature fluctuation from the steady conduction state,

$$T(x,y,z) = T_c(z) + \theta(x,y,z), \tag{1.13}$$

with T_c as the conduction temperature profile, σ is the pressure fluctuation, $\hat{z}$ is the buoyancy direction, Δ is the temperature difference between the two plates that are separated by a distance d, ν is the kinematic viscosity, and κ is the thermal diffusivity.

We solve a nondimensionalized version of the RBC equations, which are obtained using d as the length scale, $(\alpha g\Delta d)^{1/2}$ as the velocity scale, and Δ as the temperature scale:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla\sigma + \theta\hat{z} + \sqrt{\frac{\mathrm{Pr}}{\mathrm{Ra}}}\nabla^2\mathbf{u}, \tag{1.14}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = u_z + \frac{1}{\sqrt{\mathrm{RaPr}}}\nabla^2\theta. \tag{1.15}$$

Here the two important nondimensional parameters are the Rayleigh number Ra and the Prandtl number Pr are defined as

$$\mathrm{Ra} = \alpha g\Delta d^3/\nu\kappa \tag{1.16}$$

$$\mathrm{Pr} = \nu/\kappa. \tag{1.17}$$

We perform our simulations in a cubic fluid domain of unit size in each direction.

For the scaling analysis, we solve the RBC equations (Eqs. (1.14, 1.15)) with `FFF` and `SFF` basis functions. We also perform a production run for computing the energy spectrum and energy flux during the statistical steady state. Note that the `SFF` basis function corresponds to the free-slip boundary condition for which the velocity field at the top and bottom plates ($z = 0, 1$):

$$u_z = \partial_z u_x = \partial_z u_y = 0, \tag{1.18}$$

and periodic boundary conditions imposed on the vertical side walls. For the temper-

ature field, we employ isothermal boundary condition ($\theta = 0$) at the top and bottom plates, and periodic boundary conditions at the side walls.

Many fluid flow simulations, especially Rayleigh-Bénard convection, employ no-slip boundary condition that requires Chebyshev basis functions (`ChFF`), which are somewhat complex to implement. Note however that the `ChFF` basis has its own limitations. For example, the energy spectrum and flux computations performed in the Fourier space requires a uniform grid. The collocation points in `ChFF` are nonuniform, and hence we need to interpolate the data to a uniform mesh that induces errors. Therefore, the free-slip basis functions that involves uniform mesh are convenient for studying the energy spectrum and flux. We remark the no-slip boundary condition captures the boundary layers near the walls. The flow near the walls contributes to the energy spectrum at small scales or large wavenumbers, therefore the boundary layers at the top and bottom walls do not significantly impact the inertial-range energy spectrum and flux. For example, for turbulent thermal convection, Kumar et al. [45], and Verma et al. [98] observed Kolmogorov-like energy spectrum for free-slip basis function, while Kumar and Verma [46] observe the same spectrum for no-slip basis. See Verma [91] for more details. For such studies, the free-slip basis suffices.

The above example illustrates that we can easily perform simulations with different boundary conditions using Tarang solvers just by changing the basis function in the input file. We report our results for the RBC solvers in Chapter 6.

1.3 Complexity of turbulence simulations

Flows become turbulent when the Reynolds number becomes large. But, the transition Reynolds number depends on the system. For example, for pipe flows, transition to turbulence occurs at around Re $\sim$ 2000. For the simulation of turbulence flows, the computation turns out to be very expensive. It can be shown that the grid size increases with Re as $N \propto \mathrm{Re}^{3/4}$ [47]. For instance, for Re as high as $\approx 10^8$, the required grid size is $\approx 10^6$, while that for Re $= 10^5$ requires grid size of 6000^3. Simulation with such a large number of grid points requires a large amount of memory and consumes a lot of time resource. For a typical simulation of this size, it may take months or even years to get good results. To reduce the simulation time, a highly parallel and efficient code is essential to resolve the relevant scales of the problem. For spectral codes, we need to optimize the Fourier transforms, which is the topic of the next section.

1.4 Parallel Fourier Transforms: present status

The Fast Fourier Transform (FFT), first ideated by Cooley and Tukey [17]1, is an important tool for image and signal processing, and radio astronomy. Using FFT one can solve partial differential equations, fluid flows, density functional theory, many-body theory, etc. For a three-dimensional N^3 grid, FFT has large time complexity $O(N^3 \log N)$ for large N (e.g. 4096 or 8192). Hence, a parallel algorithm becomes necessary to compute FFT of large grids.

FFTW (Fastest Fourier Transform in the West) is one of the most popular open-source FFT libraries in which a three-dimensional (3D) array is divided into slabs (hence called *slab decomposition*). In this decomposition, we can employ a maximum of N cores in FFT operations on an array of size N^3. This is a severe limitation since present-day supercomputers offer several hundreds of thousands of cores for use. To overcome this issue, Pekurovsky [63] employed a *pencil decomposition* in which the data is divided into pencils. This method allows usage of a maximum of N^2 cores, equal to the maximum number of pencils.

The most well-known pencil-based FFT library is P3DFFT, written by Pekurovsky [63]. Pekurovsky computed that the total time T for a FFT operation as the sum of computation time a/p and communication time $b/p^{2/3}$, where p is the number of cores. This scaling was deduced based on runs using a grid of 8192^3 points on a Cray XT5 with a 3D torus interconnect, and 65536 cores. In these tests, the communication time dominates the computation time due to the `MPI_Alltoall` data transfer. Chan *et al.* [11] studied scaling of P3DFFT on a 16384 nodes of Blue Gene/L system; they reported that a combination of the network topology and the communication pattern of P3DFFT can affect performance.

Pippig and Potts [65] devised a similar FFT named PFFT, and ran it on a large number of cores; they observed that PFFT has a similar scaling as P3DFFT. Czechowski *et al.* [18] analyzed the memory hierarchy traffic and network communication in GPU-based FFT, DiGPUFFT. Mininni *et al.* [57] employed hybrid scheme (MPI + OPENMP) to use large number of cores optimally; their FFT implementations scales well on 1536^3 and 3072^3 grids for approximately 20000 cores with 6 and 12 threads on each core.

We have designed another pencil-based FFT called FFTK (FFT Kanpur) and tested it on 65536 cores of Blue Gene/P (Shaheen I) and 196608 cores of Cray XC40 (Shaheen

[1] It is worth making a remark that historically Gauss was first to discover an algorithm similar to Cooley and Tukey; it is however unpublished in his lifetime.

II) of KAUST [12]. We measure separately the time required for the computation and communications during the FFT process and show that the computation time scales linearly, while the communication component approaches the ideal bisection bandwidth scaling for large arrays. Note that the bisection bandwidth is defined as the net bandwidth available between bisected partitions of the network. It is essentially proportional to the area in the network topology. In this thesis, we present the performance of FFTK on Blue Gene/P and Cray XC40, and show that the relative speed of cores and switch matters for the efficiency. We show later that the per-core efficiency of Cray XC40 is lower than that of Blue Gene/P because the speed of the interconnect of Cray XC40 has not increased in proportion to the increase in speed of the processor. FFTK library is available for download at http://www.turbulencehub.org/codes/fftk. In this thesis, we describe the scaling results of FFTK library in Chapter 3. We remark that FFTK is as efficient as P3DFFT, yet it offers transforms for a more general set of basis functions. We detail them in the introduction of Chapter 3.

In the next section, we describe the present status of large-scale spectral solvers.

1.5 Large-scale parallel spectral flows solvers: present status

As described in Section 1.3, turbulent flows with large Reynolds numbers require large memory and computation time. Hence such computations are performed on large high performance computing clusters (HPC).

Many researchers [107, 38, 29, 105, 74, 30, 28, 26, 103, 104, 73, 20] have performed high resolution turbulence simulations. Yokokawa *et al.* [107] performed first turbulence simulation on 4096^3 grid using the *Earth Simulator*. Donzis *et al.* [29] performed turbulence simulation on 4096^3 grid using P3DFFT library; they employed 32768 cores of Blue Gene/L and Cray XT4, and reported that the effective performance of the FFT is approximately 5% of the peak performance due to the extensive communication and cache misses. Yeung *et al.* [105] and Clay *et al.* [16] performed pseudo spectral simulations of fluid turbulence on one of the highest resolution grids (8192^3) to study extreme events. Biferale *et al.* [3] simulated rotating turbulence on a 4096^3 grid and studied its energy spectrum. Rosenberg *et al.* [74] simulated rotating stratified turbulence on a 4096^3 grid. Scheel and Schumacher [78] simulated Rayleigh-Bénard Convection using a spectral-element code Nek5000 with 6.27 million element and 13th order polynomials

within each element.

Finite difference scheme is also used for turbulence simulations. For example, van der Poel et al. [87] employed this scheme to simulate turbulent wall-bounded flow on grids up to 4096^3. They appear to observe perfect strong and weak scaling for their simulation up to large number of processors due lower amount of data communication. A detailed comparison of efficiency between pseudo-spectral and finite difference codes will be useful.

1.6 Structured spectral solver: Tarang

Tarang is a pseudo-spectral object-oriented parallel code that can simulate flows in fluids, convections, magneto-fluids, magneto-convection, etc. The convective flow module can also solve a stratified flow, passive scalar and Rayleigh-Taylor instability, etc. We designed Tarang in an object-oriented fashion for generality and easy adaptability to solve various problems. We have put all boundary condition related functions in virtual functions of C++ so that subsequent calculations become independent of boundary condition.

Tarang follows the standard procedure of pseudo-spectral method [6, 9]. The Navier-Stokes and related equations are solved numerically with an initial condition of the fields. The fields are time-stepped using one of Euler, Runge-Kutta second and fourth order (RK2 or RK4). The nonlinear terms, e.g. $\mathbf{u} \cdot \nabla \mathbf{u}$, transform to convolutions in the spectral space, which are very expensive to compute. Orszag devised a clever scheme to compute the convolution in an efficient manner using Fast Fourier Transforms (FFT) [6, 9]. In this scheme, the fields are transformed from the Fourier space to the real space, multiplied with each other, and then transformed back to the Fourier space. Note that the spectral transforms could involve Fourier functions, sines and cosines, Chebyshev polynomials, spherical harmonics, or a combination of these functions depending on the boundary conditions [6, 9].

In this thesis we describe salient features and innovations of Tarang. I have contributed significantly to the code, namely in the construction of FFTK and I/O library called H5SI. These topics will be detailed in Chapter 2. We performed scaling tests of the fluid and thermal-convection solvers of Tarang. We observe that Tarang solvers scale well up to 196608 processors of Cray XC40. The scaling results are described in Chapter 4.

1.7 Hydrodynamic simulation using Tarang

We performed a large-scale simulation of hydrodynamic turbulence on 4096^3 grid using 65536 processors of Cray XC40. Using the numerical data, we compute the energy spectrum and flux for isotropic and homogeneous hydrodynamic turbulence. The most well-known theory of turbulence is by Kolmogorov [43]. According to this theory, the energy supplied at the large scales cascades to small scales. The wavenumber range dominated by forcing is called *forcing range*, while that dominated by dissipation is called *dissipative range*. The wavenumbers between these two ranges are termed as *inertial range*. According to Kolmogorov [43], the energy cascade rate or the energy flux is constant in the inertial range. Quantitatively, the one-dimensional energy spectrum $E_u(k)$ and the energy flux $\Pi_u(k)$ in the inertial range are

$$E_u(k) = K_{\mathrm{Ko}}\epsilon^{2/3}k^{-5/3}, \tag{1.19}$$

$$\Pi_u(k) = \epsilon, \tag{1.20}$$

where ϵ is the energy dissipation rate, and K_{Ko} is the universal constant. The above law has been verified using experiments and high-resolution simulations (see Frisch [33], McComb [55], Davidson [22], and references therein). There is, however, a small correction, called intermittency correction [33], to the exponent $-5/3$; this issue is however beyond the scope of this paper. Kolmogorov's theory of turbulence and its ramifications have been discussed in detail in several books [48, 55, 33, 22, 47, 67].

The above $E_u(k)$ has been generalized to the following form to model the spectrum in the dissipation range of a turbulent flow:

$$E_u(k) = K_{K_0}\epsilon^{2/3}k^{-5/3}f(k/k_d), \tag{1.21}$$

where k_d, called Kolmogorov's wavenumber, denotes the dissipation wavenumber scale. Pao [62] proposed that

$$f(x) = \exp(-x^{4/3}) \tag{1.22}$$

But, according to Pope [67]

$$f(x) \sim \exp\left\{[x^4 + c_\eta^4]^{1/4} - c_\eta\right\}, \tag{1.23}$$

where c_η is a constant.

Martínez *et al.* [54] computed $E_u(k)$ for moderate Reynolds number and showed

it to be consistent with the following energy spectrum:

$$E_u(k) \sim (k/k_d)^{\alpha} \exp[-\beta(k/k_d)], \tag{1.24}$$

Ishihara *et al.* [39] showed that near the dissipation range, Eq. (1.24) is a good approximation for high Reynolds number flows. Here, we perform numerical simulations on very high-resolution (up to 4096^3 grid), and discuss $E_u(k)$ and $\Pi_u(k)$ in detail, both in inertial and dissipative regime. We find that Pao's model is a better fit in the dissipation range.

The energy flux of a laminar flow has not been investigated in detail, either by numerical simulation or experiments. Typically $\Pi_u(k)$ in the dissipative range is assumed to be very small and it is typically ignored. Verma *et al.* [97] showed that $E_u(k) \sim \Pi_u(k) \sim \exp(-k)$. This issue will not be discussed in this thesis.

1.8 Simulation of thermal convection using Tarang

Our spectral code can solve the equations, Eqs. (1.14, 1.15), of thermal convection with horizontal thermal plates at the top and bottom, as well as using plates along all sides. At present, the velocity boundary condition implemented in Tarang is free-slip. The boundary condition for the temperature field is thermally conducting, i.e., $\theta = 0$ at the walls.

Bolgiano [4] and Obukhov [60] first gave a theory of stably-stratified flow for which energy spectrum is $E_u(k) \sim k^{-11/5}$. Later L'vov and Falkovich [53] argued that the turbulent thermal convection also follow Bolgiano and Obukhov's (BO) scaling. In this thesis, we report our results of a very high-resolution numerical simulation.

Researchers have performed a large number of numerical simulations with an aim to identify which among the two, BO or Kolmogorov-like, scaling is applicable to RBC [51]. Orszag [5] and Škandera *et al.* [80] found Kolmogorov's scaling for the velocity and temperature fields. Grossmann and Lohse [36] studied RBC for $\mathrm{Pr} = 1$ under Fourier-Weierstrass approximation and reported Kolmogorov's scaling. Based on periodic boundary condition, Borue and Kerr [41] reported the horizontal spectrum as a function of horizontal wavenumber and observed Kolmogorov's spectrum. Verzicco and Camussi [100], and Camussi and Verzicco [8] reported BO scaling using the frequency spectrum of real space probe data. Kaczorowski and Xia [40] published Kolmogorov scaling for the longitudinal velocity structure functions, but BO

scaling for the temperature structure functions in the centre of a cubical cell. Kumar *et al.* [45] computed $E_u(k)$ and $\Pi_u(k)$ in RBC and showed Kolmogorov-like behaviour, i.e., $E_u(k) \sim k^{-5/3}$ and $\Pi_u(k) \sim$ const.

We performed high-resolution simulations of Rayleigh-Bénard convection (RBC) by solving Eqs. (1.14, 1.15). The fluid is assumed to be contained in a box of unit dimension. Presently, we report the scaling results for `SFF` (free-slip boundary condition) basis functions. Note that in `SFF` basis, $u_z = \partial_z u_x = \partial_z u_y = 0$ at the top and bottom plates, and periodic along the side walls. The temperatures at the top and bottom plates are assumed to be constant (conducting walls), while at the side walls, the temperature is assumed to be periodic. For the energy spectrum and flux computations, we employ a free-slip boundary condition.

We compute the energy spectrum and show that the convective turbulence has a behavior similar to fluid turbulence. We also measure energy transfer diagnostics and show that the energy flux is constant in the inertial range, the shell-to-shell energy transfer is local and forward, and the ring-to-ring energy transfers are nearly isotropic. The ring spectrum also exhibits a near-isotropic behavior. These conclusions were possible due to the extreme resolution of our simulation. We remark that the grid resolution of our simulation is highest in the field. Also, we achieved the largest Rayleigh number among spectral codes. These results are described in Chapter 6.

In the next chapter, we describe the features of Tarang.

Chapter 2

Implementation of a pseudospectral code as general PDE solver

We encounter many versions of fluid flows, e.g. hydrodynamics, thermal convection, magnetohydrodynamics, rotating flows, liquid metal flows, etc. In a research group, we need to solve several versions of such flows. Writing different spectral codes for each of the above flows and then maintaining all of them is a tedious task. To avoid this problem, the group of Verma has developed a general-purpose partial-differential equation (PDE) solver named Tarang [93] for turbulence and instability studies. Tarang is a parallel and modular code written in the object-oriented language C++ that can solve incompressible flows involving pure fluid, Rayleigh-Bénard convection, passive and active scalars, magnetohydrodynamics, liquid metals, etc. For efficiency, we use a meta-template C++ library `blitz++` for array manipulation, `hdf5` for parallel I/O, `yaml` parser for reading parameters, and CMake for building the codes.

Tarang is an open-source code and it can be downloaded from the group website `http://turbulencehub.org`. In this chapter, we will describe some details of the code. I will also highlight my contribution to the code Tarang. The scaling results will be presented in Chapter 4.

Before detailing the features of Tarang, we describe how a spectral method is used for solving fluid flows.

2.1 Spectral method for solving turbulent flows

We illustrate the steps of a spectral method for solving incompressible fluid flow. The governing equations for such flows are the Navier-Stokes equation and the incompressibility condition:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}, \tag{2.1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2.2}$$

where $\mathbf{u}$ is the velocity field, p is the pressure field, and ν is the kinematic viscosity. We set the density ρ to be a constant, equal to unity. In Fourier space, the above equations are transformed to [47]

$$(\partial_t + \nu k^2)u_j(\mathbf{k}) = -ik_l u_l u_j(\mathbf{k}) - ik_j p(\mathbf{k}), \tag{2.3}$$

$$k_j u_j(\mathbf{k}) = 0, \tag{2.4}$$

where $\mathbf{u}(\mathbf{k})$ and $p(\mathbf{k})$ are the Fourier transforms of $\mathbf{u}(\mathbf{r})$ and $p(\mathbf{r})$ respectively.

In spectral method, the Navier-Stokes equation is solved in Fourier space. The main advantage of pseudo-spectral method is that derivative computation in this method is very accurate. For example, in central differencing scheme, two points are used for finding derivatives, but in pseudo spectral method, all the points contribute to the derivative. The $u_l u_j$ term of Eq. (2.3) is a convolution in spectral space that is very expensive to compute. Orszag [61] devised an efficient scheme in which $\mathbf{u}(\mathbf{k})$ is transformed to real space, components of which are multiplied with each other, and the product is then transformed back to Fourier space, as in Figure 2.1. Due to the multiplication of arrays in real space, this method is called *pseudospectral method*. This multiplication however generates aliasing errors, which are mitigated by filling up only 2/3 of the array in each direction. See Boyd [6] and Canuto [9] for details.

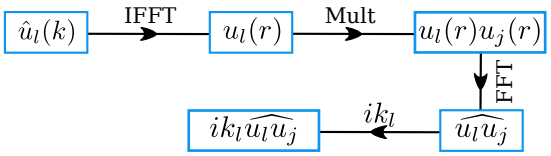


FIGURE 2.1: A schematic diagram of the computation of Fourier transform of the nonlinear term $(\mathbf{u}\cdot\nabla)\mathbf{u}$ in a pseudo-spectral solver.

The main steps of a pseudo-spectral method are as follows:

1. We compute the nonlinear terms $(\mathbf{u}\cdot\nabla)\mathbf{u}$ in real space using following steps (see Fig. 2.1):

   (a) We compute $\mathbf{u}(\mathbf{r})$ by performing inverse Fourier transforms of $\mathbf{u}(\mathbf{k})$ using the Fast Fourier Transform (FFT) [93].

   (b) We compute the products $u_i(\mathbf{r})u_j(\mathbf{r})$ at each grid point in real space. Here $i,j \in (x,y,z)$.

   (c) For incompressible flows, we compute the nonlinear terms using $ik_j\mathrm{FT}[u_i(\mathbf{r})u_j(\mathbf{r})]$ for $i,j \in (x,y)$, where FT denotes Fourier Transform.

2. In computation of the nonlinear terms, wavenumber of some generated modes is larger than the maximum wavenumber, $k_{\max}$, and therefore they introduce errors in computation. We use 2/3 dealiasing rule to remove this error, in which we set the coefficients of the last one-third Fourier modes to zero before computing the nonlinear terms, and consider only the first two-third modes.

3. Using the incompressibility condition ($\mathbf{k}\cdot\mathbf{u}(\mathbf{k}) = 0$), we compute the pressure field as
$$\sigma(\mathbf{k}) = \frac{i}{k^2}[\mathbf{k}\cdot\mathbf{N}_u(\mathbf{k})], \tag{2.5}$$
where $\mathbf{N}_u(\mathbf{k})$ is the Fourier transform of the nonlinear term $\mathbf{u}\cdot\nabla\mathbf{u}$ that is computed following the previous step.

4. Having computed each term of Eqs. (2.3) except the $\partial\mathbf{u}(\mathbf{k})/\partial t$, we typically evolve the fields $\mathbf{u}(\mathbf{k})$ in time using the fourth order Runge-Kutta (RK4) method. Our code however has options of Euler and RK2 schemes as well. The time step dt is computed using the Courant-Friedrichs-Lewy (CFL) condition.

5. The aforementioned steps are repeated until a steady state is reached.

6. In the turbulent regime, the two relevant time scales, the large-eddy turnover time and the small-scale viscous time, are very different (order of magnitude apart). To handle this, we have incorporated the "exponential trick" that absorbs the viscous term using a change of variable [10].

7. The code provides an option for dealiasing the fields. The 3/2 rule is used for dealiasing [10].

8. The wavenumber components k_i are
$$k_i = \frac{2\pi}{L_i}n_i \tag{2.6}$$

where L_i is the box dimension in the i-th direction, and n_i is an integer. We use parameters

$$\text{kfactor}_i = \frac{2\pi}{L_i} \tag{2.7}$$

This feature is to control the box size, especially for Rayleigh-Bénard convection. Note that typical spectral codes take $\text{kfactor}_i = 1$, or $k_i = n_i$.

2.2 Design and implementation issues of Tarang

In Tarang, we use the object-oriented features of C++ to design general purpose libraries to create solvers for the incompressible fluid flows. We use classes to abstract fluid variables as vector and scalar fields. We have designed general functions that act on these fields. For example, we construct function `compute_nlin(FluidVF& U)` to compute $\mathbf{U} \cdot \nabla \mathbf{U}$, while `compute_nlin(FluidVF& U, FluidSF& T)` to compute $\mathbf{U} \cdot \nabla \mathbf{U}$ and $\mathbf{U} \cdot \nabla T$; here the classes `FluidVF` and `FluidSF` represent vector and scalar fields respectively. The above design helps us solve various systems like magnetohydrodynamics, fluid, convection, and rotating turbulence [93].

A solver containing many complex features including boundary conditions and more field variables are easily constructed using these functions. Main features including the class structures of Tarang are described below.

2.2.1 Basis functions

A pseudo-spectral code uses basis functions to expand the fields in Fourier-space. The choice of basis functions depends critically on the boundary conditions. Complex-exponential functions such as $\exp(i\mathbf{k} \cdot \mathbf{r})$, where $\mathbf{k}$ and $\mathbf{r}$ are the wavenumber and real-space coordinates respectively, are natural choices for periodic boundary conditions. Many problems in turbulence however involve walls, e.g. convection, channel flows, etc. All the components of the velocity field must vanish at the wall for no-slip boundary condition. Chebyshev and Legendre polynomials are often used as basis functions for no-slip boundary conditions. For free-slip boundary condition, the normal component of the velocity field to the wall, and the perpendicular gradient of the horizontal velocity components are zero. Sine and cosine functions are simple choices for simulations involving free-slip boundaries. Similarly, spherical harmonics are also natural choices as basis functions for turbulence simulations in spheres.

Note that a general inverse Fourier transform is defined as

$$f(x,y,z) = \sum_{k_x,k_y,k_z} f(k_x,k_y,k_z)\phi_{k_x}(x)\phi_{k_y}(y)\phi_{k_z}(z) \tag{2.8}$$

where $f(k_x,k_y,k_z)$ is the Fourier transform of $f(x,y,z)$. Most commonly used basis functions $\phi_k(s)$ are:

$$\text{Fourier}: \quad \phi_k(s) = \exp(iks), \tag{2.9a}$$

$$\text{Sine}: \quad \phi_k(s) = 2\sin(ks), \tag{2.9b}$$

$$\text{Cosine}: \quad \phi_k(s) = 2\cos(ks), \tag{2.9c}$$

where k could be k_x, k_y, or k_z, and s could be x, y, or z. For instance, an RBC simulation with free-slip boundary condition at the top and bottom plates, and periodic boundary condition along the horizontal is implemented using

$$\phi(x)\phi(y)\phi(z) = \exp(ik_x x + ik_y y)[\sin(k_z z), \cos(k_z z)] \tag{2.10}$$

basis functions. We term the above basis function as `SFF` (Sin/Cos Fourier Fourier). We define other basis functions in a similar manner.

Tarang focuses on basis-independent libraries, so we have designed the basis functions in a modular fashion. For this purpose we use `virtual` functions of C++. Virtual functions can be overridden by derived classes. We have a `Universal` class that has many virtual functions for operations common to all basis but their implementation is basis dependent.

At present Tarang has four basis functions that can simulate flows in box geometry — FFF: under periodic boundary conditions along all directions, SFF: periodic along two directions and walls along one direction, SSF: periodic along one direction and walls along two directions, and SSS: walls along all directions. We have designed a class UNIVERSAL containing various virtual classes that are overridden in various basis classes i.e. FFF, SFF, SSF, and SSS. The advantage of this implementation is that fluid solvers are independent of basis functions. *Universal function and structures were designed and implemented by me.*

Primary functions related to the basis functions are *forward_transform* (transformation from the real space to the Fourier space), *inverse_transform* (transformation from the Fourier space to the real space), computation of energy spectrum etc. We use the FFTK library for majority of transform operations.

In the next section, we describe various important classes of Tarang.

2.3 Classes of Tarang

The class structure of Tarang is illustrated in Fig. 2.2. We describe the main features of the classes below.

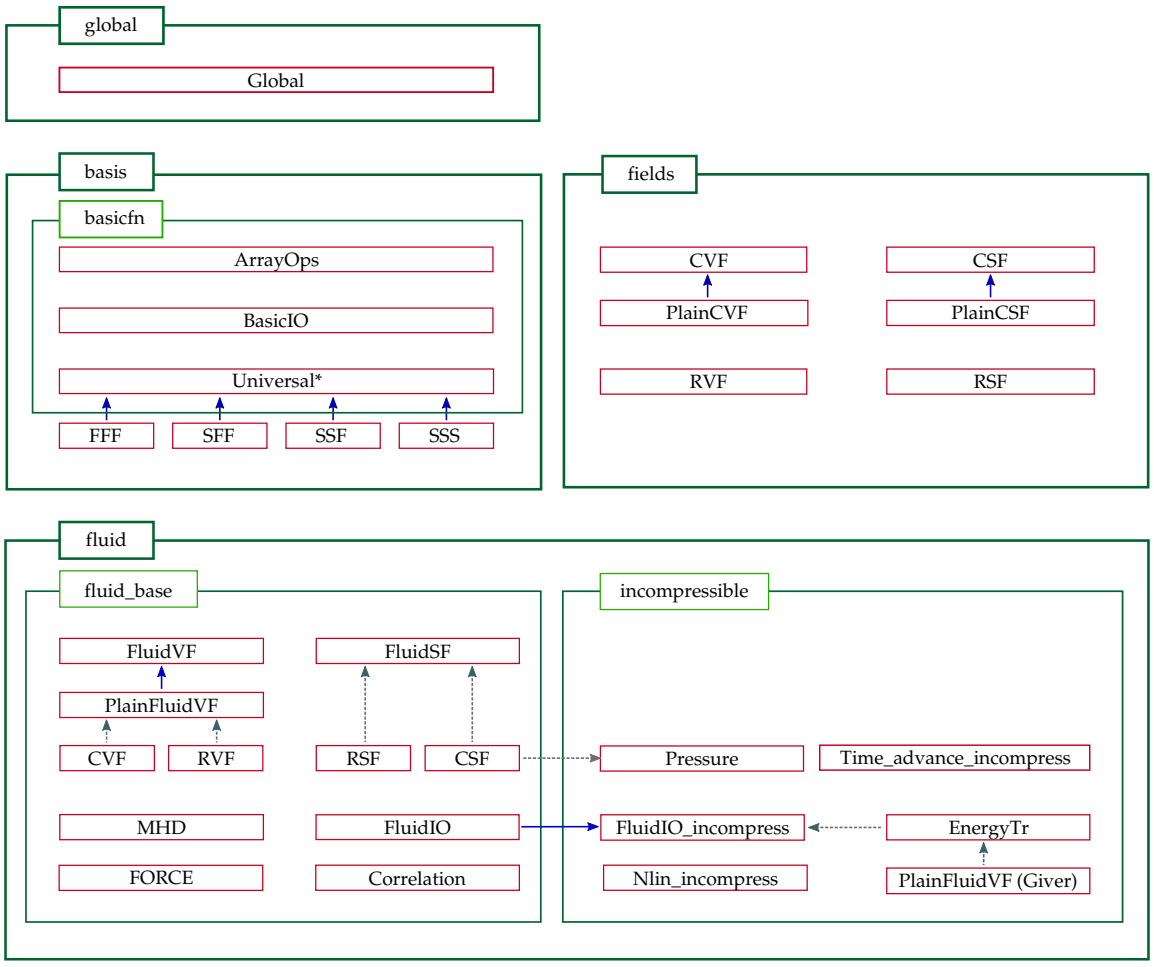


FIGURE 2.2: Organisation of various classes of Tarang. Green outer boxes represent folder structures. Red inner boxes represent classes. Blue solid arrows represent is-a relation between classes (top class is derived from the bottom class) and gray dashed arrows represent has-a relation between classes (top class has an instance of the bottom class). The (*) represents that this class contains virtual functions and hence cannot be instantiated without inheriting and defining virtual functions.

Global

The instance of Global class is a globally available object that includes all the input parameters, MPI coordinates and parameters, a set of secondary parameters (those derived from input parameters) such as `min_radius_outside` of shells, and a few temporary arrays. It also includes the `para.yaml` reader. This class resides in directory `global`.

Basis

The following two classes defined within the folder `basicfn` contain some basic array operations that are required throughout the program.

ArrayOps is an acronym for *Array Operations*. This class contains functions for multiplying and dividing arrays in point-by-point fashion so that they can be threaded easily.

BasicIO is acronym for *Basic Input Output* in the interface of Tarang, and `H5SI` for reading/writing HDF5 data.

The following classes contain boundary condition dependent functions. These classes reside in directory `basis`.

Universal class contains all boundary condition dependent functions. All functions in Universal class are `virtual` functions. These functions can be overridden by derived classes. There are some functions which have same definition for all boundary conditions, such as `Get_total_energy`. `Get_total_energy` calls `Get_local_energy` which has different definitions for various boundary condition and uses `MPI_reduce` to get the total energy. The instance of this class is a pointer to one of **FFF**, **SFF**, **SSF**, or **SSS** depending on `basis_type` in `para.yaml`.

FFF is acronym for *Fourier Fourier Fourier* and represents periodic boundary condition along all axes. Since the value of a mode in a periodic boundary can be anything at the boundary, they are best-represented by exponential modes. This class is derived from the **Universal** class.

SFF, acronym for *Sin/Cos Fourier Fourier*, represents periodic boundary condition along Y-axis and Z-axis and wall along X-axis. This class is derived from the **Universal** class.

SSF, acronym for *Sin/Cos Sin/Cos Fourier*, represents periodic boundary condition along Z-axis and wall along X-axis and Y-axis. This class is derived from the **Universal** class.

SSS, acronym for *Sin/Cos Sin/Cos Sin/Cos*, represents wall along all axes. This class is derived from the **Universal** class.

```
void Universal::Compute_divergence(Array<Complex,3> Ax, Array
   <Complex,3> Ay, Array<Complex,3> Az, Array<Complex,3> div,
    string field_or_nlin, Real &max_abs_div, bool
   print_switch)
{
  global.program.sincostr_switch = sincostr_switch_Vx;
  Xderiv(Ax, div);

  global.program.sincostr_switch = sincostr_switch_Vy;
  Add_Yderiv(Ay, div);

  global.program.sincostr_switch = sincostr_switch_Vz;
  Add_Zderiv(Az, div);

  if (field_or_nlin == "field") {
    max_abs_div = max(abs(div));

    if ((print_switch) && (max_abs_div > MYEPS2)) {
      if (master)
      cout << "NON-ZERO DIVERGENCE for the following modes:"
          << endl;
      Print_large_Fourier_elements(div, "Divergence");
    }
  }
}
```

LISTING 2.1: Function Compute_divergence

```
void FFF_PENCIL::Xderiv(Array<Complex,3> A, Array<Complex,3>
   B)
{
  Real Kx;
  for (int lx=0; lx<maxlx; lx++) {
    Kx = Get_kx(lx)*kfactor[1];
    B(lx,Range::all(),Range::all()) = Complex(0,Kx)*A(lx,
       Range::all(),Range::all());
  }
}

void FFF_PENCIL::Add_Yderiv(Array<Complex,3> A, Array<Complex
   ,3> B)
{
  Real Ky;
  for (int ly=0; ly<maxly; ly++) {
    Ky = Get_ky(ly)*kfactor[2];
    B(Range::all(),ly,Range::all()) += Complex(0,Ky)*A(Range
       ::all(),ly,Range::all());
  }
}

void FFF_PENCIL::Add_Zderiv(Array<Complex,3> A, Array<Complex
   ,3> B)
{
  Real Kz;
  for (int lz=0; lz<maxlz; lz++) {
    Kz = Get_kz(lz)*kfactor[3];
    B(Range::all(),Range::all(),lz) += Complex(0,Kz)*A(Range
       ::all(),Range::all(),lz);
  }
}
```

LISTING 2.2: Derivative functions

Fields

These are the Vector/Scalar fields based on which Vector/Scalar Fluid and Pressure are created. These classes reside in directory `fields`.

CVF stands for *Complex Vector Field*. It contains three dynamic arrays associated with the three components of the velocity or magnetic fields in the Fourier space. Size of each array is $N_x \times N_y \times N_z/2+1$ that spans wavenumbers $(k_x, k_y, k_z) = [-N_x/2 : N_x/2-1, -N_y/2 : N_y/2-1, 0 : N_z/2-1]$ in FFF basis. Size of the array is same in **SFF**, and **SSF** basis whereas the wave number span for **SSS** is $(k_x, k_y, k_z) = [0 : N_x - 1, -N_y/2 : N_y/2-1, 0 : N_z/2]$ and $(k_x, k_y, k_z) = [0 : N_x - 1, 0 : N_y - 1, 0 : N_z/2-1]$ in **SSS** basis. These arrays contain complex numbers that represent the Fourier amplitudes of the vector field. In **SSS** basis, the complex numbers contain real values of two wavenumbers. These arrays are created dynamically at run-time.

RVF stands for *Real Vector Field*, and it contains three dynamic arrays to represent the vector fields in real space. These arrays are of size $N_x, N_y, N_z + 2$.

CSF stands for *Complex Scalar Field*. It contains a dynamic array associated with a scalar field, e.g. temperature, in the Fourier space. The indexing of the array is similar to that of **CVF**.

RSF stands for *Real Scalar Field*, and it contains a dynamic array associated with a scalar. The array features are same as that for **RVF**. Forward and inverse transforms, and input/output of vector fields are some of the main functions of these four classes – `CVF`, `RVF`, `CSF`, and `RSF` that in turn call corresponding functions of **Universal**.

Fluid Base Classes

These classes store and compute various aspects of a fluid that are independent of compressibility. These classes reside in directory `fluid/fluid_base`.

PlainFluidVF stands for *Plain Fluid Vector Field*. It contains an instance of **CVF** (*Complex Vector Field*) and **RVF** (*Real Vector Field*). **CVF** is used to store the fields in Fourier space, and **RVF** is used to store them in real space. Forward and inverse transforms are the main functions of the class *PlainFluidVF*. These functions in turn call those of `CVF` class and `RVF` class.

FluidVF stands for *Fluid Vector Field* and it inherits `PlainFluidVF`. It contains

three dynamic arrays associated with the three components of the non-linear term and force terms in Fourier space. The size of these arrays are same as those in **CVF**. These arrays contain complex numbers that represent the Fourier amplitudes of the vector field. This class also stores the **dissipation_coefficient** and **hyper_dissipation_coefficient** of the associated fluid. **Compute_divergence** and **Get_dt**, are some of the main functions of the class **FluidVF**.

FluidSF stands for *Fluid Scalar Field*. It contains an instance of **CSF** and **RSF**. **CSF** is used to store a scalar field in Fourier space and **RSF** is used to store that in real space. It also contains two dynamic arrays associated with one component of the non-linear term and one for the force term in Fourier space. For properties of the fluid, it stores **dissipation_coefficient**, **hyper_dissipation_coefficient** and **hyper_dissipation_exponent**. **Forward_transform**, **Inverse_transform** and **Get_dt**, are some of the main functions of the class **FluidSF**.

The **Correlation** class can compute autocorrelations and cross correlations. For example, in fluid simulations, the function Compute_shell_spectrum computes the autocorrelation of velocity field, whereas the same function computes cross correlation of velocity field and magnetic field for an MHD simulation. This class contains correlation functions for shell spectrum, ring spectrum, cylindrical ring spectrum, some helicity computation functions, etc.

Force has functions to compute various kind of forces such as random forcing, RBC forcing, Coriolis forcing etc.

MHD has functions that are specific to MHD. Presently they compute Elsasser field, Elsasser force, and Elsasser nlin.

FluidIO stands for *Fluid Input Output*. It has handler for all files, functions of various initial conditions and printing data to output files. It computes various parameters during the printing process such as total energy, total helicity, etc.

2.3.1 Fluid Incompressible Classes

These classes are related to incompressible fluid simulations. These classes reside in directory `fluid/incompressible`.

Nlin_incompress computes the non-linear term for various kinds of simulations such as Fluid, RBC, MHD assuming Boussinesq approximation for incompressible fluid.

EnergyTr stands for *Energy Transfer*. This contains functions to compute shell-to-shell, and ring-to-ring (spherical and cylindrical) energy transfers. This class contains an instance of **PlainFluidVF** as `Giver`. It depends on **Nlin_incompress**. More details on energy transfers can be found in Sec. 2.7.

Time_advance_incompress contains functions necessary for time stepping such as Euler and Runge-Kutta integration schemes. It also contains functions to evaluate various terms of Fluid, RBC and MHD equations and add them.

FluidIO_incompress contains functions to output Energy transfers and pressure. This class inherits **FluidIO**.

Pressure contains functions to compute pressure. This class inherits **CSF**.

Nlin_incompress class has many nlin computation functions. One of them is *Compute_nlin*. We will describe this function as an illustration of a Tarang function:

```
void Nlin_incompress::Compute_nlin()
{

...

//Perform Inverse Transform of field in U.cvf and store it in
    U.rvf
U.Inverse_transform();

//Compute u_i.k_i u
Compute_nlin_diag(U);

//Compute u_i.k_j u (i!=j)
Compute_nlin_offdiag(U);
```

```
...

}
```

LISTING 2.3: Function Compute_nlin

The comments above the C++ statements explain the logic of the functions. There are related functions that compute the nonlinear term in the presence of scalar field and another vector field.

2.4 Solvers of Tarang

We invoke the library functions discussed above to create solvers for fluid, magneto-hydrodynamics, passive scalar, RBC flows etc. We illustrate a code segment containing the time-loop of fluid solver for an illustration.

```
//  A code segment of the fluid solver
// Read initial condition

FluidVF U(...);   // Arguments have parameter from yaml file
FluidVF helicalU("helicalU");
Pressure P;
FORCE  Force;
Time_advance_incompress  time_advance_incompress;

fluidIO_incompress.Read_init_cond(U);

int  iter=0;  // iterations

do
{
iter++;
time_advance_incompress.Time_advance_step(U, P, Force);
fluidIO_incompress.Output_all_inloop(U, P, helicalU);
}
while (global.time.now < global.time.final);
```

LISTING 2.4: Fluid Solver

In the above code segment, U is an instantiation of the class *FluidVF* which contains the incompressible velocity field.

For RBC, the above code segment is modified slightly. We create an instantiation T of the class *FluidSF* to represent the temperature field.

```
// A code segment of the RBC solver

...

FluidVF  U(...);  // Arguments have parameter from yaml file
FluidVF helicalU("helicalU");
FluidSF T(...);   // Arguments have parameter from yaml file
Pressure P;
FORCE  Force;
Time_advance_incompress  time_advance_incompress;

// Read initial condition
fluidIO_incompress.Read_init_cond(U, T);

int  iter=0;  // iterations
do {
iter++;
time_advance_incompress.Time_advance_step(U,T,  P, Force);
fluidIO_incompress.Output_all_inloop(U, T, P, helicalU);
}
while (global.time.now < global.time.final);
```

LISTING 2.5: RBC solver

2.5 External libraries

The parameter file of Tarang is written in YAML format. We use YAML-CPP library to read this file. Using this format, a program can easily read a numeric, a string, or an array from a text file.

The handling of arrays and mathematical functions is quite inefficient in C++. Fortunately, several efficient C++ libraries are available to perform these tasks. We use *Blitz++* for Tarang since it handles multidimensional arrays in a very nice and compact manner. The other libraries that have similar functions are *boost, ndarrays*, and *eigen*, but presently we continue our development with Blitz++.

Pseudo-spectral codes use FFT (Fast Fourier Transforms) heavily. In a typical pseudo -spectral code, approximately 70-80% of the total computational time is spent in performing FFTs. We have therefore developed a home made library FFTK that uses FFTW for 1D transforms. Tarang uses FFTK in its implementation.

Performing input-output (IO) of data fields using ASCII (American Standard Code for Information Interchange) format consumes more space, and it is less precise as well. Therefore we store our data in HDF5 (Hierarchical Data Format 5) format. It uses MPI-IO in the underlying layers. Writing parallel IO using HDF5 is quite involved and therefore, we have developed a new library, H5SI (HDF5 Simple Interface) on top of the HDF5 library. The API (Application Programming Interface) of H5SI is similar to that of `h5py` (Python interface of HDF5).

2.6 The H5SI Library

The H5SI library is a thin layer on top of the HDF5 library. Its purpose is to simplify the IO API, especially in parallel operations. The parallel API of HDF5 is quite involved and cumbersome. Thus we have developed H5SI. The API of H5SI is highly inspired from h5py of Python and is very much similar to it. To compile the parallel API, H5SI_ENABLE_MPI flag must be supplied to it during compilation.

To open a file in sequential mode for writing:

```
h5::File file;
file.open("foo.h5", "w");
```

LISTING 2.6: Open a file for writing

The open function supports the following modes:

r Read-only, file must exist.

r+ Read/write, file must exist.

w Create file, truncate if exists.

w- Create file, fail if exists.

a Read/write if exists, create otherwise.

By default it uses the SEC2 driver; this is the default driver of HDF5 which uses Posix file-system functions like read and write to perform I/O to a single file.

To use other drivers they must be initialized before opening the file, such as,

```
h5::File file;
file.mpiioInit(MPI_COMM_WORLD);
file.open("foo.h5", "w");
```

LISTING 2.7: Open a file in parallel defaults to slab division

The following drivers are supported by H5SI library.

CORE: coreInit(size_t increment, hbool_t backing_store)

This enables an application to work with a file in memory, speeding up the read and write operations as no disk access is made. File contents are stored only in memory until the file is closed. The `backing_store` parameter determines whether file contents are ever written to disk.

The argument `increment` specifies the increment by which allocated memory is to be increased each time more memory is required.

If the `backing_store` is set to 1 (TRUE), the file contents are flushed to a file with the same name as this core file when the file is closed or access to the file is terminated in memory.

If there is an existing file with HDF5_CORE driver in Read/Write mode with the `backing_store` set to 1, any change to the file contents are saved to the disk when the file is closed. If it is opened in Read/Write mode with `backing_store` set to 0, any change to the file contents will be lost when the file is closed. If it is opened in Read only mode, no change to the file is allowed either in memory or on disk.

Returns a non-negative value if successful. Otherwise returns a negative value.

Note: Currently this driver cannot create or open family of multiple files.

STDIO: `stdioInit()`

Buffered I/O using functions from stdio.h.

Returns a non-negative value if successful. Otherwise returns a negative value.

MPI-IO: `mpiioInit(MPI_Comm MPI_COMMUNICATOR)`

Stores MPI IO communicator information and some optimization parameters to the File object.

This function is available only when compiled with the parallel HDF5 library (by passing `H5SI_ENABLE_MPI` during compilation) and is not a collective function.

`MPI_COMMUNICATOR` is the MPI communicator to be used for opening the file.

1. This function does not create a duplicated communicator. Modifications to the communicator after this function call returns may have an undetermined effect on the access property list. Users should not modify the communicator while it is defined in a property list.
2. Raw dataset chunk caching is not currently supported when using this file driver in read/write mode. All IO operations will access the disk directly, and `H5Pset_cache` and `H5Pset_chunk_cache` will have no effect on performance. Raw dataset chunk caching is supported when this driver is used in read-only mode.

Returns a non-negative value if successful. Otherwise returns a negative value.

2.6.1 Group

A group is like a folder that can store more groups or dataset (an array of data). To create a new Group the following functions may be used

```
file.createGroup("group_name");
```

LISTING 2.8: Creating new group

createGroup fails if the group already exists. To open a group and create one if it does not exist, requireGroup must be used. It also will fail if the group already exists.

```
file.requireGroup("group_name");
```

LISTING 2.9: Creating new group

To open an existing group for reading, file object is used as a dictionary.

```
h5si::Group group = file["group_name"];
```

LISTING 2.10: Open group for reading

2.6.2 Dataset

A Dataset is like a file where the data is stored. Dataset creation is described in below

```
h5::Dataset ds = file.create_dataset("dataset_name", h5::
   shape(512,512,512), "double");
```

LISTING 2.11: Dataset creation

For datatypes, refer to Appendix 2.

In the following Listings, we illustrate the usage of above-mentioned functions.

```
void write_array() {
  Array<Complex,3> A(32,64,64);

  init_array(A);

  h5::File f;
  f.mpiioInit(MPI_COMM_WORLD);
  f.open("foo.h5", "w");

  //Method1
  h5::Dataset ds1 = f.create_dataset("U.V1", h5::shape
     (64,64,64), H5Complex);
  ds1 << A.data();

  //Method 2
  h5::Plan plan;
  plan.set_plan(MPI_COMM_WORLD, h5::shape(64,64,64), h5::
     Select::all(3), h5::shape(64,64,64), h5::Select::all
     (3), h5::Dtype(H5Complex));
  h5::Dataset ds2 = f.create_dataset("U.V2", plan);
  ds2 << A.data();
}
```

LISTING 2.12: Sample h5si writer

```
void read_array() {
  Array<Complex,3> A(32,64,64);

  h5::File f("foo.h5", "r");
  f.mpiioInit(MPI_COMM_WORLD);

  //Method 1
  f["U.V1"] >> A.data();
  cout << A << endl;

  //Method 2
  h5::Plan plan;
  plan.set_plan(MPI_COMM_WORLD, h5::shape(64,64,64), h5::
     Select::all(3), h5::shape(64,64,64), h5::Select::all
     (3), h5::Dtype(H5Complex));

  h5::Dataset ds = f["U.V2"];
  ds.set_plan(plan);
  ds >> A.data();
  cout << A << endl;
}
```

LISTING 2.13: Sample h5si reader

In the next section we describe the energy transfer formalism for turbulent flows.

2.7 Energy transfers in turbulent flows

In turbulent flows, the nonlinear terms trigger interactions among various scales of the flow. For example, the nonlinear terms of Navier-Stokes equation ($\mathbf{u} \cdot \nabla \mathbf{u}$) exchanges energy from one Fourier mode to another. Accurate quantification of such transfer is quite difficult. Fortunately, Fourier decomposition allows us to do this analysis as one can study structures at different scales. Here, we describe how to quantify various energy transfers among various scales of turbulence.

Due to nonlinear interactions, say a Fourier mode $\mathbf{k}$ generates two modes with

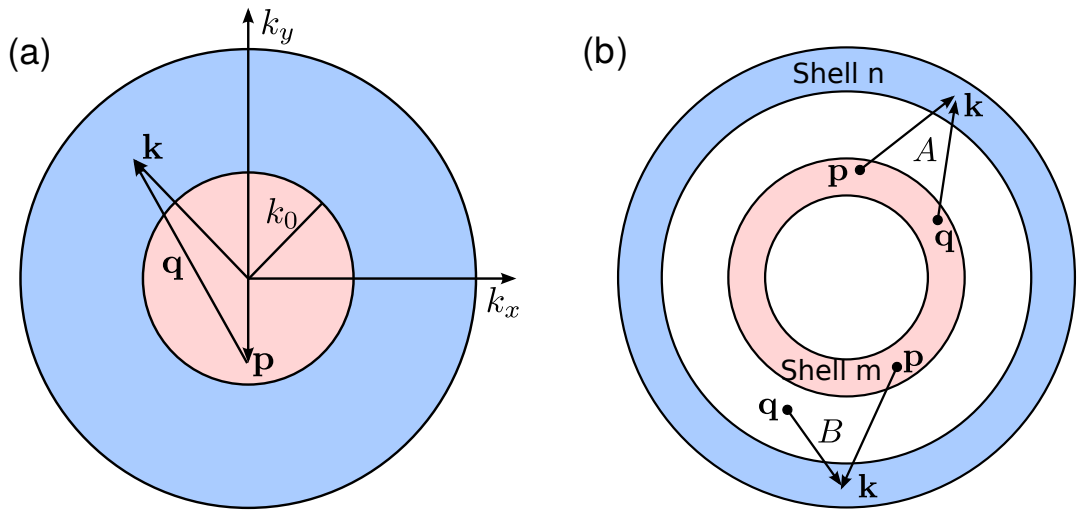


FIGURE 2.3: (a) Schematic diagram of the energy flux from a wavenumber sphere of radius k_0. The red region denotes the modes inside the sphere, and the blue region the modes outside the sphere. (b) A depiction of the shell-to-shell energy transfers from a wavenumber-shell m (red) to another wavenumber-shell n (blue).

wavenumbers, $\mathbf{p}$ and $\mathbf{q}$. The nature of nonlinearity is such that $\mathbf{k} = \mathbf{p} + \mathbf{q}$. Dar *et al.* [21] and Verma [88] derived a formula to compute the energy transfer from the Fourier mode $\mathbf{p}$ to the mode $\mathbf{k}$ with the mode $\mathbf{q}$ acting as a mediator:

$$S^{uu}(\mathbf{k}|\mathbf{p}|\mathbf{q}) = Real[-i\{\mathbf{k}\cdot\mathbf{u}(\mathbf{q})\}\{\mathbf{u}(\mathbf{p})\cdot\mathbf{u}^*(\mathbf{k})\}], \tag{2.11}$$

where Re stands for the real part of the argument. This formalism is very useful for studying energy transfers in various fluid applications, for example, field reversals in RBC and dynamo. We use it to derive energy transfers from a set of Fourier modes in a region A to another set of modes in region B of the spectral space:

$$\begin{aligned} T^{u,A}_{u,B} &= \sum_{\mathbf{k}\in B}\sum_{\mathbf{p}\in A} S^{uu}(\mathbf{k}|\mathbf{p}|\mathbf{q}) \\ &= Real\sum_{\mathbf{k}\in B}\left\{\sum_{\mathbf{p}\in A} -i[\mathbf{k}\cdot\mathbf{u}(\mathbf{q})]\mathbf{u}^A(\mathbf{p})\right\}\cdot\left\{\mathbf{u}^B(\mathbf{k})\right\}^* \\ &= Real\sum_{\mathbf{k}\in B}\{-\mathbf{N}(\mathbf{k})\}\cdot\left\{\mathbf{u}^B(\mathbf{k})\right\}^*, \end{aligned} \tag{2.12}$$

where

$$\mathbf{u}^{A,B}(\mathbf{k}) = \begin{cases} \mathbf{u}(\mathbf{k}) & \text{for } \mathbf{k} \in (A,B) \\ 0 & \text{otherwise} \end{cases}, \tag{2.13}$$

and $\mathbf{N}(\mathbf{k})$ is the nonlinear field induced by the velocity field $\mathbf{u}^A$. The sum of the term $\mathrm{Re}\left\{\mathbf{N}(\mathbf{k}) \cdot \mathbf{u}^{*B}(\mathbf{k})\right\}$ over the region B yields the energy transfer from the Fourier modes of region A to the Fourier modes of region B. We can also define energy transfer rate of entropy (θ^2) in a similar manner [45].

We have implemented the energy transfer functions in our pseudo-spectral code. The computation of the nonlinear term $\mathbf{N}(\mathbf{k})$ is the most expensive operation in the code, and it is computed using FFT as described in Fig. 2.1. We use the above technique to compute various energy transfers. The kinetic energy flux $\Pi(k_0)$, defined as the energy coming out from the wavenumber sphere of radius k_0, is computed using the following formula:

$$\Pi_u(k_0) = \sum_{k>k_0} \sum_{p\le k_0} S^{uu}\left(\mathbf{k}\,|\mathbf{p}|\,\mathbf{q}\right). \tag{2.14}$$

Here the region A is the volume within the sphere, while the region B is the region outside the sphere [see Fig. 2.3(a)].

For viewing the energy transfers in more detail, the wavenumber space is divided into a set of wavenumber shells. The energy contents of wavenumber shell of radius k and of unit width is denoted by $E(k)$. The shell-to-shell energy transfer rate from the velocity field of the mth shell to the velocity field of the nth shell [see Fig. 2.3(b)] is defined as

$$T^{u,m}_{u,n} = \sum_{\mathbf{k}\in n} \sum_{\mathbf{p}\in m} S\left(\mathbf{k}\,|\mathbf{p}|\,\mathbf{q}\right). \tag{2.15}$$

In Kolmogorov's theory of fluid turbulence, the maximum shell-to-shell energy transfer occurs from shell m to shell $(m+1)$, hence the energy transfer in fluid turbulence is *local* and *forward*. We will explore using numerical simulations whether the energy transfers in RBC is local and forward.

The energy flux and shell-to-shell energy transfers do not capture the dependence on the polar angle ζ of Fig. 2.4. Therefore they do not quantify the anisotropy of the flow. Convective flows are anisotropic due to buoyancy. Hence it is important to quantify anisotropy using quantities that are dependent on the polar angle. For this, we divide a wavenumber shell into *rings* as shown in Fig. 2.4. The energy contents of the rings is called *ring spectrum* $E(k,\beta)$, where β represents the sector index for the polar

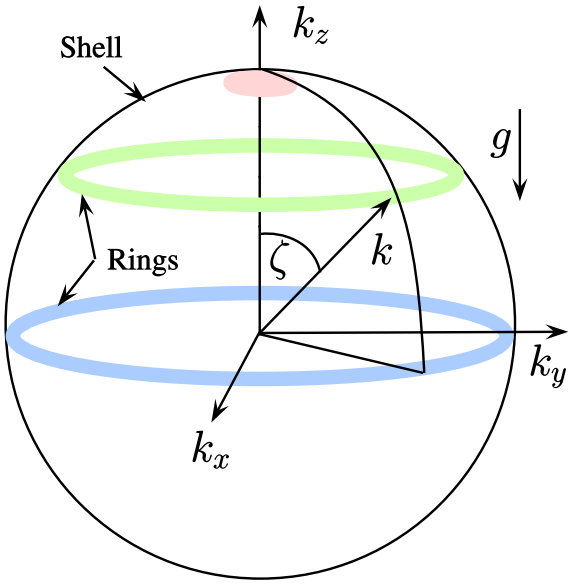


FIGURE 2.4: Schematic diagram of the ring decomposition in Fourier space.

angle. The ring-to-ring energy transfer from ring (m, α) to ring (n, β) is

$$T^{(u,n,\beta)}_{(u,m,\alpha)} = \sum_{\mathbf{k}\in(n,\beta)} \sum_{\mathbf{p}\in(m,\alpha)} S\left(\mathbf{k}\,|\mathbf{p}|\,\mathbf{q}\right). \tag{2.16}$$

These ring-to-ring energy transfer calculations for all m and n's are computationally expensive. In thermal convection, the ring spectrum was computed by Nath *et al.* [59]. Also, Teaca *et al.* [83] and Verma [90] performed similar analysis for magnetohydrodynamic turbulence and liquid metal flows respectively.

In the next subsection, we describe the class structure of EnergyTr in some detail.

2.7.1 Class EnergyTr

EnergyTr (Energy Transfer) class computes flux, helicity, magnetic helicity, enstrophy flux, and magnetic enstrophy flux. Here we present the definition of `Compute_flux` as a code demonstration.

```
void EnergyTr::Compute_flux(FluidVF &U)
{

  flux_self = 0.0;
```

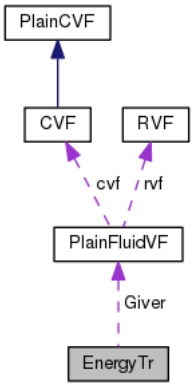


FIGURE 2.5: Inheritance diagram of EnergyTr (Energy Transfer) class. It has an instance of **PlainFluidVF** for computing **Giver** which again has instance of **RVF** and **CVF**. **CVF** inherits **PlainFluidVF**

```
  for (int sphere_index = 1; sphere_index <= global.
     energy_transfer.flux.no_spheres; sphere_index++) {
    Fill_in_sphere(sphere_index, U);

    Nlin_incompress::Compute_nlin(U, Giver);
    //  U.nlin = U.grad Giver<

    flux_self(sphere_index) = -Prod_out_sphere_nlinV(
       sphere_index, U, U);
    // flux_self = -(U.grad U<). U> = U.nlin< . U>
  }
}
```

LISTING 2.14: Compute_flux

2.8 Summary

We have developed an object-oriented code Tarang that is highly modular. Such a structure enables easy implementation of new forcing, initial-condition, etc. In subsequent chapters, we perform high-resolution fluid simulation using spectral method on Blue Gene/P and Cray XC40, i.e., we solve the Navier-Stokes equation along with the continuity equation on these systems. We assume the flow to be incompressible, and use periodic boundary condition for which FFF basis function is appropriate.

In the next chapter we describe the parallel implementation of our FFT library FFTK.

Chapter 3

Parallelisation of FFT and its scaling

The Fast Fourier Transform (FFT) is an important tool for image and signal processing, and radio astronomy. It is also used to solve partial differential equations, fluid flows, density functional theory, many-body theory, etc. Note that FFT was first discovered by Cooley and Tukey [17]. For a three-dimensional N^3 grid, FFT has large time complexity $\mathcal{O}(N^3 \log N)$ for large N (e.g. 4096 or 8192). Hence, parallel algorithms have been devised to compute FFT of large grids. In this chapter we will discuss an implementation of parallel FFT that can compute transforms on very large grids using a large number of processors. We remark that in a pseudospectral code, typically 70% to 80% of the total time is spent on the forward and inverse Fourier transforms. So, optimisation of FFT is critical for spectral code.

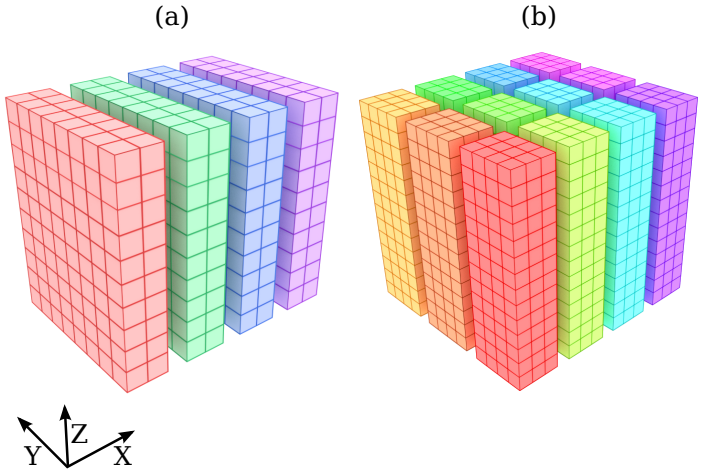


FIGURE 3.1: (a) Slab decomposition of an array. (b) Pencil decomposition of an array. Taken from Chatterjee *et al.* [12].

One of the most popular opensource FFT libraries is FFTW (Fastest Fourier Transform in the West) [32] in which a three-dimensional (3D) array is divided into slabs (hence called *slab decomposition*) as shown in Fig. 3.1(a). Hence, we can employ a maximum of N cores in FFTW operations on an array of size N^3. This is a severe limitation for present-day supercomputers that offer several hundreds of thousands of cores. For example, one of the largest clusters available today has 10,649,600 cores (National Supercomputing Center in Wuxi China). Therefore using just N cores yields suboptimal performance of the cluster.

To overcome this difficulty, Pekurovsky [64] employed a *pencil decomposition* in which the data is divided into pencils, as shown in Fig. 3.1(b). This method allows usage of a maximum of N^2 cores, equal to the maximum number of pencils. This pencil-based FFT library is called P3DFFT. In P3DFFT, the total time T for a FFT operation is a sum of computation time a/p and communication time $b/p^{2/3}$, where p is the number of cores. This scaling was deduced based on runs using a grid of 8192^3 points on a Cray XT5 with a 3D torus interconnect, and 65536 cores. Chan *et al.* [11] studied scaling of P3DFFT on a 16384 nodes of Blue Gene/L system. These tests reveal that the communication time dominates the computation time due to the `MPI_Alltoall` data transfer.

Pippig and Potts [66] devised a similar FFT named PFFT and showed that PFFT has a similar scaling as P3DFFT. Richards *et al.* [70] performed scalability analysis for their two-dimensional pencil FFT library on Blue Gene/P. Czechowski *et al.* [19] constructed a GPU-based FFT, DiGPUFFT. Mininni *et al.* [57] employed hybrid scheme (MPI + OPENMP) to use large number of cores optimally; their FFT implementations scale well on 1536^3 and 3072^3 grids for approximately 20000 cores with 6 and 12 threads on each socket.

We have devised another pencil-based FFT called FFTK (FFT Kanpur) and tested it on 65536 cores of Blue Gene/P (Shaheen I) and 196608 cores of Cray XC40 (Shaheen II) of KAUST. We will describe the scaling results of FFTK in this chapter. We computed separately the time required for the computation and communications during the FFT process, and showed that the computation time scales linearly, while the communication component approaches the ideal bisection bandwidth scaling for large arrays. We also compare the performance of FFTK on Blue Gene/P and Cray XC40, and show that the relative speed of cores and switch matters for the efficiency. We show later that the per-core efficiency of Cray XC40 is lower than that of Blue Gene/P because the speed of the interconnect of Cray XC40 has not increased in commensurate with the speed of the processor. FFTK library is available for download at

http://www.turbulencehub.org/codes/fftk.

We have implemented various kinds of boundary conditions in FFTK—1) Periodic in all directions, 2) Free-slip wall in one direction and periodic along the other directions, 3) Free-slip walls along two directions and periodic along the third direction, and 4) Free-slip walls along all the three directions. P3DFFT supports boundary conditions of types 1) and 2) only. Thus, FFTK implementation is more general than P3DFFT.

In subsequent discussion in this chapter, we describe the numerical scheme and parallelization strategy, as well as the scaling results of FFTK on Blue Gene/P and Cray XC40 of KAUST. *These results are taken from our journal paper [12].*

3.1 Parallelization Strategy

Here we describe the `FFF` basis of the FFTK library. Implementation of the other basis functions is similar. The data is equally divided in cores for load balancing. In pencil decomposition the data is divided in p_{row} rows and p_{col} columns such that $p_{\text{row}} \times p_{\text{col}} = p$ as shown in Fig. 3.2. The MPI processes are divided among two communicators `MPI_COM_ROW` and `MPI_COM_COL` (see Fig. 3.2). In real space [Fig. 3.2(a)] each core has $N_x/p_{\text{col}} \times N_y/p_{\text{row}} \times N_z$ data points.

The forward FFT transform (from real to complex) follows the following set of steps:

1. We perform forward FFT, `r2c` real-to-complex, along the Z-axis for each data column.

2. We perform `MPI_AlltoAll` operation among the cores in `MPI_COM_COL` to transform the data from the real configuration [Fig. 3.2(a)] to the intermediate configuration [Fig. 3.2(b)].

3. After interprocess communication, we perform forward c2c (complex-to-complex) transform along the Y-axis for each pencil of the array.

4. We perform `MPI_AlltoAll` operation among the cores in `MPI_COM_ROW` to transform the data from the intermediate configuration [Fig. 3.2(b)] to the Fourier configuration [Fig. 3.2(c)].

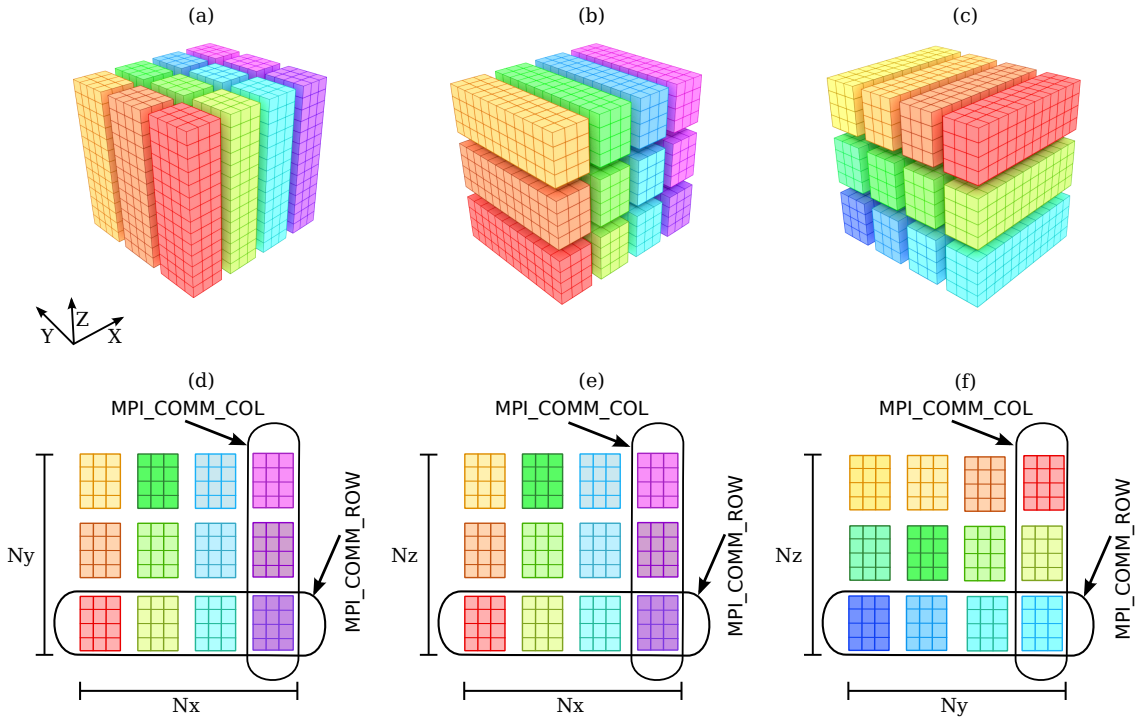


FIGURE 3.2: Pencil decomposition of data for FFT: (a) real space data, (b) intermediate configuration, (c) data in Fourier space. (d, e, f) Division of cores into p_{row} and p_{col} such that $p = p_{\text{row}} \times p_{\text{col}}$ corresponding to the above data division. We maintain consistent colour convention in all the figures. In the subfigure (a), $N = 12$, $p_{\text{row}} = 3$, $p_{\text{col}} = 4$, thus each core contains $N_x/p_{\text{col}} \times N_y/p_{\text{row}} \times N_z = 3 \times 4 \times 12$ data points. Taken from Chatterjee *et al.* [12].

5. Now we perform forward c2c transform along the X-axis for each pencil [see Fig. 3.2(c)].

For one-dimensional FFT operations as described above, we use the FFTW transforms in our library. To perform the FFT along y and z we do not perform local transpose of the data, as it is prone to cache misses. Rather, we employ strided FFT for non-contiguous data along y and x since the strided FFT is more efficient than the local transpose. We also remark that FFTK and Tarang use a meta-template C++ library, `blitz++` [86] for array manipulation; this library provides efficient operations for arrays. Strided FFT along Y and X takes double the time taken by contiguous FFT along Z [12]. We describe strided Transpose-free FFT in Appendix A.

This completes the forward transform. The inverse transform is reverse of the above operation. Note that the above strategy is general, and it works for all the basis functions. Our library also works for two-dimensional (2D) data, for which we set $N_y = 1$ and 1D transform, for which we set $N_x = N_y = 1$. The intermediate state is avoided for 2D Fourier transforms. Also note that the slab FFT can be performed by setting $p_{\text{row}} = 1$, and again, the configuration (b) of Fig. 3.2 is avoided.

In the next section, we provide some details on the library structure of FFTK.

3.2 The FFTK Library

The FFTK class has following major functions. It uses FFTW for performing 1D transforms.

3.2.1 void Init(string basis, int Nx, int Ny, int Nz, int num_p_rows)

This function initialises an instance according to basis, size of the grid and the division of data. We have three spaces during a 3D transform:- Real space, Intermediate space and Fourier space (Fig 3.2). This function sets the number of points along each axis for all the spaces. e.g. In Fourier space the size of grid is $Nx \times Ny \times Nz/2 + 1$. It calls several helper functions to set up the following:

- MPI variables (process rank, num of processors, MPI row communicator and MPI column communicator , number of rows and columns in a pencil division) and size of local arrays.

- Allocates some temporary buffers and sets up several parameters required during data transfer operation.
- Sets FFTW plans and normalization factor according to the chosen basis, the size of the grid, and the division of data.

3.2.2 Forward_transform

This executes the forward FFTW plans initialized earlier and invokes data transfer functions according to the order

- Forward transform along Z
- Divide data along Z and X
- Forward transform along Y
- Divide data along Y and Z
- Forward transform along X
- Normalize the data

3.2.3 Inverse_transform

This executes the inverse FFTW plans initialized earlier in the init function and invokes data transfer functions according to the order

- Inverse transform along X
- Divide data along Y and Z
- Inverse transform along Y
- Divide data along Z and X
- Inverse transform along Z

In the following discussion we will report the scaling results of FFTK . But before that we introduce the HPC clusters on which the tests were performed.

3.3 About the HPC systems

We performed scaling tests of our FFT library and pseudospectral code on Shaheen I, a Blue Gene/P cluster, and Shaheen II, a Cray XC40 cluster, of King Abdullah University of Science and Technology (KAUST). Here are some of the relevant details of these systems.

3.3.1 Blue Gene/P

The Blue Gene/P supercomputer consists of 16 racks with each rack containing 1024 quad-core, 32-bit, 850 MHz PowerPC compute nodes. Hence the total number of cores in the system is 65536. It also has 65536 GB of RAM. The Blue Gene/P nodes are interconnected by a three-dimensional point-to-point *torus* network. The theoretical peak speed of the Blue Gene/P supercomputer is 222 Tera FLOP/s (Floating point operations per second). This system has now been decommissioned by KAUST.

3.3.2 Cray XC40

The Cray XC40 supercomputer has 6174 dual-socket compute nodes each containing two Intel Haswell processors with 16 cores. Each of these cores run at a clock speed of 2.3 GHz. In aggregate, the system has a total of 197568 cores and 790 TB of memory. The compute nodes, contained in 36 water-cooled XC40 cabinets, are connected via a Aries High Speed Network. Cray XC40 adopts a dragonfly topology that yields 57% of the maximum global bandwidth between the 18 groups of two cabinets [37]. Shaheen II delivers a theoretical peak performance of 7.2 Peta FLOP/s and a sustained LINPACK performance of 5.53 Peta FLOP/s. According to the latest TOP500 list, announced in June 2017, Shaheen II is ranked as 18th best supercomputer in the world.

In the next section, we describe the scaling results of FFTK. *These results are taken from our journal paper [12].*

3.4 Scaling of FFTK

Total time in a parallel program can be divided into computation time T_{comp}, communication time across nodes T_{comm}, and latency T_{lat}, i.e.

$$T = T_{\text{comp}} + T_{\text{comm}} + T_{\text{lat}}. \tag{3.1}$$

We report T_{comp} and T_{comm} of FFTK library on the Blue Gene/P and Cray XC40 clusters. For large data size, T_{lat} is quite small compared to the above two, and hence can be neglected. Since the data is divided equally among all the cores, we expect $T_{\text{comp}} \sim p^{-1}$, where p is the number of cores; we observe the above scaling in all our tests. However since FFT involves `MPI_Alltoall` communications, communication time is the most dominant component of the total time.

We have performed forward and inverse transforms using FFTK several times (100 to 1000) for large N^3 grids, and then report the average time taken for a pair of forward and inverse transforms. We computed T_{comp} and T_{comm} for various combinations of grid sizes and number of cores, and observed that

$$T_{\text{comp}} = c_1 p^{-\gamma_1}, \tag{3.2a}$$

$$T_{\text{comm}} = c_2 n^{-\gamma_2}, \tag{3.2b}$$

$$T^{-1} = \frac{1}{C} p^{\gamma}, \tag{3.2c}$$

where c_1, c_2, C, γ_1, γ_2, and γ are constants, p is the number of cores, and n is the number of nodes. Therefore the total time per FFT operation is

$$T = c_1 D \left(\frac{1}{p^{\gamma_1}}\right) + c_2 D \left(\frac{1}{n^{\gamma_2}}\right) + T_0 = C \left(\frac{1}{p^{\gamma}}\right) \tag{3.3}$$

where $D = N^3$ is the data size, and T_0, a measure of latency, is a constant. We measure T_{comp} and T_{comm} by computing the time taken by the respective code-segments using the MPI function `MPI_Wtime`. We record the time when the process enters and leaves the code segment, and then take their difference that yields T_{comp} and T_{comm}.

We now describe our results specific to the Blue Gene/P cluster.

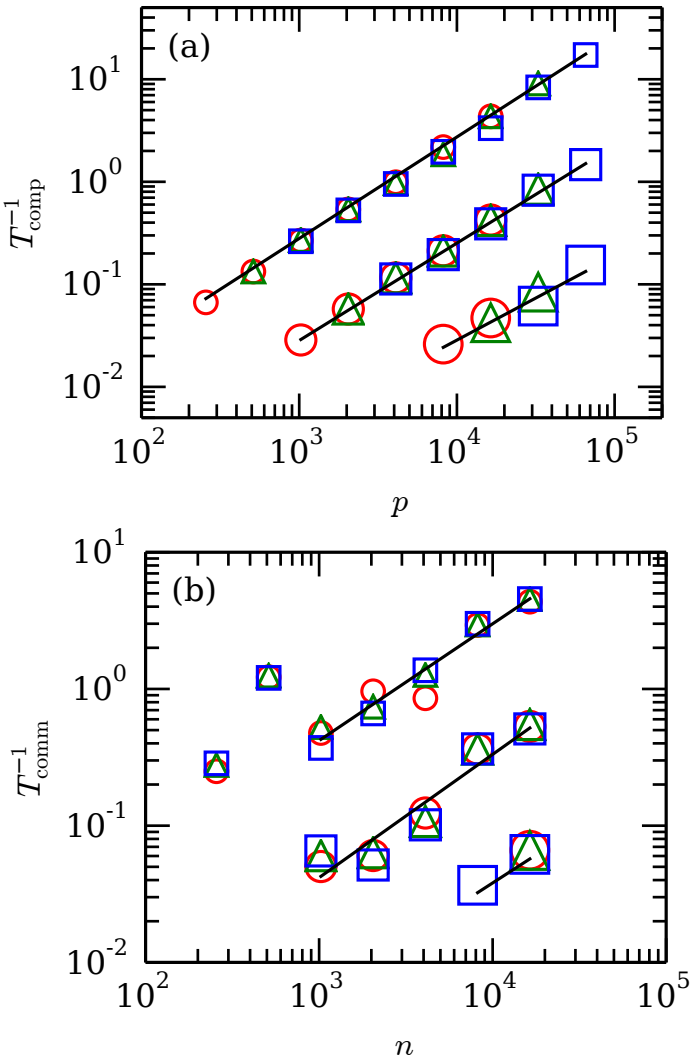


FIGURE 3.3: Scalings of the FFTK library on Blue Gene/P: (a) Plot of inverse computation time $T_{\rm comp}^{-1}$ vs. p (number of cores) for `1ppn` (red circle), `2ppn` (green triangle), `4ppn` (blue square). Here `ppn` represents number of MPI processes per node. The data for grids 2048^3, 4096^3, and 8192^3 are represented by the same symbols but with increasing sizes. The plots show that FFTK exhibits strong scaling in Blue Gene/P. (b) Plot of inverse communication time $T_{\rm comm}^{-1}$ vs. n (number of nodes) with the above notation. $T_{\rm comm}^{-1}$ for $p = 256$ and 512 exhibits a better scaling due to the slab decomposition employed. Taken from Chatterjee *et al.* [12].

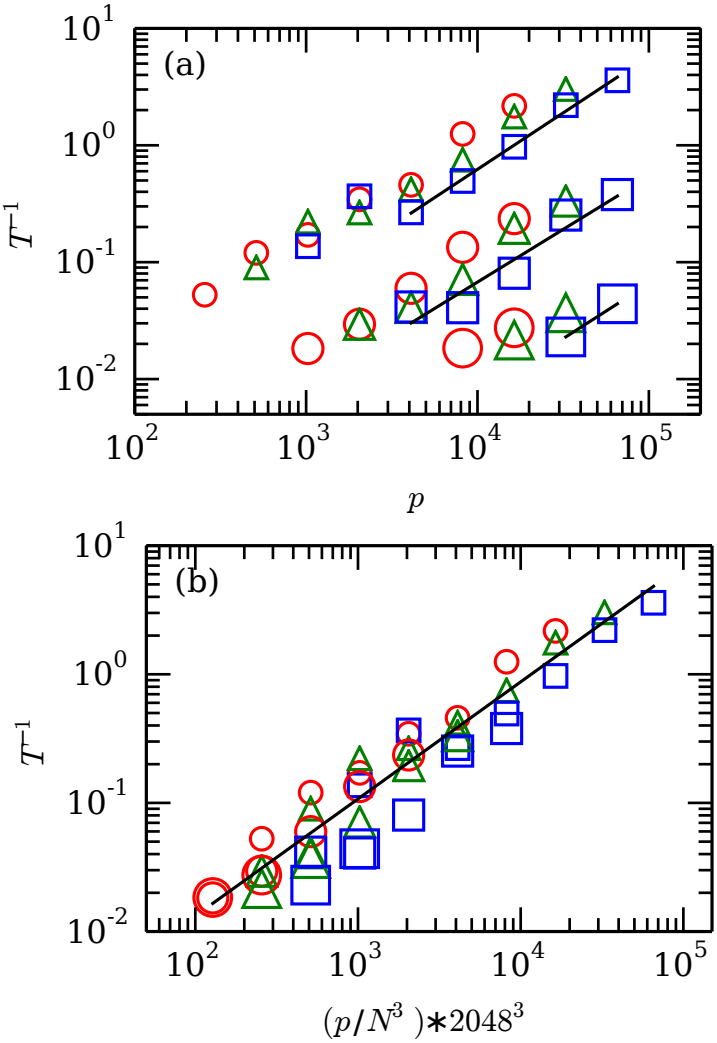


FIGURE 3.4: Scalings of FFTK in Blue Gene/P: (a) Plot of inverse of total time T^{-1} vs. p exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with the exponent $\gamma = 0.91 \pm 0.04$. We follow the same colour and symbol convention as Fig. 3. Taken from Chatterjee *et al.* [12].

3.4.1 Scaling on Blue Gene/P

In Fig. 3.3 we plot T_{comp} and T_{comm}. The results show that the computation time scales as $T_{\text{comp}} \propto p^{-1}$. In the following discussion, we sketch the scaling arguments for T_{comm} that was first provided by Pekurovsky [63].

A Blue Gene/P cluster has torus interconnect for which the *bisection bandwidth* B, defined as the net bandwidth available between bisected partitions of the network, is proportional to the area in the network topology. Hence $B \propto (n')^{2/3}$, where n' is the number of nodes used in communication.

TABLE 3.1: FFTK scaling on Blue Gene/P: The exponents γ_1 for the computation time (T_{comp}), γ_2 for the communication time (T_{comm}), and γ for the total time (T) [refer to Eq. (3.2) for definition]. The maximum nodes used: 16384 with `1ppn`, `2ppn`, and `4ppn`.

γ	ppn	2048^3	4096^3
γ_1	1	1.00 ± 0.01	0.97 ± 0.01
	2	1.00 ± 0.02	0.96 ± 0.01
	4	1.00 ± 0.03	0.95 ± 0.03
γ_2	1	0.7 ± 0.1	0.9 ± 0.1
	2	0.7 ± 0.1	0.8 ± 0.2
	4	0.7 ± 0.1	0.8 ± 0.2
γ	1	0.87 ± 0.05	0.94 ± 0.05
	2	0.81 ± 0.05	0.96 ± 0.09
	4	0.76 ± 0.07	0.9 ± 0.1

In the slab division, $n' = n$, the total number of nodes, and the data to be communicated in the network is $D = N^3$. Therefore, the inverse of the communication time for each FFT is

$$T^{-1}_{\text{comm,slab}} \sim \frac{B}{D} \sim n^{2/3}. \tag{3.4}$$

Here, we show that T_{comm} depends only on number of interacting nodes, not cores because the intra-node communication (among the cores within a node) is typically much faster than the inter-node communication across an interconnect.

In pencil division, the nodes are divided into row nodes (n_{row}) and column nodes (n_{col}). We estimate $n_{\text{row}} \approx n_{\text{col}} \approx n^{1/2}$. Hence each communication within a row (or

a column) involves $n' \approx n^{1/2}$ number of nodes during `MPI_COM_ROW` or `MPI_COM_COL` communications. Hence, the effective bisection bandwidth, B_e, is

$$B_e \sim (n')^{2/3} = (n^{1/2})^{2/3} = n^{1/3}. \tag{3.5}$$

In this decomposition, the data per node is N^3/n. During a communication, either row nodes or column nodes are involved. Hence, the data to be communicated during a `MPI_Alltoall` operation is

$$D = (N^3/n) \times n^{1/2} = N^3/n^{1/2}. \tag{3.6}$$

Therefore, the inverse of communication time for each FFT is

$$T_{\rm comm}^{-1} \sim \frac{B_e}{D} \sim n^{5/6} \approx n^{0.83}. \tag{3.7}$$

We carried the scaling tests on arrays of size 2048^3, 4096^3, and 8192^3 on cores ranging from 256 to 65536. Blue Gene/P has 4 cores in each node, therefore we executed our efficiency test runs on 1, 2, and 4 cores per node, denoted as `1ppn`, `2ppn`, and `4ppn` respectively. We present these results in Fig. 3.3, with the subfigures (a,b) showing the inverse of computation and communication timings respectively. We represent `1ppn`, `2ppn`, and `4ppn` results using circles, triangles, and squares respectively, and the grid sizes 2048^3, 4096^3, and 8192^3 using increasing sizes of the same symbols. Fig. 3.3 shows that $T_{\rm comm}^{-1}$ for $p = 256$ and 512 shows better scaling than those for larger number of processors. This is attributed to the slab decomposition. Note however that the slab decomposition is possible only when number of processors is smaller than the number of planes of the data (N for N^3 grid). In Fig. 3.4(a,b), we exhibit T^{-1} vs. p and T^{-1} vs. p/N^3 to test strong and weak scaling, respectively.

We compute the exponents γ_1, γ_2, and γ of Eq. (3.3) using linear regression on the data of Fig. 3.3 and Fig. 3.4(a). In Table 3.1, we list the exponents for `1ppn`, `2ppn`, and `4ppn` and grids sizes of 2048^3 and 4096^3. As expected, the exponent $\gamma_1 \approx 1$ since the data is equally distributed among all the cores. The exponent γ_2 is approximately 0.7 for 2048^3 for all three cases, but it ranges from 0.8 to 0.9 for 4096^3. The increase in γ_2 with the grid size is probably due to the larger packets communicated for 4096^3 grids. The exponent γ_2 is quite close to the theoretical estimate of $5/6 \approx 0.83$ for 4096^3 grid [see Eq. (3.7)]. Our computation also shows the best match for γ_2 with the theoretical estimate is for `1ppn`, and it decreases slightly for for larger `ppn`. The variations with `ppn` is due to *cache misses*.

TABLE 3.2: Comparison of FFTK and P3DFFT on Blue Gene/P for 8192 nodes with 1ppn and 4ppn. Here p = nodes $\times$ ppn.

Grid	p	time/step (s) FFTK	time/step (s) P3DFFT
4096^3	8192×1	8.18	8.06
4096^3	8192×4	4.14	4.06
8192^3	8192×1	71.2	70.0
8192^3	8192×4	45.7	46.2

Now we present the scaling of the total time spent in a pair of FFT computation (T). In Fig. 3.4(a), that shows a power law scaling $T^{-1} \propto p^{\gamma}$, we get a scaling close to the ideal exponent $\gamma = 5/6$ [see Eq. (3.7)]. This feature is called *strong scaling*. Naturally the larger grids take longer time than the smaller grids. However, when we increase p and N^3 proportionally, all our results collapse into a single curve, as exhibited in T^{-1} vs. p/N^3 plot of Fig. 3.4(b). Thus FFTK exhibits both strong and weak scaling.

In Blue Gene/P, it is seen that T_{comp} and T_{comm} are comparable to each other due to the fact that the interconnect is quite fast, but the compute processors are slow (850 MHz). Hence the total time T is affected by both T_{comp} and T_{comm}. Thus, γ is very close to unity, yielding an almost linear scaling, at least for the 4096^3 grid. Moreover, we have performed FFT for 8192^3 grid with 8192 and 16384 nodes; it was not possible with lower number of nodes due to memory limitations. Due to lack of data points for 8192^3 grid, we do not have a reliable scaling exponent for this grid .

We compare the total time per step of FFTK with the popular library P3DFFT for 4096^3 and 8192^3 grids on 8192 nodes with 1ppn and 4ppn. The comparison listed in Table 3.2 shows that both the libraries are equally efficient. We also compute the effective FLOP rating of FFTK and P3DFFT. A pair of forward and inverse FFT involves $2 \times 5N^3 \log_2 N^3$ floating point operations [32]. Using this formula we estimate the effective FLOP rating of the cluster for various grid sizes and ppn. The results are listed in Table 3.3. A comparison of the above performance with the average theoretical rating of each core (approximate 3.4 Giga FLOP/s) suggests that the efficiency of the system for a FFT ranges from approximately 5% (for 4ppn) to 10% (for 1ppn) of the peak performance. The loss of performance is due to large communication time and cache misses during a FFT operation. Typically, efficiency of a HPC system is measured using

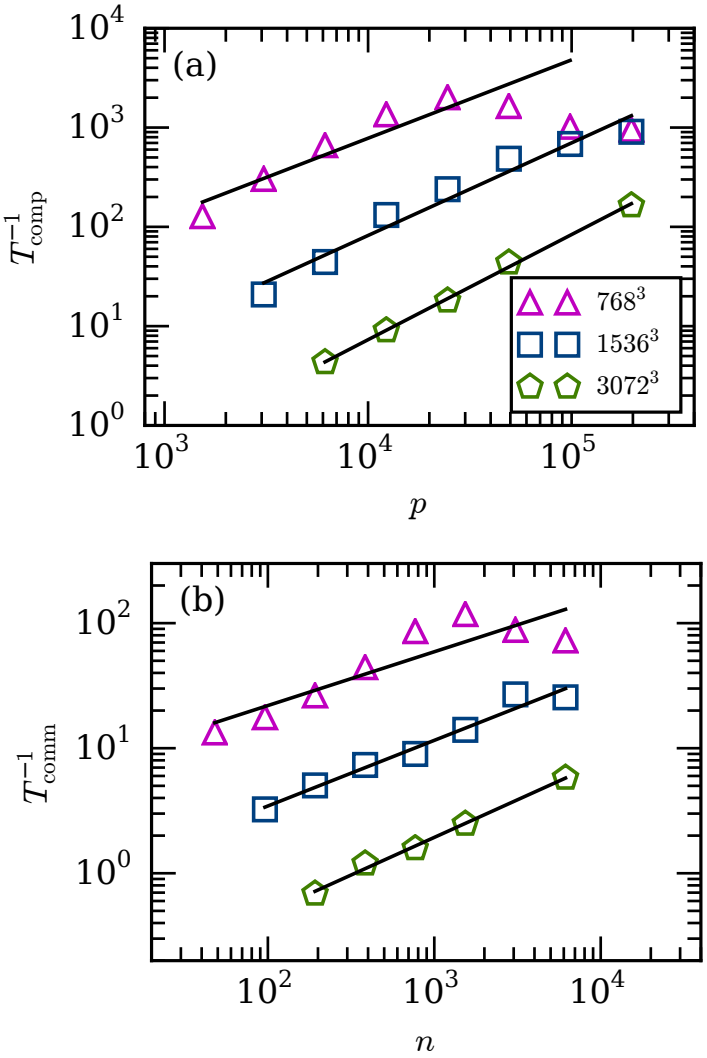


FIGURE 3.5: Scalings of FFTK on Cray XC40: (a) Plots of T^{-1}_{comp} vs. p (number of cores) for 768^3, 1536^3, and 3072^3 grids. (b) Plots of T^{-1}_{comm} vs. n (number of nodes) using the above convention. Taken from Chatterjee *et al.* [12].

TABLE 3.3: FFTK on Blue Gene/P: Effective FLOP rating in Giga FLOP/s of Blue Gene/P cores for various grid sizes and ppn. The efficiency E is the ratio of the effective per-core FLOP rating and the peak FLOP rating of each core (approximately 3.4 Giga FLOP/s).

Grid	ppn	Giga FLOP/s	E
2048^3	1	0.38	0.11
	2	0.28	0.082
	4	0.17	0.050
4096^3	1	0.36	0.11
	2	0.25	0.073
	4	0.14	0.041
8192^3	1	0.36	0.11
	2	0.26	0.076
	4	0.15	0.044

$T_1/(pT_p)$ where T_p is the time taken to perform operation using p processors. The data for large grids, e.g. 1024^3, cannot fit in the memory of a single processor, hence we cannot compute T_1 and hence $T_1/(pT_p)$. Therefore we use a more stringent measure. We measure the efficiency (E) as the ratio of the per-core FLOP rating and the peak rating. We list this efficiency in Table 3.3.

In the next subsection we discuss the scaling of FFTK on Cray XC40.

3.4.2 Scaling on Cray XC40

Each node of Cray XC40 has 32 compute cores, with each core having an approximate rating of 36 Giga FLOP/s. Thus each core of Cray XC40 is approximately 10 times faster than that of Blue Gene/P. Hence, for given grid size and p, T_{comp} for Cray XC40 is much smaller than that for Blue Gene/P. Total number of cores in Cray XC40 is 3×2^{16}.

The Cray XC40 employs dragonfly topology which consists of hierarchy of star topology that yields bandwidth proportional to the number of interacting nodes. Hence

the bandwidth is

$$B_e \sim n', \tag{3.8}$$

where n' is the number of interacting nodes. For the pencil decomposition the total number of nodes n is divided as $n = n_{\text{row}} \times n_{\text{col}}$. Hence $n' = n/n_{\text{row}}$ for `MPI_COMM_COL` communicator and $n' = n/n_{\text{col}}$ for `MPI_COMM_ROW` communicator. Note that the data to be communicated during `MPI_Alltoall` operation is $(N^3/n)n'$. Hence, the inverse of the communication time is

$$T_{\text{comm}}^{-1} \sim \frac{B_e}{D} \approx \frac{n'}{(N^3/n)n'} \sim n, \tag{3.9}$$

implying a linear scaling. However, as will be shown later, such a scaling is not observed in practice. For example, Hadri *et al.* [37] showed that the maximum global bandwidth between the 18 groups of two cabinets is approximately 57% of the peak performance, hence we expect suboptimal performance for communications for FFT due to `MPI_Alltoall` data exchange.

We executed FFTK scaling on grid sizes 768^3 to 6144^3 using cores ranging from 1536 to 196608 (3×2^{16}). Each node of Shaheen II contains 32 cores. We use all the cores in a given node for maximum utilization. Both the grid sizes and number of cores are of the form 3×2^n, since the total number of cores is divisible by 3.

We do the scaling analyses for `FFF` and `SFF` basis and present the T_{comp}, T_{comm}, and T in Figs. 3.5 and 3.6 for the `FFF` basis only since `SFF` basis has similar behavior. We observe that the total computation time for the process is an order of magnitude smaller than the total communication time. Hence the efficiency of FFT is dominated by the `MPI_Alltoall` communication of the FFT. The figures show that $T_{\text{comp}}^{-1} \sim p^{\gamma_1}$, $T_{\text{comm}}^{-1} \sim n^{\gamma_2}$, and $T^{-1} \sim p^{\gamma}$, with minor deviations from the power law for 768^3 grid with large p's ($p \geq 98000$). Thus the data exhibits a strong scaling nearly up to 196608 cores. We also observe that all the data nearly collapse to a single curve when we plot T^{-1} vs. p/N^3, hence FFTK exhibits both weak and strong scaling nearly.

We performed scaling tests of our FFT library on a Cray XC40 cluster (Shaheen II) of KAUST for SFF basis. We divide the cores and data in two different ways:

1. In the first decomposition, `SFF I`, we vary our grid size from 768^3 to 6144^3, and the number of cores from 1536 to 196608 (3×2^{16}). For the scaling analysis, both the grid sizes and number of cores are of the form 3×2^n since the maximum number of cores in Cray XC40, 3×2^{16}, is divisible by 3. We observe that T_{comm} does not vary significantly under the variation of p_{col}, similar to P3DFFT [63].

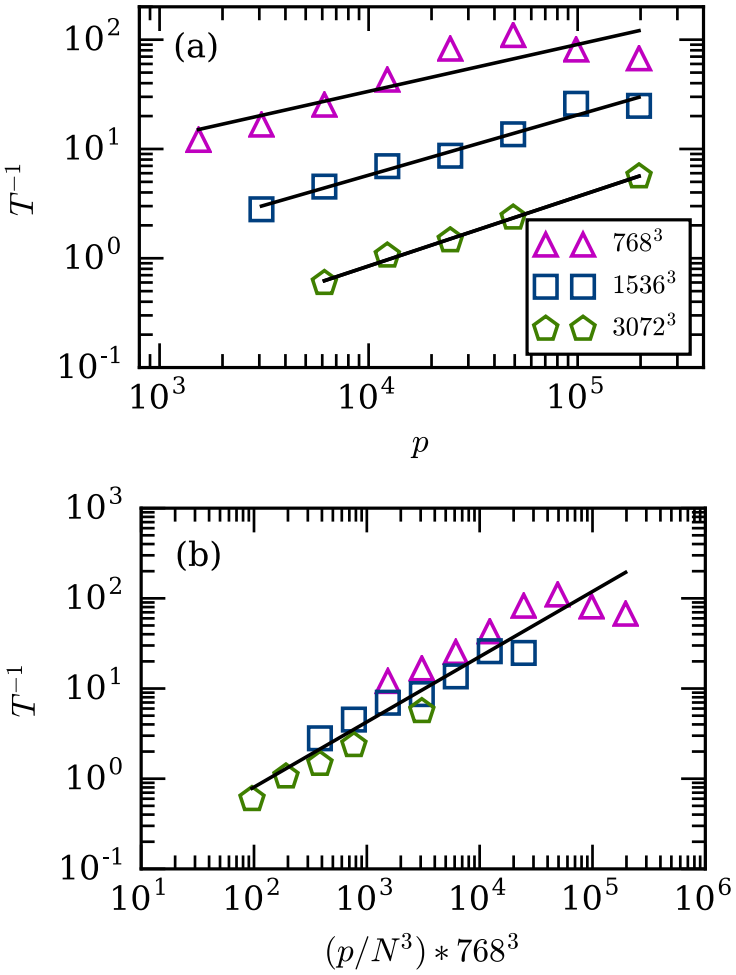


FIGURE 3.6: Scaling of FFTK on Cray XC40 for the FFF basis: (a) plots of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids. (b) plots of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.72 \pm 0.03$. Taken from Chatterjee *et al.* [12].

Here we report our results for $p_{\text{col}} = N$ (where N^3 is the grid size) keeping uniformity in mind.

2. In the second decomposition, `SFF II`, we divide the data $N_x \times N_y \times N_z$ and cores in such a way that large blocks of data and cores remain within a rack. The total number of nodes in a Shaheen II rack is either 171 or 172. We choose lower of the two numbers for our processor division. Since $171 = 19 \times 9$, it is best to take p_{row} and p_{col} as multiples of 19 and 9 respectively. Each of the $36 = x_2 y_2$ racks of Shaheen II consists of nodes carrying $32 = x_1 y_1$ cores each. Therefore, for efficient and modular decomposition, we take $p_{\text{row}} = 19 x_1 x_2$ and $p_{\text{col}} = 9 y_1 y_2$. For the data decomposition, using Fig. 3.2 we deduce that N_z and N_x are divisible by p_{row} and p_{col} respectively, but N_y is divisible by $\text{LCM}(19 x_1 x_2, 9 y_1 y_2)$, where LCM is the least common multiple. For our efficiency test runs we take $x_1 = 8, y_1 = 4, x_2 = (1, 2, 4)$ and $y_2 = (1, 3, 9)$, that yields $p = 5472, 10944, 21888, 16416, 32832, 65664, 49248, 98496$ and 196992. We choose the grids as $\text{G}_1 = 684 \times 5472 \times 288$, $\text{G}_2 = 2052 \times 5472 \times 576$, and $\text{G}_3 = 6156 \times 5472 \times 2304$.

The exponents for various grids for the `FFF`, `SFF I` and `SFF II` basis are listed in Table 3.4. We observe that $\gamma_1 \approx 1$, except for 768^3, thus yielding a linear scaling for the computation time. The exponent γ_2 of the communication time ranges from 0.43 to 0.82 as we increase the grid size from 768^3 to 3072^3, which may be due to increased efficiency of the network for larger data size. As described above, the total time is dominated by the communication time, hence $\gamma \approx \gamma_2$. Also, the `SFF II` basis appears to be slightly more efficient than the `SFF I` which is more efficient than the `FFF` basis. We however remark that these exponents have significant errors, ranging from 1% to 20%, These errors are due to errors in time measurements, as well as due to lack of data points.

We compute the effective FLOP rating of the cluster by dividing the total number of floating operations for a pair of FFT ($2 \times 5N^3 \log_2 N^3$) with the total time taken. This operation yields the average rating of each core of Cray XC40 as 0.45 to 0.64 Giga FLOP/s that translates to 1.3% to 1.8% of the peak performance (36 Giga FLOP/s) (see Table 3.5). It is important to contrast the efficiencies of the two HPC clusters discussed in this chapter. The efficiency of Blue Gene/P at approximately 4% (for `4ppn`) is higher than that of Cray XC40 ($\sim$ 1.5%). A node of Cray XC40 comprises of 32 cores, each with a peak rating of 36 Giga FLOP/s rating. Hence maximum compute power per node is 1152 Giga FLOP/s. On the other hand, a Blue Gene/P node contains 4 cores with a peak rating of $4 \times 3.4 = 13.6$ Giga FLOP/s. Thus, each node of Cray XC40 is

TABLE 3.4: FFTK scaling on Cray XC40 for the FFF and SFF basis: The exponents γ_1 for the computation time (T_{comp}), γ_2 for the communication time (T_{comm}), and γ for the total time (T) [refer to Eq. (3.2) for definition]. Maximum cores used: 196608.

FFF			
Grid	γ_1	γ_2	γ
768^3	0.79 ± 0.14	0.43 ± 0.09	0.43 ± 0.09
1536^3	0.93 ± 0.08	0.52 ± 0.04	0.55 ± 0.04
3072^3	1.08 ± 0.03	0.60 ± 0.02	0.64 ± 0.02
SFF I			
Grid	γ_1	γ_2	γ
768^3	0.82 ± 0.13	0.44 ± 0.03	0.46 ± 0.04
1536^3	0.97 ± 0.07	0.63 ± 0.02	0.66 ± 0.01
3072^3	0.99 ± 0.04	0.70 ± 0.05	0.73 ± 0.05
SFF II			
Grid	γ_1	γ_2	γ
$684 \times 5472 \times 288 (G_1)$	0.88 ± 0.05	0.55 ± 0.02	0.58 ± 0.03
$2052 \times 5472 \times 576 (G_2)$	0.82 ± 0.08	0.57 ± 0.08	0.70 ± 0.08
$6156 \times 5472 \times 2304 (G_3)$	0.87 ± 0.04	0.78 ± 0.05	0.81 ± 0.04

TABLE 3.5: FFTK on Cray XC40: Effective FLOP rating in Giga FLOP/s of Cray XC40 cores for various grid sizes and ppn. The efficiency E is the ratio of the effective per-core FLOP rating and the peak FLOP rating of each core (approximately 36 Giga FLOP/s).

Grid Size	768^3	1536^3	3072^3
GFlop/s	0.45	0.53	0.64
E	0.013	0.015	0.018

approximately 100 times faster but with an interconnect of Cray XC40 that is not 100 times faster than the interconnect of Blue Gene/P. Therefore, for data communication the processors of Cray have to remain idle than those in Blue Gene/P. For the Blue Gene/P, the relatively slower processors do not have to stay idle as long. This is especially critical for FFT which is communication intensive. Thus, the faster processor and relatively slower interconnect of Cray XC40 result in an overall lower efficiency than the Blue Gene/P. In this sense, the efficiency of hardware depends on the application; for FFT computation, a faster switch is more important than a faster processor.

3.5 Hybridization

Using threads as well as MPI in a program is called hybrid code in terms of parallelization. FFTW supports threaded transforms. In pure threaded transforms in p threads, the N^2 columns are equally divided into p threads. In a hybrid scenario, columns in each pencil are divided into p threads. GNU C++ supports two kinds of threads, i.e. OpenMP and pthreads (posix threads). FFTW uses pthreads as it provides finer control over thread creation and destruction. Moreover there are two kinds of parallelism, 1) data parallelism and 2) task parallelism. In data parallelism, the total data is equally distributed among the processor cores. On the other hand, in task parallelism different task is given to processor cores.

We wrote a hybrid code in which we use data parallelism. Speedup (Figure 3.7(a)) of a parallel program is defined as $S = T_s/T_p$, where T_s is time taken by serial program and T_p is the time taken by its parallel counter part. Ideally it should scale as $S \propto p^{-1}$. We get the exponent to be $\gamma = 0.53 \pm 0.03$. Efficiency (Figure 3.7(b)) of a parallel code is defined as $E = S/p$. Ideally it should scale as $E = p^0$. We get the exponent

$\gamma = -0.47 \pm 0.03$. The Strong scaling (Figure 3.7(c)) is defined as $T^{-1} = 1/T_p$. Ideally it should scale as $T^{-1} \propto p^1$. We get the exponent to be $\gamma = 0.53 \pm 0.06$. The Weak scaling (Figure 3.7(d)) is plotted as p/N^3 vs T^{-1}. Ideally all plots should fall into one line of slope p^1. We get the exponent to be $\gamma = 0.94 \pm 0.07$.

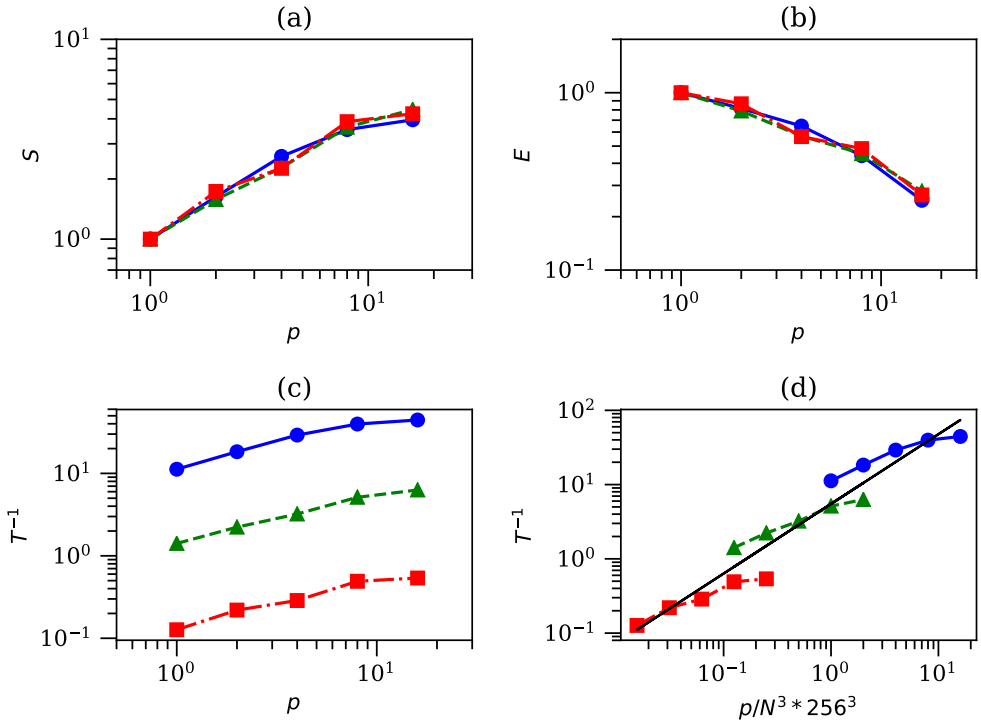


FIGURE 3.7: In these figures, blue circles, green triangles and red squares respectively represent 256^3, 512^3 and 1024^3 grids. γ is averaged over three lines except for weak scaling for which it is averaged over two. (a) Speedup, $S = T_s/T_p$, $\gamma = 0.53 \pm 0.03$, (b) Efficiency $E = S/p = T_s/pT_p$, $\gamma = -0.47 \pm 0.03$ (c) Total time inverse, $\gamma = 0.53 \pm 0.06$ (d) Weak scaling, $\gamma = 0.94 \pm 0.07$

Kindly note that these are preliminary runs. We plan to perform a more detailed set of hybrid runs.

3.6 Summary of the chapter

In this chapter we describe the features of our FFT library FFTK, and the scaling results of FFTK. On Blue Gene/P, FFTK scales well up to 8192^3 grid with core count up to 65536, and on Cray XC40, it scales up to 3072^3 with core count up to 196608 cores. The scaling is not linear due to communications involved in MPI_All_to_All operations.

However, the scaling is close to the predicted ones based on bi-sectional widths.

In the next chapter, we describe the scaling of Tarang solvers.

Chapter 4

Parallel scaling of Tarang solvers

In Chapter 2 we described the features of Tarang. In this chapter we will describe the scaling results of some of the solvers of Tarang—hydrodynamic turbulence and thermal turbulence. We performed scaling tests of Tarang, a pseudospectral code on Shaheen I, a Blue Gene/P cluster, and Shaheen II, a Cray XC40 cluster, of King Abdullah University of Science and Technology (KAUST). We varied the grid size from 768^3 to 4096^3, and the number of cores up to 65536 on Blue Gene/P, and up to 196608 on Cray XC40.

The most expensive part of a pseudospectral simulation is the FFT that consumes approximately 70% to 80% of the total time. In addition, a flow solver involves array operations such as array multiplication; fortunately this array operation scales linearly with the number of processors. Also, a large amount of data makes the read and write operations from the hard disc quite expensive. In Tarang, we employ parallel I/O using hdf5 library. Due to the diverse set of functions in fluid solvers, it is difficult to separate the computation and communication times. Hence we report scaling of the total time for these solvers. *The results presented in this Chapter are taken from our journal paper [12].*

4.1 Scaling of fluid solver on HPC systems

We solve equations (1.5, 1.6) using Tarang. A fluid simulation requires 15 arrays each of size N^3 to store three components of the velocity field in real and Fourier spaces, the force field, the nonlinear terms $\mathbf{u} \cdot \nabla \mathbf{u}$, and three temporary arrays [93]. We employ the

fourth-order Runge Kutta scheme for time stepping, and dealias the nonlinear terms using the 2/3-rd rule [6, 9]. Each time step requires 36 FFT operations. Refer to Verma *et al.* [93] for further details and validation tests of the fluid solver. We run our efficiency test runs for 10 to 100 time steps depending on the grid size. The time reported in the present section is an average over these time steps. The above times do not include those of I/O operations.

4.1.1 Scaling on Blue Gene/P

We performed fluid efficiency test runs on 2048^3 and 4096^3 grids using cores ranging from 1024 to 65536. In Fig. 4.1(a) we plot the inverse of the total time per solver iteration vs. p. We observe that $T^{-1} \sim p^{\gamma}$. The exponents γ listed in Table 4.1 show that $\gamma = 0.95 \pm 0.05$ and $\gamma = 0.8 \pm 0.1$ for grid sizes of 2048^3 and 4096^3, respectively. This demonstrates that the fluid solver exhibits a strong scaling.

The above data nearly collapses into a single curve in the plot of T^{-1} vs. p/N^3, as shown in Fig. 4.1(b). Hence we conclude that our fluid solver also exhibits weak scaling. The exponent of the weak scaling is $\gamma = 0.97 \pm 0.06$.

4.1.2 Scaling on Cray XC40

We performed fluid efficiency test runs on grid sizes of 768^3, 1536^3 and 3072^3 using core counts ranging from 1536 to 196608. The choice of the aformentioned grid is due to the fact that the number of cores of Cray XC40 used for our scaling is 3×2^{16}, which is close to the maximum available on Cray XC40. We employ periodic boundary conditions along all the walls. As done for Blue Gene/P cluster, we compute the total time taken for each iteration of the solver.

Fig. 4.2(a) shows that the plots of $T^{-1} \propto p^{\gamma}$, except for 768^3 grid with large p's ($p \geq 98000$). Thus FFTK exhibits a strong scaling, except for 768^3 grid when p is large ($p \lessapprox 98000$). We observe that γ for the 768^3, 1536^3 and 3072^3 grids are approximately 0.28, 0.44 and 0.68 respectively. The three curves for the three different grids collapse into a single curve when the x axis is chosen as p/N^3 (except for $p \geq 98000$). This result shows a common scaling when the number of cores and data sizes are increased by an equal factor. Thus our fluid solver exhibits both weak and strong scaling nearly up to 196608 cores of the Cray XC40.

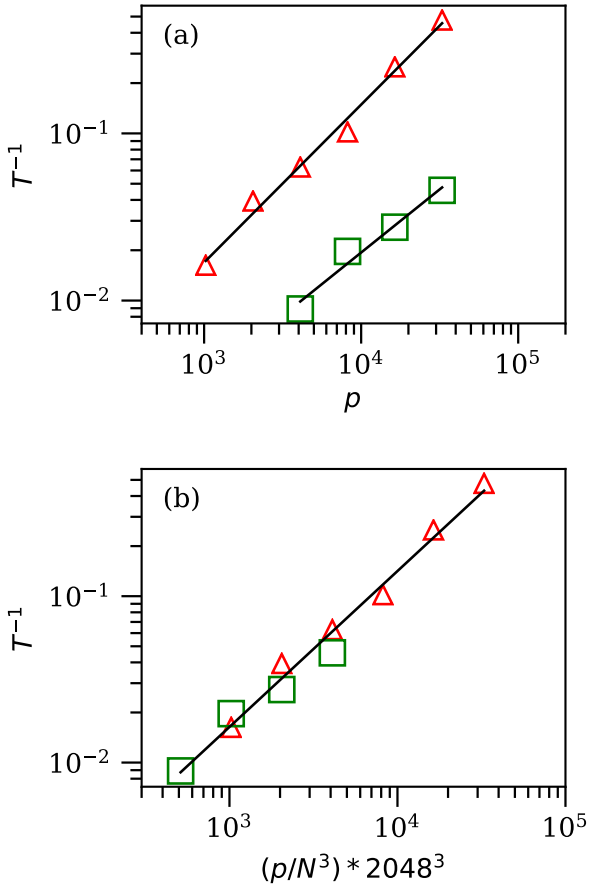


FIGURE 4.1: Scaling of the fluid spectral solver on Blue Gene/P: (a) Plot of T^{-1} vs. p for 2048^3 (red triangle) and 4096^3 (green square) grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.97 \pm 0.06$. Taken from Chatterjee *et al.* [12].

TABLE 4.1: Scaling exponent of the total time of the fluid solver on Blue Gene/P and Cray XC40 for various grids (definition: $T \sim p^{-\gamma}$).

Blue Gene/P		Cray XC40	
Grid size	γ	Grid size	γ
2048^3	0.95 ± 0.05	768^3	0.28 ± 0.15
4096^3	0.8 ± 0.1	1536^3	0.44 ± 0.06
-	-	3072^3	0.68 ± 0.02

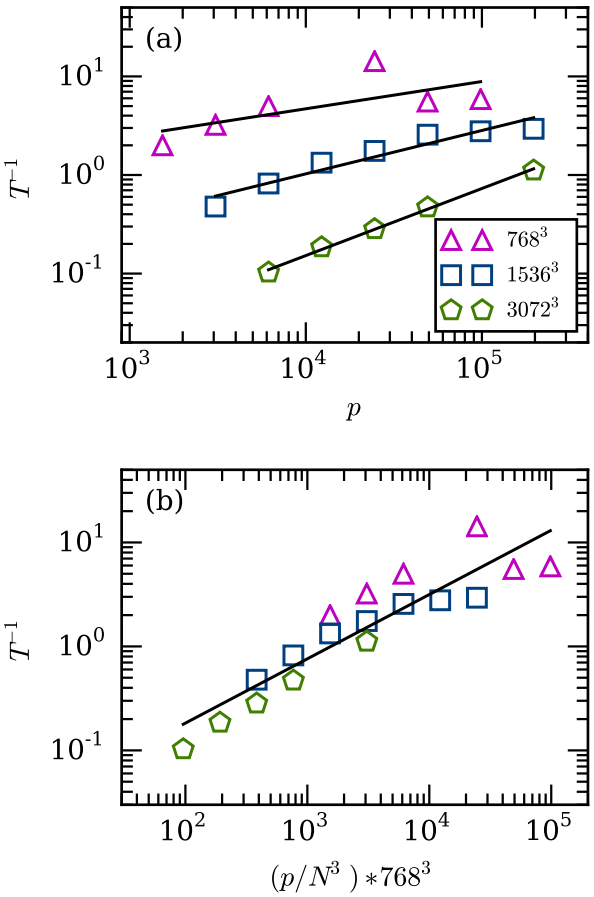


FIGURE 4.2: Scaling of the fluid spectral solver on Cray XC40: (a) Plot of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent $\gamma = 0.62 \pm 0.07$. Taken from Chatterjee *et al.* [12].

In the next section, we describe scaling of turbulent convection.

4.2 Scaling of turbulent convection module of Tarang

For Rayleigh Benard Convection (RBC) simulations, we solve Eqs. (1.10, 1.11, 1.12) using Tarang. A simulation of RBC requires 18 arrays of size N^3. We use fourth-order Runge Kutta scheme for time stepping that needs 52 FFT per time step [93]. For scaling tests, we run our efficiency test runs for 10 to 100 time-steps. In this section, the reported time is an average over appropriate number of time steps. The timing data from our test runs on the Blue Gene/P and Cray XC40 clusters are as follows:

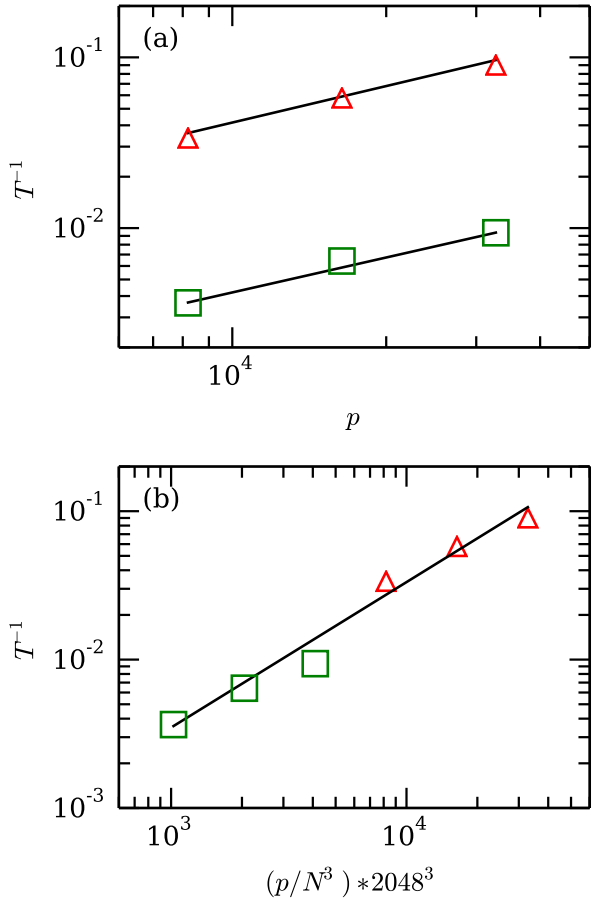


FIGURE 4.3: Scaling of the RBC solver on Blue Gene/P for the FFF basis: (a) Plot of T^{-1} vs. p for 2048^3 (red triangle) and 4096^3 (green square) grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.68 \pm 0.08$. Taken from Chatterjee *et al.* [12].

TABLE 4.2: Scaling exponents of the total time of the RBC solver on Blue Gene/P and Cray XC40 for various grids for the `FFF` and `SFF` basis functions (definition: $T \sim p^{-\gamma}$).

Blue Gene/P		Cray XC40	
		`FFF`	
Grid Size	γ	Grid Size	γ
2048^3	0.71 ± 0.04	768^3	0.49 ± 0.14
4096^3	0.68 ± 0.08	1536^3	0.64 ± 0.04
-	-	3072^3	0.74 ± 0.03
		`SFF`	
Grid Size	γ	Grid Size	γ
-	-	768^3	0.62 ± 0.06
-	-	1536^3	0.74 ± 0.09
-	-	3072^3	0.80 ± 0.05

4.2.1 Blue Gene/P

We performed RBC efficiency test runs on 2048^3 and 4096^3 grids using cores ranging from 8192 to 65536. In Fig. 4.3(a,b) we plot T^{-1} vs. p and T^{-1} vs. p/N^3 respectively. Here T is the time taken per step, and p is the number of cores. We observe that $T^{-1} \sim p^{\gamma}$ with the exponent $\gamma = 0.71$ and 0.68 for the 2048^3 and 4096^3 grids respectively (see Table 4.2). Thus the RBC solver indicates a strong scaling. As exhibited in Fig. 4.3(b), the data nearly collapses into a single curve when we plot T^{-1} vs. p/N^3, hence exhibiting a weak scaling as well.

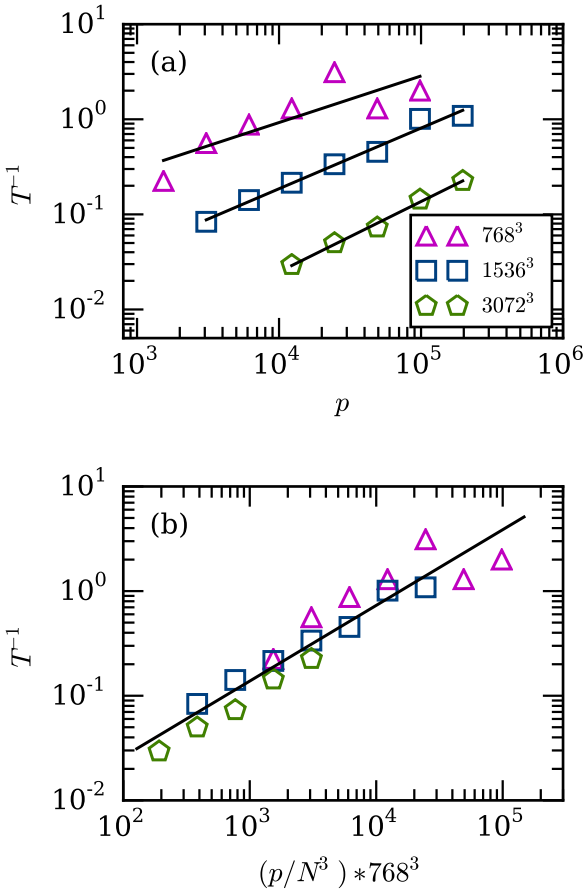


FIGURE 4.4: Scaling of the RBC spectral solver for the FFF basis on Cray XC40: (a) Plot of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.72 \pm 0.06$. Taken from Chatterjee *et al.* [12].

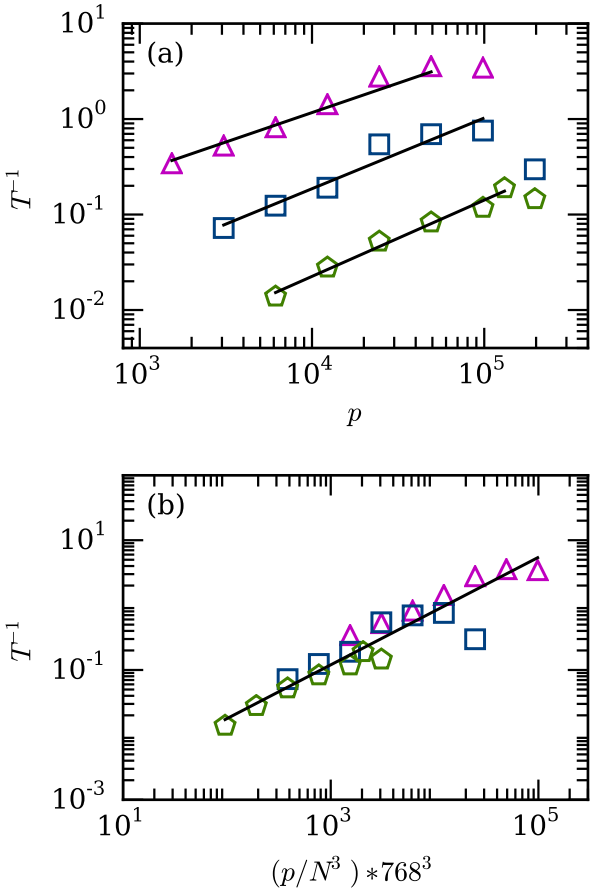


FIGURE 4.5: Scaling of the RBC spectral solver for the SFF basis on Cray XC40: (a) Plot of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.83 \pm 0.03$. Taken from Chatterjee *et al.* [12].

4.2.2 Cray XC40

We simulated RBC on 768^3, 1536^3 and 3072^3 using cores ranging from 1536 to 168608 for FFF and SFF basis. For 3300 iterations of RBC simulation on 2048^3 grid, the total simulation is 1.7×10^5 seconds, thus the time per iteration of RBC on 2048^3 grid is approximately 51.5 seconds. The inverse of the total time plotted in Fig. 4.4(a) scales as $T^{-1} \sim p^{\gamma}$ with $\gamma = 0.49$, 0.64 and 0.74 for 768^3 (except for $p = 196608$), 1536^3 and 3072^3 grids respectively (also see Table 4.2). The plot indicates a strong scaling for the RBC solver. In Fig. 4.4(b) we plot T^{-1} vs. p/N^3; here the data collapses into a single curve (see Fig. 4.4) thus indicating a weak scaling. In Fig. 4.5 we plot T^{-1} vs. p and T^{-1} vs. p/N^3 for the SFF basis. The exponents listed in Table 4.2 show that FFF and SFF scale in a similar manner, with the SFF basis scaling slightly better than the FFF basis. The plots and the scaling exponents demonstrate that our RBC solver exhibits both strong and weak scaling up to nearly 196608 cores.

4.3 Summary and discussion

The results discussed in the chapter indicate that Tarang solvers for fluid flows and thermal convection scale well up to 65536 cores of Blue Gene/P and 196608 cores of Cray XC40. Note, however, that the scaling is not linear essentially due to the large data communications during FFT operations. The other functions of a spectral solver that require parallelization are multiplication of array elements and input/output (I/O). Multiplication of array elements is trivial to parallelize. Since the data-size involved in high-resolution turbulence simulation is very large, of the order of several terabytes, it is more efficient to use parallel I/O. In our spectral code, we use the HDF5 library to perform parallel I/O.

In Tarang there are functions to compute energy flux, shell-to-shell energy transfers, and ring-to-ring energy transfers. These quantities are computed in spectral space [88, 59]. In Sec. 6.5, we will briefly describe the computation of the energy flux for RBC. We also remark that the Object-oriented design of Tarang helps implement new basis function, or include new communication strategies or add a new force type, etc. We have used this design to implement FFF, SFF, SSF, SSS basis functions in two and three dimensions in a single code. We also make use of efficient libraries such as blitz++ and HDF5. These are some of the unique features of FFTK and Tarang.

In the next two chapters we will describe how Tarang solvers are used to compute

energy spectrum and flux of hydrodynamic and thermal turbulence.

Chapter 5

High-resolution simulation of hydrodynamic turbulence

Hydrodynamic turbulence remains an unsolved problem of classical physics. Using Tarang we attempt to address one of the key issues of hydrodynamic turbulence—$E_u(k)$ and $\Pi_u(k)$ for large-Re flows in the inertial and dissipative range. This is the topic of this chapter. We also discuss the shell-to-shell energy transfer in hydrodynamic turbulence using very-large scale simulation. *These results are taken from our journal paper [97].*

5.1 Models of energy spectrum in inertial and dissipative range

Kolmogorov [42, 43] proposed that for a homogeneous, isotropic, and steady turbulence under the limit of Re $\to \infty$, the energy spectrum $E_u(k)$ and the energy flux $\Pi_u(k)$ in the inertial range follow:

$$E_u(k) = K_{\mathrm{Ko}} \epsilon^{2/3} k^{-5/3}, \tag{5.1}$$

$$\Pi_u(k) = \text{const} = \epsilon_u, \tag{5.2}$$

where ϵ_u is the energy dissipation rate, and K_{Ko} is Kolmogorov's constant. The above functions are universal, i.e., they are independent of forcing, dissipative mechanisms, fluid properties, etc. Equation (5.1) has been generalised so as to include the dissipative

range that includes the dependence on the kinematic viscosity ν, and it is commonly written as

$$E_u(k) = K_{\mathrm{Ko}} \epsilon^{2/3} k^{-5/3} f(k/k_d), \tag{5.3}$$

where $f(k/k_d)$ is a universal function, and $k_d \sim (\epsilon/\nu^3)^{1/4}$ is the dissipation wavenumber scale, also called Kolmogorov's wavenumber. Pao [62], Pope [67], and Martínez *et al.* [54] proposed models for $f(k/k_d)$.

Pao [62] proposed that $f(x) \sim \exp(-x^{4/3})$, but according to Pope [67]

$$f(x) \sim \exp\left\{[x^4 + c_\eta^4]^{1/4} - c_\eta\right\}, \tag{5.4}$$

where c_η is a constant. Pope's model [67] is in good agreement with earlier experimental results (see Saddoogchi and Veeravalli [76] and references therein), which are given below. Pao [62] argued that his predictions fit well with the experimental results of Grant *et al.* [35]. Martínez *et al.* [54] proposed that

$$E_u(k) \sim (k/k_d)^\alpha \exp[-\beta(k/k_d)], \tag{5.5}$$

and found good agreement between their predictions and numerical results for moderate Reynolds numbers. We however caution that the above formula are approximate forms of the energy spectrum, and they cannot be termed as model in the strict sense. However, these formulas could be useful for modelling turbulent flows.

In the following discussion we will describe Pao's and Pope's models.

5.2 Model description

We start with the flow equations In Fourier space [47]

$$\left(\frac{d}{dt} + \nu k^2\right) \mathbf{u}(\mathbf{k}, t) = -i\mathbf{k} p(\mathbf{k}, t) - i \sum_{\mathbf{k}=\mathbf{p}+\mathbf{q}} \mathbf{k} \cdot \mathbf{u}(\mathbf{q}) \mathbf{u}(\mathbf{p}) + \mathbf{F}_u(\mathbf{k}), \tag{5.6}$$

$$\mathbf{k} \cdot \mathbf{u}(\mathbf{k}) = 0, \tag{5.7}$$

where $\mathbf{u}(\mathbf{k})$, $p(\mathbf{k})$, and $\mathbf{F}_u(\mathbf{k})$ are the Fourier transforms of $\mathbf{u}(\mathbf{r})$, $p(\mathbf{r})$, and $\mathbf{F}_u(\mathbf{r})$ respectively. We multiply the above equation with $\mathbf{u}^*(\mathbf{k}, t)$, and add the resulting equation

to its complex conjugate. This process yields

$$\frac{d}{dt}E_u(\mathbf{k}) = \sum_{\mathbf{p}} \Im\left[\{\mathbf{k}\cdot\mathbf{u}(\mathbf{q})\}\{\mathbf{u}(\mathbf{p})\cdot\mathbf{u}^*(\mathbf{k})\}\right] + \mathrm{Re}[\mathbf{F}_u(\mathbf{k})\cdot\mathbf{u}^*(\mathbf{k})] - 2\nu k^2 E_u(\mathbf{k}),$$

where $E_u(\mathbf{k}) = |\mathbf{u}(\mathbf{k})|^2/2$. We sum the above equation for all the modes in shell k, i.e., $(k-1:k)$. The above equations yield the following equation for one-dimensional energy spectrum $E_u(k)$ [47]:

$$\frac{\partial E_u(k,t)}{\partial t} = T_u(k,t) - 2\nu k^2 E_u(k,t) + \mathcal{F}(k,t). \tag{5.8}$$

Note that the one-dimensional energy spectrum $E_u(k)$ is

$$E_u(k) = \sum_{k-1<k'\leq k} \frac{1}{2}|\mathbf{u}(\mathbf{k}')|^2, \tag{5.9}$$

$T(k,t) = d\Pi_u(k)/dk$ is the energy transfer to the wavenumber shell k due to nonlinearity, $\mathcal{F}(k)$ is the energy feed by the force, and $-2\nu k^2 E_u(k)$ is the dissipation spectrum [47, 98]. The energy flux $\Pi(k)$, rate of energy transfer from a wavenumber sphere of radius k, is computed using the following formula [21, 88]:

$$\Pi(k) = \sum_{k'>k}\sum_{p\leq k} \delta_{\mathbf{k}',\mathbf{p}+\mathbf{q}}\mathrm{Im}([\mathbf{k}'\cdot\mathbf{u}(\mathbf{q})][\mathbf{u}^*(\mathbf{k}')\cdot\mathbf{u}(\mathbf{p})]). \tag{5.10}$$

In Kolmogorov's model, the force feed $\mathcal{F}(k)$ is active at large length scales (for $k < k_f$, where k_f is the forcing wavenumber), and it is absent in the inertial and dissipative range. Therefore, $\partial E_u(k,t)/\partial t = 0$ during a steady state. Hence, for a steady-state, the energy flux $\Pi_u(k)$ varies with k as [47, 98]

$$\frac{d\Pi_u(k)}{dk} = -2\nu k^2 E_u(k). \tag{5.11}$$

The turbulence models of Pao [62] and Pope [67] are based on the above equation.

It is more convenient to work with non-dimensionalized version of the above equation. The wavenumber k is non-dimensionalized using the Kolmogorov wavenumber k_d, which is defined as

$$k_d = \left(\frac{\epsilon}{\nu^3}\right)^{1/4}. \tag{5.12}$$

The energy flux $\Pi_u(k)$ is nondimensionalized using ϵ as its scale. Hence, following

Eq. (5.3), we obtain

$$\tilde{k} = \frac{k}{k_d}, \tag{5.13}$$

$$\tilde{\Pi}(\tilde{k}) = \frac{\Pi_u(k)}{\epsilon}, \tag{5.14}$$

$$\tilde{E}(\tilde{k}) = \frac{E_u(k)}{\epsilon^{2/3}k^{-5/3}} = K_{\mathrm{Ko}} f_\eta(\tilde{k}). \tag{5.15}$$

Substitution of the above variables in Eq. (5.11) yields

$$\frac{d\tilde{\Pi}(\tilde{k})}{d\tilde{k}} = -2K_{\mathrm{Ko}}\tilde{k}^{1/3} f_\eta(\tilde{k}). \tag{5.16}$$

In the next subsections, we discuss the models of Pao [62] and Pope [67]. The energy fluxes and spectra of these models are labelled differently. Variables in Pao's model such as the energy spectrum, energy flux, nondimensional dissipative function, etc. are labelled with superscript (1). Thus they are respectively written as $\tilde{E}^{(1)}(\tilde{k})$, $\tilde{\Pi}^{(1)}(\tilde{k})$ and $f_\eta^{(1)}(\tilde{k})$; for Pope's model [67] we use the superscript (2) for the corresponding quantities.

5.2.1 Pao's model of turbulent flow

In Eq. (5.16), $\tilde{\Pi}(\tilde{k})$ and $f_\eta(\tilde{k})$ are two unknown functions. Hence, to close the equation, Pao [62] assumed that $E_u(k)/\Pi_u(k)$ is independent of ν, and depends only on ϵ and k, or

$$\frac{E_u(k)}{\Pi_u(k)} \sim \epsilon^{-1/3}k^{-5/3}. \tag{5.17}$$

Hence, Eq. (5.15) implies that

$$\tilde{\Pi}^{(1)}(\tilde{k}) = \frac{1}{K_{\mathrm{Ko}}}\tilde{E}^{(1)}(\tilde{k}) = f_\eta(\tilde{k}). \tag{5.18}$$

In other words, the dissipative spectrum for both $E_u(k)$ and $\Pi_u(k)$ should be of the same form. Thus $\Pi_u(k) = \epsilon f_\eta(k)$ and $E_u(k) = K_{\mathrm{Ko}}\epsilon^{2/3}k^{-5/3}f_\eta(k)$, substitution of

which in Eq. (5.16) yields

$$f_\eta^{(1)}(\tilde{k}) = \exp\left(-\frac{3}{2}K_{\mathrm{Ko}}\tilde{k}^{4/3}\right), \tag{5.19}$$

$$\tilde{\Pi}^{(1)}(\tilde{k}) = \exp\left(-\frac{3}{2}K_{\mathrm{Ko}}\tilde{k}^{4/3}\right), \tag{5.20}$$

$$\tilde{E}^{(1)}(\tilde{k}) = K_{\mathrm{Ko}}\exp\left(-\frac{3}{2}K_{\mathrm{Ko}}\tilde{k}^{4/3}\right). \tag{5.21}$$

We can rewrite Eq. (5.11) as

$$\begin{aligned}\frac{d\Pi_u(k)}{dk} &= -2\nu K_{\mathrm{Ko}}k^2k^{-5/3}[\epsilon]^{2/3}f_\eta(k)\\ &= -2\nu K_{\mathrm{Ko}}k^2k^{-5/3}[\epsilon f_\eta(k)]^{2/3}[f_\eta(k)]^{1/3}\\ &= -2\nu K_{\mathrm{Ko}}k^2k^{-5/3}[\Pi_u(k)]^{2/3}[f_\eta(k)]^{1/3}.\end{aligned} \tag{5.22}$$

In the above equation $\Pi_u(k)$ can be interpreted as a variable energy flux. Verma [89, 99] utilised this interpretation to compute energy spectrum and flux for the two-dimensional flows with Ekman friction, and for quasi-static MHD turbulence.

In the next subsection we discuss Pope's model for turbulent flow.

5.2.2 Pope's model of turbulent flow

Another popular model for the turbulent flow is by Pope [67]. For this model, we denote the energy spectrum, flux, nondimensional dissipative function $f_\eta(\tilde{k})$ as $\tilde{E}^{(2)}(\tilde{k})$, $\tilde{\Pi}^{(2)}(\tilde{k})$ and $f_\eta^{(2)}(\tilde{k})$ respectively. Pope [67] proposed that

$$E^{(2)}(k) = K_{\mathrm{Ko}}\epsilon^{2/3}k^{-5/3}f_L(kL)f_\eta^{(2)}(k/k_d) \tag{5.23}$$

with the functions $f_L(kL)$ and $f_\eta(k\eta)$ specifying the large-scale and dissipative-scale components, respectively:

$$f_L(kL) = \left(\frac{kL}{[(kL)^2+c_L]^{1/2}}\right)^{5/3+p_0}, \tag{5.24}$$

$$f_\eta^{(2)}(\tilde{k}) = \exp\left[-\beta\left\{[\tilde{k}^4+c_\eta^4]^{1/4}-c_\eta\right\}\right], \tag{5.25}$$

where the c_L, c_η, p_0, β are constants. Since we focus on the inertial and dissipative ranges, we set $f_L(kL) = 1$. In the high Reynolds number limit, $c_\eta \approx 0.47\beta^{1/3}/K_{\mathrm{Ko}}$ [67]. We refer to Pope [67] for the detailed derivation of above relation. We keep $\beta = 5.2$

as prescribed by Pope [67]. Substitution of $\tilde{E}^{(2)}(\tilde{k})$ and $\tilde{\Pi}^{(2)}(\tilde{k})$ in Eq. (5.16) yields the following solution

$$\tilde{\Pi}^{(2)}(\tilde{k}) = \tilde{\Pi}^{(2)}(\tilde{k}_0) - 2K_{\mathrm{Ko}} \int_{\tilde{k}_o}^{\tilde{k}} \tilde{k}'^{1/3} f_\eta^{(2)}(\tilde{k}') d\tilde{k}', \tag{5.26}$$

which is solved numerically given $f_\eta^{(2)}(\tilde{k})$ of Eq. (5.25). We set $\tilde{\Pi}^{(2)}(\tilde{k}_0) = 1$ at small k_0.

In the next section, we will compare the predictions of the above two models with those from direct numerical simulation.

5.3 Numerical Validation of the Models

We solve Eqs. (1.5, 1.6) using Tarang. We carry out the simulations on 512^3, 1024^3, and 4096^3 grids and employ periodic boundary conditions on all sides of a cubical box of size $2\pi \times 2\pi \times 2\pi$. We use the fourth-order Runge-Kutta scheme for time advancement with variable Δt, which is chosen using the CFL condition. We dealiase using the 2/3 rule.

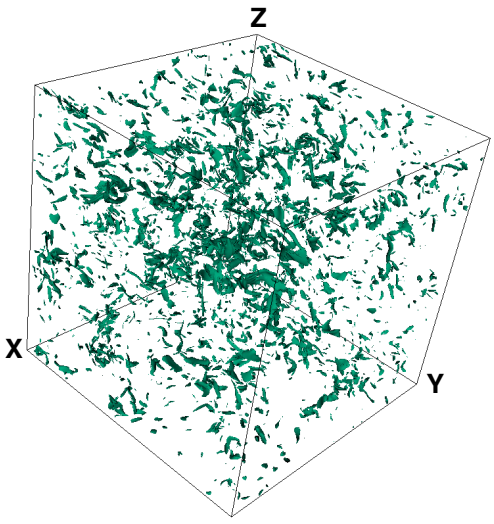


FIGURE 5.1: Isosurface of the contours of constant vorticity $|\nabla \times \mathbf{u}|$ (30% of the maximum value). The figure indicates regions of strong vorticity in the flow.

To obtain a steady turbulent flow, we apply random forcing [69] in the wavenumber band $2 \leq k \leq 4$ for 1024^3 and 2048^3 grids, but in the band $1 \leq k \leq 3$ for 512^3 grid. We choose random initial conditions for the 512^3-grid simulation. The steady-state data of 512^3 was used as an initial condition for the 1024^3-grid run, whose steady-state

data is used for 4096^3-grid simulation. In all the three cases, the small scales are well resolved as $k_{\max}\eta$ is always greater than 1.5, where $k_{\max}$ is the highest wavenumber represented by the grid points, and $\eta \sim 1/k_d$ is the Kolmogorov's length. The Reynolds numbers for the 512^3, 1024^3, and 4096^3 grid simulations are 5.7×10^3, 1.4×10^4, and 6.8×10^4 respectively. We observe that the energy flux in the inertial range, the energy supply rate, and the energy supply rate by the forcing match with each other within 2-3%. Also note that $\epsilon \approx u_{\rm rms}^3/L$, where L is the box size. The parameters of our runs for turbulent flows are described in Table 5.1. We compute the energy spectra and fluxes of all the simulations at steady state.

TABLE 5.1: Parameters of our direct numerical simulations (DNS) for turbulent flow: grid resolution; kinematic viscosity ν, Reynolds number Re, Kolmogorov constant $K_{\rm Ko}$, Kolmogorov wavenumber k_d, $k_{max}\eta$, and $\epsilon/(u_{\rm rms}^3/L)$. For all our runs the energy supply rate is 0.1, and the energy dissipation rate $\epsilon \approx 0.1$ with 2-3% error. In the Table, we report the value of $\epsilon/(u_{\rm rms}^3/L)$ which is approximately unity for all three simulations.

Grid	ν	Re	Re_λ	$K_{\rm Ko}$	k_d	$k_{max}\eta$	$\epsilon/(u_{\rm rms}^3/L)$
512^3	10^{-3}	5.7×10^3	106	2.2 ± 0.3	4.1×10^1	2.5	0.9
1024^3	4×10^{-4}	1.4×10^4	139	2.2 ± 0.4	9.5×10^1	2.4	1.0
4096^3	8×10^{-5}	6.8×10^4	374	1.8 ± 0.2	3.2×10^2	3.1	1.0

Fig. 5.1 exhibits an isosurface of the contours of constant magnitudes of the vorticity under steady state. The intense region in the figure indicate regions where the vorticity is strong.

Figure 5.2(a, b, c) shows the normalized spectrum $\tilde{E}(\tilde{k})/K_{\rm Ko}$ for the 512^3, 1024^3, and 4096^3 grid simulations. Note that the gray shaded region in the figure represents the forcing band. The figure shows that the DNS results are consistent with the predictions of both Pao and Pope. We also compute the Kolmogorov's constant $K_{\rm Ko}$ using

$$K_{\rm Ko} = \frac{E_u(k)k^{5/3}}{\epsilon^{2/3}} \tag{5.27}$$

in the inertial range. As shown in Table 5.1, the values of $K_{\rm Ko}$ varies from 2.2 to 1.8 with errors in the range of 11% to 18%. This value is in the same range as other numerical values of $K_{\rm Ko}$ reported earlier [81, 106, 34, 56, 27]. The estimate of $K_{\rm Ko}$ in DNS appears to be slightly larger than its theoretical estimate, which is approximately 1.5 [44, 101]. Note that for model predictions by Pao and Pope, we take $K_{\rm Ko} = 2.2$ for 512^3 and 1024^3

simulations, and $K_{\mathrm{Ko}} = 1.8$ for 4096^3 simulation.

In Table 5.1 we see that as the grid size increases, we are able to reach higher Re. The maximum Re one can reach for a given grid size scales as $\mathrm{Re} \approx N^{4/3}$ and Re for 1024^3 scales exactly according to this. The Reynolds number of 4096^3 is around 5 times larger than that for 1024^3, which is somewhat consistent with this formula. Note that $k_{max}\eta = 3.1$ for 4096^3 grid. Hence, we could increase our Re to possibly double the present value.

A careful examination of the normalized spectrum indicates a hump in $\tilde{E}(\tilde{k})/K_{\mathrm{Ko}}$ near the transition region between the inertial range and dissipation range ($0.04 \lesssim \tilde{k} \lesssim 0.2$), which is due to the bottleneck effect [76, 31, 50, 106, 24, 94, 27]. The predicted values of $\tilde{E}(\tilde{k})/K_{\mathrm{Ko}}$ by the models of Pao and Pope always decrease with $\tilde{k}$; hence they do not capture the above hump.

In Fig. 5.2(d, e, f) we plot the non-dimensionalized energy flux $\tilde{\Pi}(\tilde{k})$ computed using the DNS data. We observe that $\tilde{\Pi}(\tilde{k})$ is approximately constant in the inertial range, consistent with the Kolmogorov's theory [43]. In the same plot, we present the energy fluxes computed using the Pao's and Pope's models [Eqs. (5.20, 5.26)]. In the inertial range, the predictions of both the models are in good agreement with the DNS results. In the dissipation range, the predictions of Pao's model is slightly larger than the numerical values of $\tilde{\Pi}(\tilde{k})$, but the predictions of Pope's model is lower than the numerical value. Thus, the models have some deficiencies, especially for predicting the energy flux.

Thus, we see that Pao's [62] and Pope's [67] model are in good agreement with the numerical results of high-resolution DNS. However, in the dissipation range, there are small differences between the model predictions and numerical values of the energy flux.

The energy transfers, according to Kolmogorov, among the inertial-range wavenumber shells are forward and local [25, 108, 92]. That is, a wavenumber shell (say m) transfers maximal energy to its nearest forward neighbour $(m+1)$ and receives maximal energy from its previous neighbour $(m-1)$. This phenomenon has been verified using various numerical simulations. However, there is only a handful of shell-to-shell energy transfer computation for the dissipative regime [25, 102]. In this thesis, we carry forward this work to a very large resolution simulation. In the following discussion, we show that the shell-to-shell energy transfers in the inertial and dissipative regimes of a turbulent flow are forward and local.

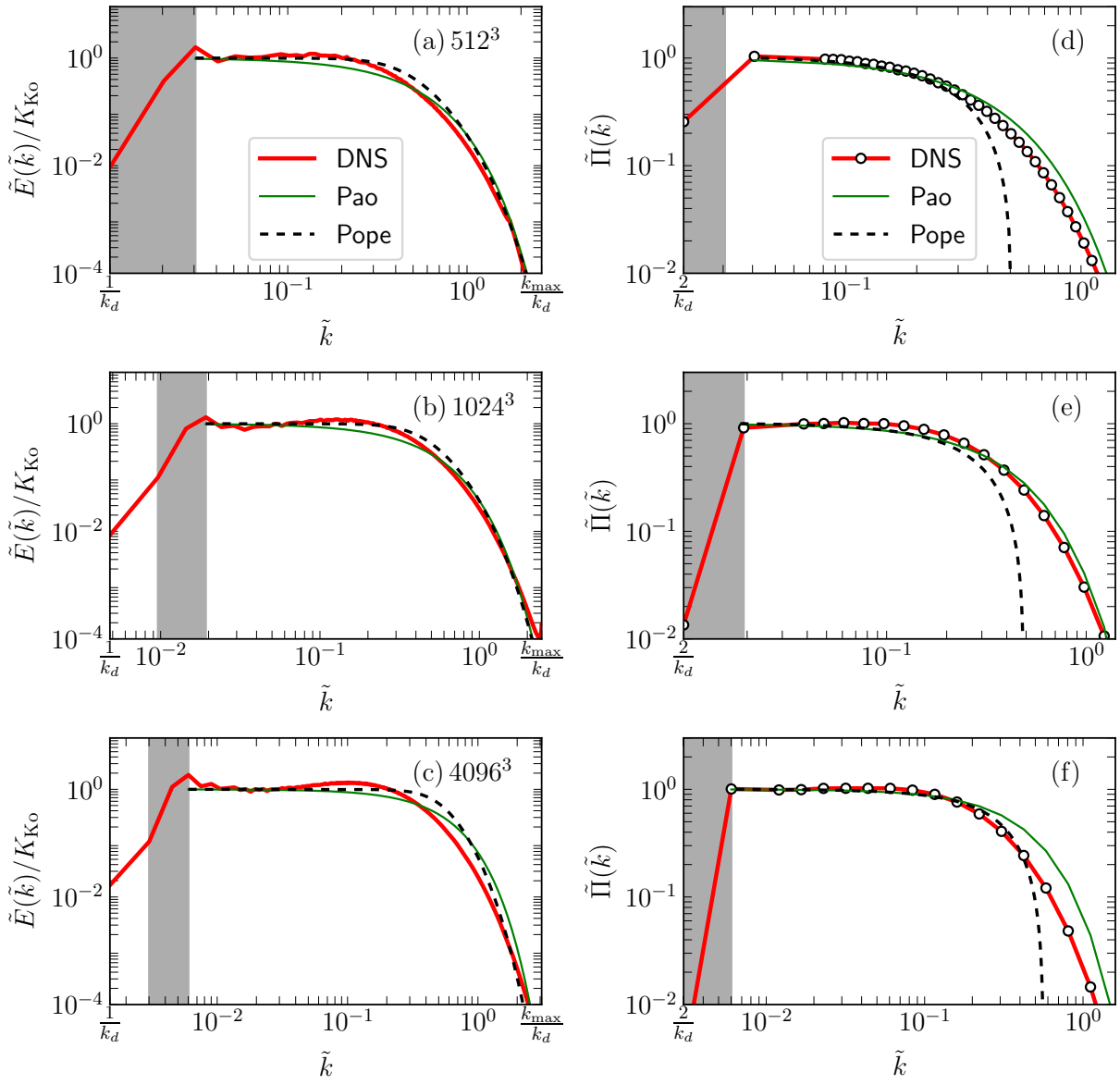


FIGURE 5.2: For the grid resolutions of 512^3, 1024^3, and 4096^3: (a,b,c) plots of the normalized energy spectrum $\tilde{E}(\tilde{k})/K_{\mathrm{Ko}}$ vs. $\tilde{k}$; (d,e,f) plots of normalized energy flux $\tilde{\Pi}(\tilde{k})$ vs. $\tilde{k}$. See Eqs. (5.14) and Eq. (5.15) for definitions. The plots include the spectra and fluxes computed using numerical data (thick solid line), and the model prediction of Pao (thin solid line) and Pope (dashed line). Taken from [97].

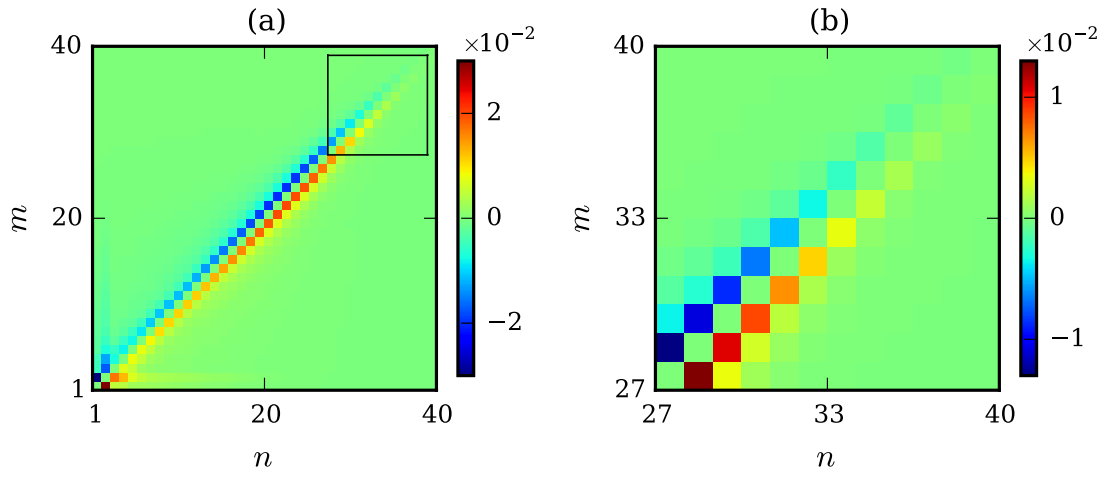


FIGURE 5.3: For the turbulent simulation on 4096^3 grid: the shell-to-shell energy transfer rate (a) for the whole wavenumber range, (b) for the dissipative range corresponding to the boxed region of subfigure (a). Here m denotes the giver shell, while n denotes the receiver shell. Our results indicate forward and local energy transfers in the inertial as well as in the dissipative wavenumber range. Taken from [97].

We study the properties of shell-to-shell energy transfers for the numerical data of 4096^3 grid. For the same, we divide the Fourier space into 40 shells, whose centers are at the origin $\mathbf{k} = (0, 0, 0)$. The inner and outer radii of the shells are k_{n-1} and k_n respectively, where $k_n = \{0, 2, 4, 8, 8 \times 2^{s(n-3)}, ..., 2048\}$, with $s = (7/35)$. The shells are logarithmically binned [23]. Note that the 27th shell, whose wavenumber range is $194 \leq k \leq 223$, separates the dissipative range from the inertial range. In Fig. 5.3(a), we exhibit the shell-to-shell energy transfers for the whole range, while Fig. 5.3(b) shows these transfers for the dissipative range only. As expected, shell m gives energy dominantly to shell $m + 1$, and it receives energy from shell $m - 1$ in the inertial regime, hence, the shell-to-shell energy transfers are forward and local [25, 108, 92]. Interestingly, similar behaviour is observed for the wavenumber shells in the dissipative regime as well. This is essentially because the correlations induced by forcing at small wavenumbers is lost deep inside the inertial and dissipative wavenumbers. As a result, the energy transfer is the function of neighbouring Fourier modes, and hence are local.

5.4 Summary

We performed large-resolution DNS of hydrodynamic turbulence using Tarang. Using numerical data we compute $E_u(k)$ and $\Pi_u(k)$. The inertial range $E_u(k)$ is described by Kolmogorov's theory, but $E_u(k)$ for the dissipation range is somewhat uncertain. Our simulation results show that Pao's model with $k^{-5/3}\exp(-k^{4/3})$ describes $E_u(k)$ in the inertial and dissipation ranges quite well. Also, we verify that the shell-to-shell energy transfer is local and forward. These results are important for hydrodynamic turbulence.

Chapter 6

Simulation of turbulent thermal convection

Turbulent convection is observed in the interiors and atmosphere of stars and planets, and hence its understanding is critical for modeling such systems. The complexity arising due to buoyancy and boundary conditions have challenged scientists and engineers for more than a century. Fortunately, modern supercomputers have enabled us to decipher the physics of turbulent convection in a significant way. Using a state-of-the-art pseudospectral code on one of the fastest supercomputers of modern times, we deduce that turbulent convection has a behavior similar to fluid turbulence, i.e., it exhibits $k^{-5/3}$ energy spectrum with a constant energy flux. Also, the shell-to-shell energy transfer is local and forward, and the energy distribution is nearly isotropic as in fluid turbulence. This is the topic of the present chapter. Parts of the results presented here are taken from [45, 12]. For completeness, we have included works on spectral properties of turbulent thermal convection [98].

We start our discussion with the equations of Rayleigh-Bénard convection (RBC).

6.1 Governing equations of Rayleigh-Bénard convection

We solve the following nondimensionalized Navier Stokes equation for Rayleigh-Bénard convection (RBC) using direct numerical simulation (DNS):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla\sigma + \theta z + \sqrt{\frac{\mathrm{Pr}}{\mathrm{Ra}}}\nabla^2\mathbf{u}, \tag{6.1}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = u_z + \frac{1}{\sqrt{\mathrm{RaPr}}}\nabla^2\theta, \tag{6.2}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{6.3}$$

where θ is the temperature fluctuation from the steady conduction state (temperature $T = T_c + \theta$ with T_c as the conduction temperature profile), σ is the pressure fluctuation, and z is the buoyancy direction. Here the two important nondimensional parameters are the Rayleigh number

$$\mathrm{Ra} = \alpha g \Delta d^3/\nu\kappa, \tag{6.4}$$

and the Prandtl number

$$\mathrm{Pr} = \nu/\kappa, \tag{6.5}$$

where Δ is the temperature difference between the two plates separated by a distance d, and ν, κ, α are the fluid's kinematic viscosity, thermal diffusivity, and thermal expansion coefficient respectively, and g is the acceleration due to gravity.

We solve the following nondimensionalized RBC equations (6.1-6.3) using direct numerical simulation:

$$\left(\partial_t + \sqrt{\frac{\mathrm{Pr}}{\mathrm{Ra}}}k^2\right) u_j(\mathbf{k},t) = -ik_l u_l u_j(\mathbf{k},t) - ik_j p(\mathbf{k},t) + \theta(\mathbf{k})\delta_{jz}, \tag{6.6}$$

$$\left(\partial_t + \sqrt{\frac{1}{\mathrm{RaPr}}}k^2\right) \theta(\mathbf{k},t) = -ik_l u_l \theta(\mathbf{k},t) + u_z(\mathbf{k}), \tag{6.7}$$

$$k_j u_j(\mathbf{k}) = 0, \tag{6.8}$$

where $i = \sqrt{-1}$. These equations are time advanced using a time stepping method, e.g., Runge Kutta scheme. The nonlinear terms $u_l u_j$ and $u_l\theta$ transform to convolutions in spectral space.

We solve the above equations using the following boundary condition. For the velocity field, the free-slip boundary condition at the top and bottom plates, and pe-

riodic boundary condition at the side walls. For the temperature field, we use conducting boundary condition at the top and bottom plates, and the periodic boundary condition at the side walls.

The above nonlinear terms yield energy and entropy transfers from one scale to another leading to energy and entropy fluxes. One of the important problems of turbulent convection is the nature of energy and entropy spectrum and their respective fluxes, which will be discussed in the next section.

6.2 Phenomenology of turbulent convection

Following the same procedure as in Sec. 5.2, we derive the time-evolution equation for $E_u(k)$ for RBC [98]

$$\frac{\partial E_u(k)}{\partial t} = T_u(k) + F_B(k) + F_{\text{ext}}(k) - D(k), \tag{6.9}$$

where $T_u(k)$ is the energy transfer rate to the shell k due to nonlinear interaction, and $F_B(k)$ and $F_{\text{ext}}(k)$ are the energy supply rates to the shell from the buoyancy and external forcing $\mathbf{F}_u$ respectively, i.e.,

$$F_B(k) = -\sum_{|\mathbf{k}|=k} \alpha g \text{Re}\langle u_z(\mathbf{k})\theta^*(\mathbf{k})\rangle, \tag{6.10}$$

$$F_{\text{ext}}(k) = \sum_{|\mathbf{k}|=k} \text{Re}\langle \mathbf{u}(\mathbf{k}) \cdot \mathbf{F}_u{}^*(\mathbf{k})\rangle. \tag{6.11}$$

For ease in calculation, we set $\rho_m = 1$. In Eq. (6.9), $D(k)$ is the viscous dissipation rate defined by

$$D(k) = \sum_{|\mathbf{k}|=k} 2\nu k^2 E_u(k). \tag{6.12}$$

The kinetic energy flux ($\Pi_u(k_0)$), defined as the kinetic energy (KE) leaving a wavenumber sphere of radius k_0 due to nonlinear interactions, is related to the nonlinear interaction term $T_u(k)$ as

$$\Pi_u(k) = -\int_0^k T_u(k)dk. \tag{6.13}$$

In steady state ($\partial E_u(k)/\partial t = 0$), using Eqs. (6.9) and (6.13), we deduce that

$$\frac{d}{dk}\Pi_u(k) = F_B(k) + F_{\text{ext}}(k) - D(k) \tag{6.14}$$

or

$$\Pi_u(k + \Delta k) = \Pi_u(k) + [F_B(k) + F_{\text{ext}}(k) - D(k)]\Delta k. \tag{6.15}$$

Computer simulations of the KE flux $\Pi_u(k_0)$ is computed by using the formula of Eq. (5.10). Similarly, the entropy flux $\Pi_\theta(k_0)$ is computed as the entropy leaving a wavenumber sphere of radius k_0, i.e.

$$\Pi_\theta(k_0) \;=\; \sum_{k>k_0}\sum_{p\le k_0} \delta_{\mathbf{k},\mathbf{p}+\mathbf{q}}\Im([\mathbf{k}\cdot\mathbf{u}(\mathbf{q})][\theta^*(\mathbf{k})\theta(\mathbf{p})]). \tag{6.16}$$

One of the most interesting problems in the field of buoyancy driven turbulence is the scaling of energy spectrum and flux [51, 71]. In the next section, we will review some of the theoretical results obtained for the aforementioned topic.

6.2.1 Classical Bolgiano-Obukhov scaling for stably-stratified turbulence (SST):

Kolmogorov [42, 43] first proposed a phenomenology for the inertial range of isotropic hydrodynamic turbulence, according to which the energy spectrum in the inertial range is independent of the fluid properties and nature of large-scale forcing. He showed that the one-dimensional energy spectrum $E(k) = K_{Ko}\Pi_u^{2/3}k^{-5/3}$ in the inertial range of wavenumbers, where $\Pi_u(k)$ is the constant energy flux, and K_{Ko} is the Kolmogorov's constant.

Note that Kolmogorov's theory may not work for the buoyancy-driven turbulence because buoyant force acts at all scales. In this section we show that the buoyancy affects the energy spectra and fluxes of the buoyancy-driven flows. For stable stratification, Bolgiano [4] and Obukhov [60] argued that the KE flux $\Pi_u(k)$ is depleted at different length scales due to the conversion of KE to PE via buoyancy ($u_z\alpha g\theta$). Subsequently, $\Pi_u(k)$ decreases with k, and $E_u(k)$ is steeper than that predicted by Kolmogorov's theory; we refer to the above as *BO phenomenology or scaling*. According to this phenomenology, for $L_B \ll l \ll L$, buoyancy is important and the buoyancy term is balanced by the nonlinear term [$\alpha g\theta \approx (\mathbf{u}\cdot\nabla)\mathbf{u}$]. Here L_B is the Bolgiano scale [4] and L is the large length scale or the box size. The force balance at wavenumber $k = 1/l$

yields

$$\alpha\theta_k g \approx k u_k^2. \tag{6.17}$$

According to BO phenomenology, PE has a constant flux, i.e., $\Pi_\rho \approx k u_k \rho_k^2 \approx \epsilon_\theta$. Hence,

$$u_k \approx \epsilon_\theta^{1/5} (\alpha g)^{2/5} k^{-3/5}, \tag{6.18}$$
$$\rho_k \approx \epsilon_\theta^{2/5} (\alpha g)^{-1/5} k^{-1/5}. \tag{6.19}$$

Therefore, the KE spectrum $E_u(k) \approx u_k^2/k$, PE spectrum $E_\rho(k) \approx \rho_k^2/k$, and $\Pi_u(k) \approx u_k^3 k$ are

$$E_u(k) = c_1 \epsilon_\theta^{2/5} (\alpha g)^{4/5} k^{-11/5}, \tag{6.20}$$
$$E_\rho(k) = c_2 \epsilon_\theta^{4/5} (\alpha g)^{-2/5} k^{-7/5}, \tag{6.21}$$
$$\Pi_u(k) = c_3 \epsilon_\theta^{3/5} (\alpha g)^{6/5} k^{-4/5}, \tag{6.22}$$
$$\Pi_\theta(k) = \epsilon_\theta, \tag{6.23}$$

where c_i's are constants. At smaller length scales ($k > k_B$), where $k_B = 1/L_B$ is the Bolgiano wavenumber, Bolgiano [4] and Obukhov [60] argued that the buoyancy is relatively weak, hence Kolmogorov-Obukhov (KO) scaling is valid in this regime, i.e.,

$$E_u(k) = K_{Ko} \epsilon_u^{2/3} k^{-5/3}, \tag{6.24}$$
$$E_\rho(k) = K_{Ba} \epsilon_u^{-1/3} \epsilon_\theta k^{-5/3}, \tag{6.25}$$
$$\Pi_u(k) = \epsilon_u, \tag{6.26}$$
$$\Pi_\theta(k) = \epsilon_\theta, \tag{6.27}$$

where K_{Ba} is the Batchelor's constant. A comparison of $\Pi_u(k)$ of Eq. (6.22) with that of Eq. (6.26) yields the crossover wavenumber k_B as

$$k_B \approx (\alpha g)^{3/2} \epsilon_u^{-5/4} \epsilon_\theta^{3/4}. \tag{6.28}$$

The associated length, the Bolgiano length, is $L_B = (2\pi)/k_B$.

The BO phenomenology implicitly assumes isotropy in Fourier space, which is a tricky assumption. For BO scaling in stably-stratified turbulence, the gravity must be strong enough to compete with the nonlinear term $\mathbf{u} \cdot \nabla \mathbf{u}$, but not too strong to make the flow quasi two-dimensional (quasi-2D). This is achieved when Froude number Fr ≈ 1 regime. Kumar *et al.* [45] verified this scaling using a detailed DNS. We also remark that a large number of earlier explorations in SST [49, 7, 1] have been for

Fr $\ll 1$ regime quasi-2D flow, and not BO scaling. These topics are beyond the scope of this thesis

6.2.2 Generalization of Bolgiano-Obukhov scaling to RBC:

Procaccia and Zeitak [68], L'vov [52], and L'vov and Falkovich [53] employed mean field theory and deduced that the Bolgiano-Obukhov scaling is applicable to convective turbulence. They argued that the kinetic energy is converted to the potential energy and therefore favored BO scaling. Rubinstein [75] employed renormalization group analysis to RBC and observed that the renormalized viscosity $\nu(k) \sim k^{-8/5}$, $E_u(k) \sim k^{-11/5}$, and $E_\rho(k) \sim k^{-7/5}$, thus claiming BO scaling for RBC.

The aforementioned theories made a major impact in the field of turbulent convection, and a large number of analytical, experimental, and numerical works attempted to verify these ideas. Lohse and Xia [51] reviewed whether BO scaling is indeed present in RBC; they studied the experimental, theoretical, and numerical results and argued that it is difficult to conclude the applicability of BO scaling in RBC. Recently Kumar *et al.* [45] and Verma *et al.* [98] showed that the BO scaling does not describe RBC turbulence since the energy supply by buoyancy in RBC is very different from that in stably stratified flow. We will provide these arguments below.

6.2.3 A phenomenological argument based on kinetic energy flux:

Kumar *et al.* [45] and Verma *et al.* [96, 95, 98] constructed a phenomenological argument based on the KE flux to derive a spectral theory of buoyancy-driven turbulence. Equation (6.15) provides important clues on the energy spectrum and flux of the buoyancy-driven flows. Here we list three possibilities for the inertial range ($k_f < k < k_d$), where k_f is the forcing wavenumber, and k_d is the dissipation wavenumber:

1. For hydrodynamic turbulence, in the inertial range, $F_B(k) = 0$ and $D(k) \to 0$, therefore $\Pi_u(k + \Delta k) \approx \Pi_u(k)$ and $E_u(k) \sim k^{-5/3}$, consistent with the Kolmogorov's theory [43]. Note that $F_{ext}(k) = 0$ in the inertial range. This is described in Chapter 5.

2. For the stably stratified flows, as argued by Bolgiano [4] and Obukhov [60], in the $k_f < k < k_B$ wavenumber band, buoyancy converts the kinetic energy of the flow to the potential energy, i.e., $F_B(k) < 0$. Hence, Eq. (6.15) predicts that $\Pi_u(k)$

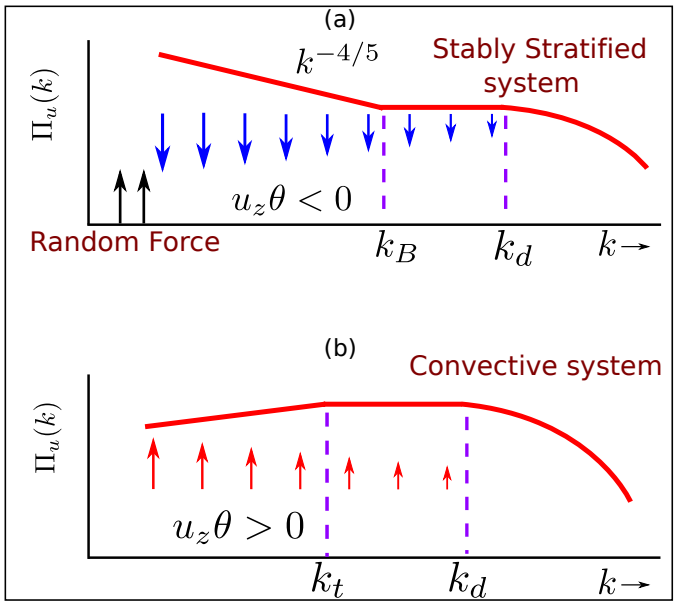


FIGURE 6.1: Schematic diagrams of the kinetic energy flux $\Pi_u(k)$ for the stably stratified system and convective system. (a) In stably stratified flows, $\Pi_u(k)$ decreases with k due to the negative energy supply rate $F_B(k)$. (b) In convective system, $F_B(k) > 0$, hence $\Pi_u(k)$ first increases for $k < k_t$ where $F_B(k) > D(k)$, then $\Pi_u(k) \approx$ constant for $k_t < k < k_d$ where $F_B(k) \approx D(k)$; $\Pi_u(k)$ decreases for $k > k_d$ where $F_B(k) < D(k)$. From Kumar *et al.* [45].

decreases with k, as shown in Fig. 6.1(a). In the wavenumber range, $k_B < k < k_d$, buoyancy becomes weaker, hence $\Pi_u(k) \approx$ constant.

3. For RBC in three dimensions, buoyancy feeds the kinetic energy, hence $F_B(k) > 0$. Therefore the KE flux $\Pi_u(k)$ increases with k. Numerical simulation of Kumar *et al.* [45] and Verma *et al.* [98] show that the energy supplied by buoyancy is dissipated by the viscous force, i.e., $F_B(k) \approx D(k)$. Hence $\Pi_u(k) \approx$ constant and they are much less than the average strength of $|\nabla\sigma|$. This is the reason for the Kolmogorov's spectrum for RBC in 3D. Note that L'vov [52] argued that $F_B(k) < 0$, which is not the case in 3D RBC with Pr ≈ 1.

6.3 Structure functions of turbulent convection

The scaling relations are also presented using the variables $\delta u_\parallel(l)$ and $\delta\theta(l)$, which are defined as

$$\delta u_\parallel(l) = [\mathbf{u}(\mathbf{x}+\mathbf{l}) - \mathbf{u}(\mathbf{x})] \cdot \frac{\mathbf{l}}{l}, \tag{6.29}$$

$$\delta\theta(l) = \theta(\mathbf{x}+\mathbf{l}) - \theta(\mathbf{x}), \tag{6.30}$$

and the structure function for the velocity and temperature fluctuations, which are defined as

$$S_q^u(l) = \langle[\delta u_\parallel(l)]^q\rangle, \tag{6.31}$$

$$S_q^\theta(l) = \langle[\delta\theta(l)]^q\rangle, \tag{6.32}$$

where $\langle.\rangle$ represent the ensemble average. Using scaling analysis similar to that given in Sec. 6.2.1, it can be derived that [14]

$$S_q^u(l) = \langle\epsilon_\theta\rangle^{q/5}(\alpha g)^{2q/5} l^{3q/5}, \tag{6.33}$$

$$S_q^\theta(l) = \langle\epsilon_\theta\rangle^{2q/5}(\alpha g)^{-q/5} l^{q/5} \tag{6.34}$$

for $l > L_B$, and

$$S_q^u(l) = \langle\epsilon_u\rangle^{q/3} l^{q/3}, \tag{6.35}$$

$$S_q^\theta(l) = \langle\epsilon_u\rangle^{-q/6}\langle\epsilon_\theta\rangle^{q/2} l^{q/3} \tag{6.36}$$

for $l < L_B$. Note that l correspond to $1/k$, and $\delta u_\parallel(l) \to u_k$.

The above arguments indicate that the structure functions for the fluctuations of RBC in 3D for Pr ≈ 1 may follow the following scaling relations:

$$S_q^u(l) = \langle \epsilon_u \rangle^{q/3} l^{q/3}, \tag{6.37}$$

$$S_q^\theta(l) = \langle \epsilon_u \rangle^{-q/6} \langle \epsilon_\theta \rangle^{q/2} l^{q/3}. \tag{6.38}$$

The above relations need to be tested using numerical simulation and experiments. Ching [13, 14] and Ching *et al.* [15] studied the structure functions for the velocity and temperature fluctuations of turbulent convection, and claimed consistency with Bolgiano-Obukhov scaling. Ching *et al.* [15] computed the anomalous scaling for the turbulent RBC.

In the next section, we briefly describe past numerical works on turbulent thermal convection.

6.4 Past work of Convective Turbulence Phenomenology

We review some of the past numerical work on convective turbulence. Kerr [41] simulated RBC with no-slip boundary conditions using Chebyshev-based pseudospectral method for Pr $= 1$ fluid and observed Kolmogorov's spectrum. Note however that the grid spacing in Chebyshev-based pseudospectral method is nonuniform, that makes computation of energy spectrum difficult.

The free-slip basis function that is based on sin and cos functions along the vertical direction does not suffer from the above difficulty. We can compute the shell spectrum based on wavenumber $(k_x^2 + k_y^2 + k_z^2)^{1/2}$ similar to fluid turbulence in a periodic box. Therefore, uniform Fourier basis helps us compute the energy flux as well as shell-to-shell and ring-to-ring energy transfers. These are some of the advantages of the RBC simulations with the free-slip boundary condition, which is what we employ in Tarang.

In 2006, Rincon [72] performed a numerical simulation using higher order finite-difference scheme for Ra $= 10^6$ and Pr $= 1$ with free-slip boundary conditions, but they could not differentiate between the spectral indices 5/3 and 11/5. Mishra and Verma [58] showed that convective turbulence for zero and small Prandtl numbers follows Kolmogorov's spectrum, but their results were inconclusive for unit and larger Prandtl numbers. Recently, Kumar *et al.* [45] and Verma *et al.* [98] simulated convective turbulence for Pr $= 1$ and proved using flux and energetics argument that the flow

shows Kolmogorov's $k^{-5/3}$ spectrum. However, the range of the spectrum over which the $-5/3$ scaling was observed was rather small to be conclusive.

Scheel and Schumacher [77] simulated RBC for Ra $= 10^{11}$ using a spectral-element code Nek5000 with 2,374,400 elements and 11th order polynomials within each element. Scheel and Schumacher [78] and Schumacher *et al.* [79] simulated low Prandtl number RBC (Pr $= 0.021$ and Pr $= 0.005$) with 6.27 million spectral elements and 13th order polynomials within each element. They computed for Ra upto 4×10^8 and reached Re $= 4.6 \times 10^4$, where Re is the Reynolds number based on large scale velocity. Stevens *et al.* [82] simulated the flow using a finite-difference code on $1081 \times 301 \times 2049$ grid for Rayleigh number of 2×10^{12}. Van der Poel *et al.* [84] simulated RBC for Ra $= 10^{12}$, Pr $= 0.7$ in a $1536 \times 512 \times 2048$ grid. However these works did not examine the energy spectrum.

In the next section we report our numerical results on 4096^3 grid performed using 65536 cores of a Cray XC40. Our RBC computations have employed the largest grid employed till date in order to achieve the largest Rayleigh number Ra $= 1.1 \times 10^{11}$ among spectral simulations. Using these data we compute the energy flux, the shell-to-shell and ring-to-ring energy transfers, and the ring spectrum.

6.5 Deducing physics of convective turbulence using very high-resolution simulations

We perform a high-resolution DNS to study convective turbulence on a 4096^3 grid for Prandtl number Pr $= 1$ and the Rayleigh number Ra $= 1.1 \times 10^{11}$ using 196608 cores of Cray XC40. For the velocity field, we employ free-slip boundary condition at the top and bottom plates, and the periodic boundary condition at the vertical side walls (see Fig. 6.2). For the temperature field, we use conducting boundary conditions at the top and bottom walls, and periodic boundary conditions for the side walls. The box size of our simulation is unity. We use the Runge-Kutta fourth order for time-stepping and dealias using 2/3 rule [6, 10]. We use the exponential trick to absorb the diffusive terms in $\partial u/\partial t$ to overcome the stiffness of the governing equations. To reach 4096^3 grid, first, we perform an RBC simulation on a 512^3 grid until the system reaches a statistically steady state. The final state of the 512^3 grid simulation is used as an initial condition for 1024^3 grid simulation. We repeat this process with higher grids until we reach statistically stable 4096^3 grid. We compute all the diagnostics like energy and

ring spectra, as well as the energy transfer rates using this data of 4096^3 grid.

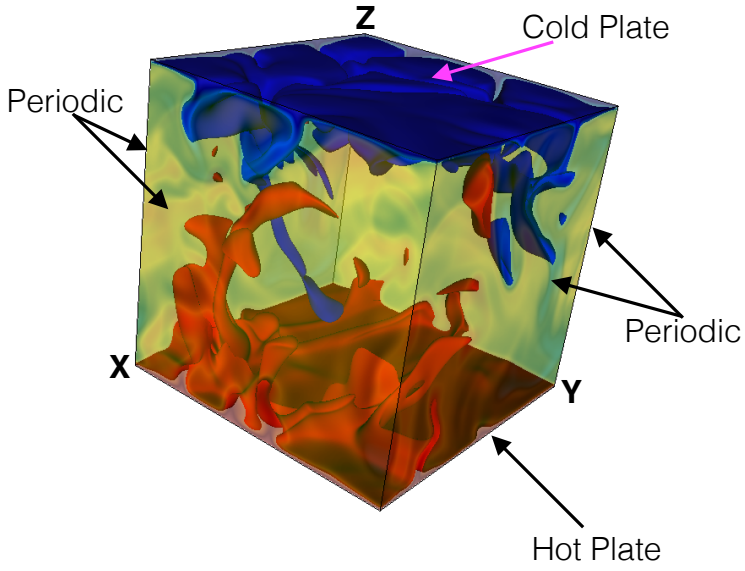


FIGURE 6.2: Isocontours of two constant temperatures. The hot and cold structures of the flow are represented by the red and blue colors respectively.

I remark that several members of our group, mainly Mahendra Verma and Abhishek Kumar, have used the aforementioned numerical data to understand physics of convective turbulence. I have contributed towards the development and optimisation of Tarang, as well as towards the data analysis. Here I present the results of thermal convection as a member of the group, not as an individual. However, I have taken a lead role in scaling of Tarang, as well as on the energy spectrum computation.

In Fig. 6.2 we illustrate two isocontours of constant temperatures of our simulation data at an instant during the steady state. The hot plumes (red colour) ascend from the hot plate at the bottom, while the cold plumes (blue colour) descend from the cold plate at the top.

We computed the spectra and fluxes of the KE using the steady state data. Figure 6.3(a) exhibits the KE spectra normalized with $k^{11/5}$ and $k^{5/3}$. The plots indicate that in the wavenumber band $15 < k < 600$ (inertial range), the shaded region of the figure, the KO scaling fits better than the BO scaling.

The energy spectrum of RBC is modelled using Pao's model [62] for fluid turbulence that includes Kolmogorov's spectrum in the inertial range and a stretched expo-

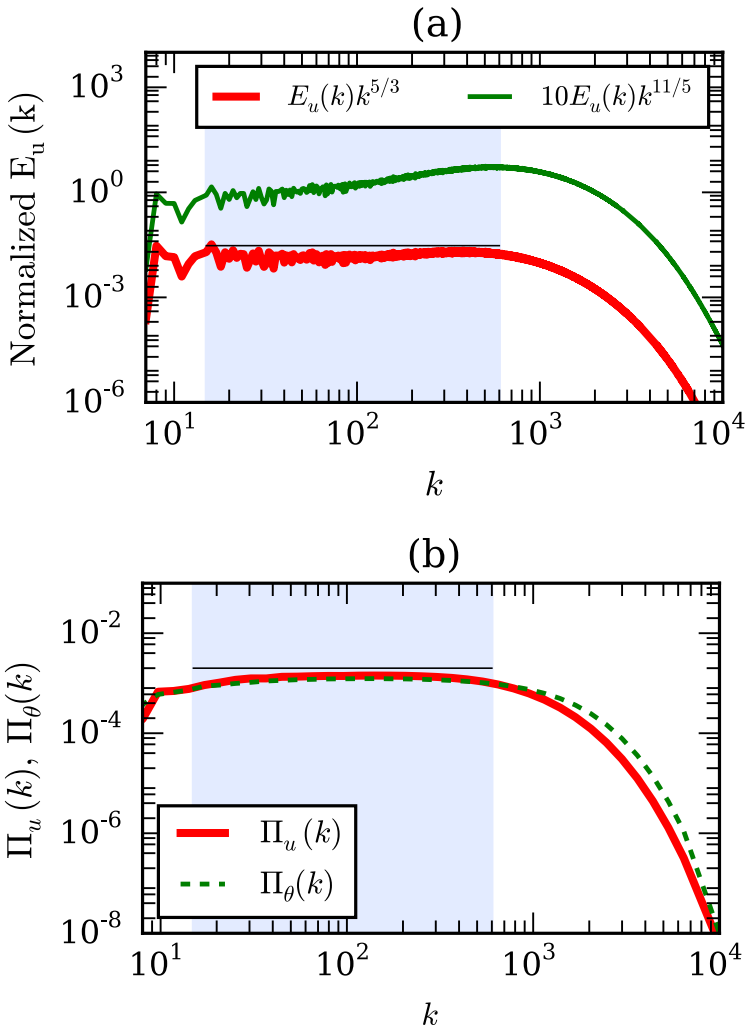


FIGURE 6.3: For the RBC simulation with Pr $= 1$ and Ra $= 1.1 \times 10^{11}$ on 4096^3 grid: (a) plots of normalized KE spectra for Bolgiano-Obukhov (BO) and Kolmogorov-Obukhov (KO) scaling; KO scaling fits better with the data than BO scaling. (b) KE flux $\Pi_u(k)$ and entropy flux $\Pi_\theta(k)$. The shaded region exhibits the inertial range. Taken from Verma *et al.* [98].

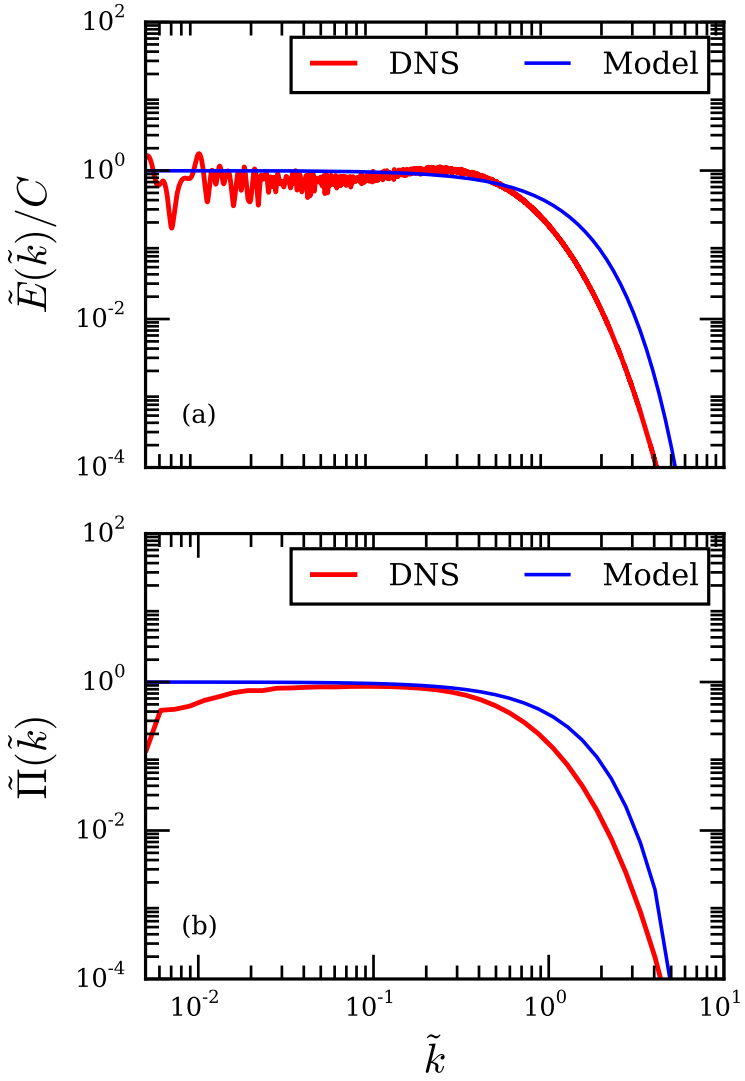


FIGURE 6.4: For RBC simulation on 4096^3 grid for Pr $= 1$ and Ra $= 1.1 \times 10^{11}$: (a) Plots of the normalised kinetic energy spectra $E(k)k^{5/3}/(K_{\mathrm{Ko}}\epsilon^{2/3})$ and $\exp(-\tilde{k}^{4/3})$ where $\tilde{k} = k/k_d$. (b) Plot of kinetic energy flux $\Pi(k)$ and $\exp(-\tilde{k}^{4/3})$. These curves demonstrate that Kolmogorov's theory of fluid turbulence describes the energy spectrum and flux of RBC quite well.

nential in the dissipation range (see Sec. 5.2.1):

$$E(k) = K_{\rm Ko}\epsilon^{2/3}k^{-5/3}\exp\left(-\tilde{k}^{4/3}\right), \tag{6.39}$$

$$\Pi(k) = \epsilon\exp\left(-\tilde{k}^{4/3}\right), \tag{6.40}$$

where $E(k)$ is the energy spectrum, $\Pi(k)$ is the energy flux, $K_{\rm Ko}$ is the Kolmogorov's constant, ϵ is the dissipation rate, and $\tilde{k} = k/k_d$ with k_d as the Kolmogorov's wavenumber:

$$k_d = \left[\frac{\epsilon}{\nu^3}\right]^{1/4}. \tag{6.41}$$

From Eqs. (6.39, 6.40) we deduce that

$$\frac{E(k)k^{5/3}}{K_{\rm Ko}\epsilon^{2/3}} = \frac{\Pi(k)}{\epsilon} = \exp\left(-\frac{3}{2}K_{\rm Ko}\tilde{k}^{4/3}\right). \tag{6.42}$$

Using our numerical data, we deduce that $K_{\rm Ko} \approx 1.4$, which is quite close to the Kolmogorov's constant for fluid turbulence.

We exhibit the KE and entropy fluxes in Fig 6.3(b). It is observed that the kinetic energy flux $\Pi_u(k)$ remains constant in the inertial range, a band where $E_u(k) \sim k^{-5/3}$. We, thus, claim that the convective turbulence exhibits Kolmogorov's power law in the inertial range.

We compare the numerical results of $E_u(k)$ and $\Pi_u(k)$ with those predicted by Pao's model. In Fig. 6.4(a,b) we show the compensated energy spectrum $E(k)k^{5/3}/(K_{\rm Ko}\epsilon^{2/3})$ and $\Pi(k)/\epsilon$, respectively, along with $\exp\left(-\tilde{k}^{4/3}\right)$. The numerical curve matches quite well with the model curves in the inertial range. Therefore we claim that convective turbulence follows Kolmogorov's model of fluid turbulence. The hump in $E(k)$ near the dissipation range is attributed to the bottleneck effect [76]. The above result is significant as it opens doors for accurate modelling of turbulent convection. For example, we can employ sub-grid models of fluid turbulence to model convective turbulence.

Verma *et al.* [98] computed $F_B(k)$, $\Pi_u(k)$, and $d\Pi_u(k)/dk$ as further tests. According to Fig. 6.5(a) $F_B(k) > 0$ in the inertial range, consistent with the discussion of Sec. 6.2.3 and Fig. 6.1(b), and it approximately balances $D(k)$. Therefore, $d\Pi_u(k)/dk \approx 0$ or $\Pi_u(k) \approx$ constant [see Eq. (6.14)]. The constancy of $\Pi_u(k)$ yields $E_u(k) \sim k^{-5/3}$, consistent with the energy spectrum plots of Fig. 6.3(a). Fig. 6.5(b) shows that $[d\Pi_u(k)/dk]/\Pi_u(k) \ll 1$ in the inertial range consistent with the constant $\Pi_u(k)$. Interestingly, $D(k) = 2\nu k^2 E_u(k) \sim k^{1/3}$, consistent with $E_u(k) \sim k^{-5/3}$. Also, $F_B(k) \sim k^{-5/3}$, thus

indicating strong forcing at low wavenumbers, similar to that in hydrodynamic turbulence. In addition, the entropy flux $\Pi_\theta(k)$ is constant, and $\Pi_u(k) \approx \Pi_\theta(k)$ in dimensionless units.

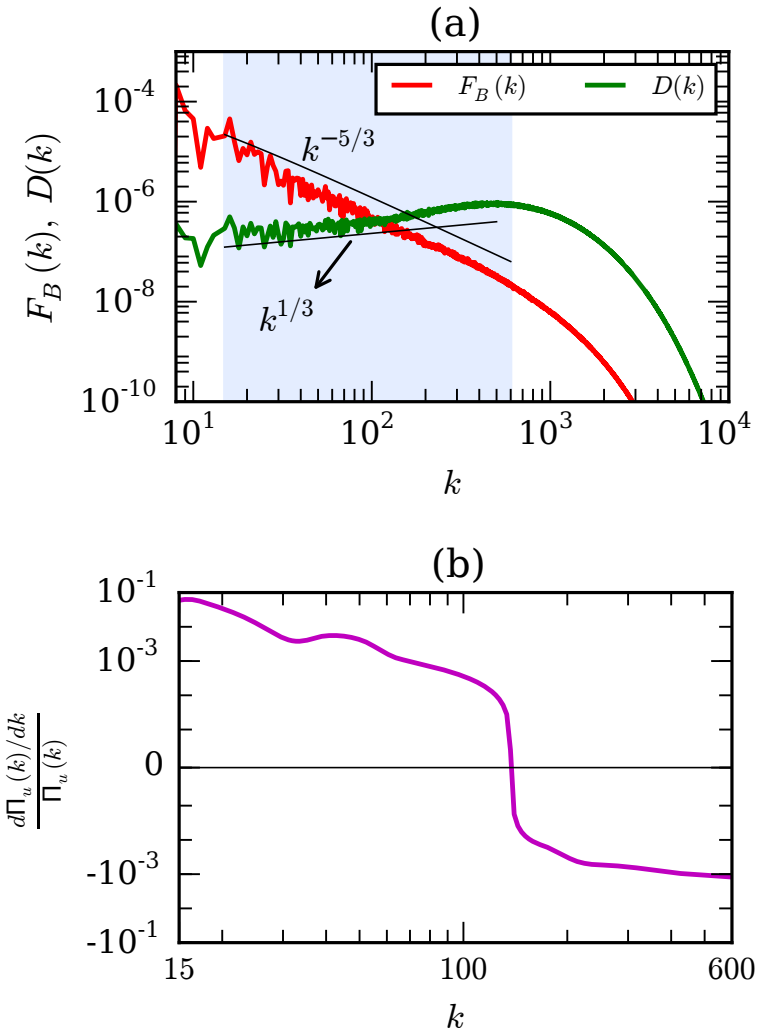


FIGURE 6.5: For the RBC simulation with Pr $= 1$ and Ra $= 1.1 \times 10^{11}$: (a) plots of $F_B(k)$ and $D(k)$. (b) plots of $[d\Pi_u(k)/dk]/\Pi_u(k)$ in the inertial range $15 < k < 600$. Taken from Verma *et al.* [98].

Using the steady-state data of Tarang, Verma *et al.* [98] computed the shell-to-shell energy transfers [Eq. (2.15)] and showed that they are local and forward. For this analysis they divided the Fourier space into 40 concentric shells. The radii of the inertial-range shells are binned logarithmically due to the power law physics of RBC in the inertial range. In Fig. 6.6 we exhibit the shell-to-shell energy transfers in which the indices of the x, y axes represent the receiver and giver shells respectively. The plot indicates that mth shell gives energy to $(m+1)$th shell, and it receives energy from the

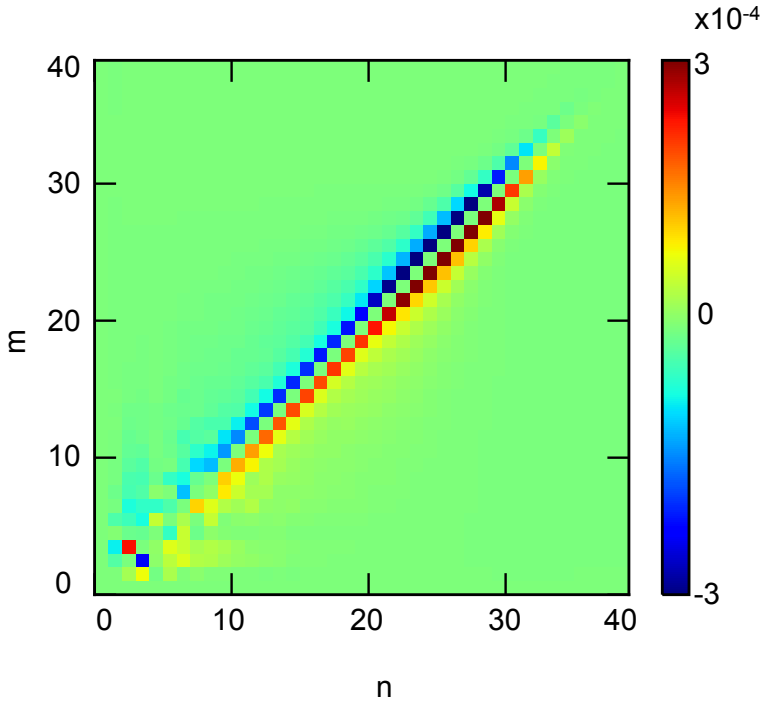


FIGURE 6.6: For RBC simulation on 4096^3 grid for Pr $= 1$ and Ra $= 1.1 \times 10^{11}$, the plot of the shell-to-shell energy transfers T_n^m of Eq. (2.15). The plot demonstrates local and forward energy transfers, similar to fluid turbulence. Taken from Verma *et al.* [98].

$(m-1)$th shell [88]. Thus the energy transfer in RBC is local and forward, similar to hydrodynamic turbulence. This result is consistent with the energy spectrum and flux studies described earlier.

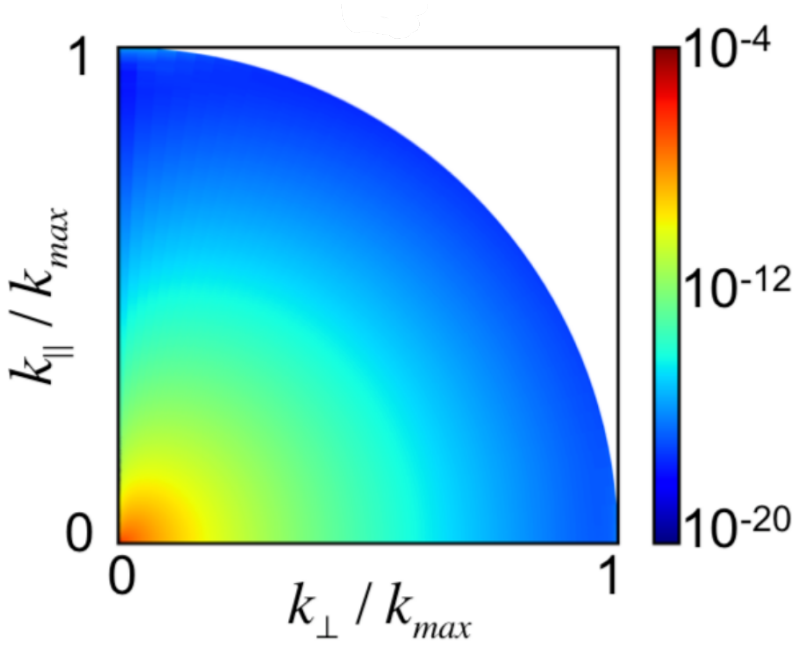


FIGURE 6.7: For RBC simulation on 4096^3 grid for Pr $= 1$ and Ra $= 1.1 \times 10^{11}$: Plot of the ring spectrum $E(k, \beta)$ demonstrates near isotropy in the Fourier space. Taken from Verma *et al.* [98].

Convective flows are expected to be anisotropic due to buoyancy; hence it is important to quantify anisotropy using the quantities that are dependent on the polar angle, the angle between z and $\mathbf{k}$. For the same, we divide a wavenumber shell into rings [59]. The energy contents of the rings are called *ring spectrum* $E(k, \beta)$, where β represents the sector index for the polar angles (for details see Nath *et al.* [59]). Also refer to Fig. 2.4 for illustration of rings. Using the numerical data, Nath *et al.* [59] studied the anisotropy in turbulent thermal convection. For the same, they computed the ring spectrum $E(k, \beta)$. As shown in Fig. 6.7(a), $E(k, \beta)$ is close to isotropic which is again similar to fluid turbulence. These results clearly confirm that the turbulent convection for Pr $= 1$ has a very similar behaviour as fluid turbulence.

Mishra and Verma [58] and Verma *et al.* [98] reported that the temperature fluctuation of turbulent RBC exhibits a unique behaviour. They observed dual branches for the entropy spectrum $(E_\theta(k))$. See Fig. 6.8 for an illustration. The upper branch varies as k^{-2} as $\theta(0, 0, k_z) \approx -1/(\pi k)$ The lower branch however shows neither KO $(k^{-5/3})$

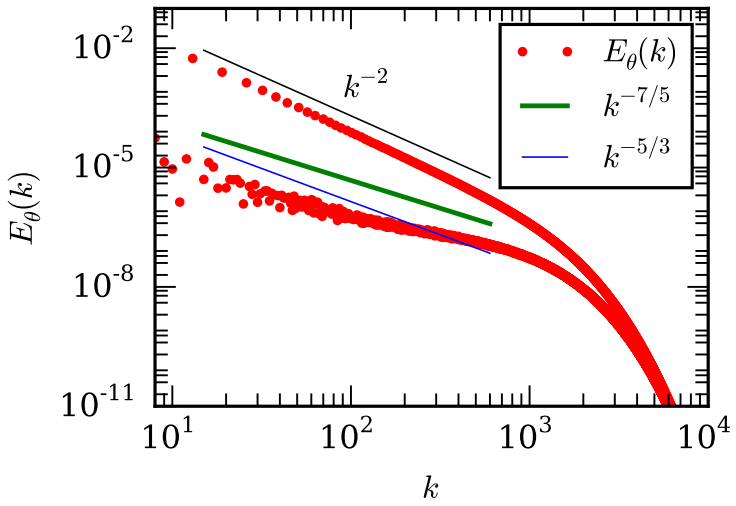


FIGURE 6.8: For RBC simulation with Pr $= 1$ and Ra $= 1.1 \times 10^{11}$, plot of the entropy spectrum that exhibits dual branches. The upper branch matches with k^{-2} quite well, while the lower part is fluctuating. Taken from Verma et al. [98].

nor BO ($k^{-7/5}$) spectrum. Note that both the branches of entropy spectrum generate a constant entropy flux $\Pi_\theta(k)$ (see Fig. 6.3(b)), and the modes $\theta(0, 0, k_z)$ also participate in energy transfers.

With this, we close our discussion on turbulent thermal convection. We summarise in the next section.

6.6 Summary of the results

In this chapter we report our numerical results of turbulent convection for the Rayleigh number Ra $= 1.1 \times 10^{11}$ and the Prandtl number Pr $= 1$ on 4096^3 grid with 65536 cores of a Cray XC40. Our numerical results provide definitive demonstration that convective turbulence is described by Kolmogorov's theory of turbulence. For analysis we also compute energy transfer diagnostics and show that the energy flux is constant in the inertial range, the shell-to-shell energy transfer is local and forward, and the ring-to-ring energy transfers are nearly isotropic. The ring spectrum also exhibits a near-isotropic behaviour. The extreme resolution of our simulation made these conclusions possible. We remark that the grid resolution of our simulation is highest in the field. Also, we achieved the largest Rayleigh number among spectral codes.

Our result that thermally-driven turbulence has behaviour similar to fluid turbulence is very important. Now we can model the convective turbulence in a better and more accurate way. For example, we can use fluid turbulence models of large-eddy simulation (LES) to convective turbulence also [85].

Our analysis open ways for future work on energy transfers in various forms of fluid flows such as fluid and magnetohydrodynamic turbulence, convective and stratified flows, rotating and channel flow, etc. These tools help us understand the physics of complex flows that are unsolved for a century. Also, the rapid development of hardware and software targeting exascale will enable us to simulate more complex flows and model them accurately.

Chapter 7

Conclusion

In this thesis we describe some of the salient features of spectral code Tarang. It is an object-oriented general purpose spectral solver that can simulate hydrodynamic and magnetohydrodynamic flows, thermal convection, stably-stratified flows. liquid metal flows, rotating flows, etc. As a developer of the code, I wrote the FFT library, FFTK, and the I/O library, H5SI. In addition, I have contributed to writing various parts of the code. We also remark that FFTK is as efficient as P3DFFT, but it has more features than P3DFFT. For example, we have implemented free-slip basis functions in FFTK and in Tarang. In 3D, the free-slip basis could be along one/two/three directions. We exploit this feature to simulate thermal convection with free-slip walls. We plan to employ these basis functions for rotating flows, and rotating convection.

In this concluding chapter, we summarise the scaling results, and the results of very-large resolution hydrodynamic turbulence and turbulent thermal convection.

7.1 Summary of hydrodynamic turbulence

We have performed scaling studies of FFTK library and Tarang on two different HPC clusters—Blue Gene/P (Shaheen I) and Cray XC40 (Shaheen II) of KAUST. The grid resolution was varied from 768^3 to 8192^3 on cores ranging from 1024 to 196608 [12]. The number of cores used for FFTK and Tarang are one of the largest in this area of research. The main results on scaling are as follows:

1. We analyse the computation and communication times for FFTK. We observe

that the computation time $T_{\text{comp}} \sim p^{-1}$ where p is the number of cores, while the communication time $T_{\text{comm}} \sim n^{-\gamma_2}$ where n is the number of nodes. For Blue Gene/P, γ_2 ranges from 0.7 to 0.9 depending the grid size. The corresponding variation for Cray XC40 is 0.40 to 0.70. The total time scales as $T \sim p^{-\gamma}$ with γ ranging from 0.76 to 0.96 for Blue Gene/P. For Cray XC40, γ lies between 0.43 to 0.73.

2. Cray XC40 exhibits lower efficiency ($\sim$ 1.5%) than Blue Gene/P ($\sim 4-10\%$). This is because the ratio of the per-node compute power of Cray XC40 and Blue Gene/P is approximately 1000, but the corresponding ratio for the interconnect is not that large. The relatively lower efficiency of Cray XC40 is due to the above reasons. Thus, the performance of a HPC system depends on the application. A faster switch is very critical for FFT computation.

3. The fluid solver of Tarang exhibits weak and strong scaling on both the cluster. The exponent γ for Blue Gene/P varies from 0.8 to 0.95, but it ranges from 0.28 to 0.68 for Cray XC40.

4. The solver for Rayleigh-Bénard convection also shows weak and strong scaling on both the clusters. The corresponding γ for Blue Gene/P ranges from 0.68 to 0.71, but it lies between 0.49 to 0.80 for Cray XC40.

5. The scaling of different basis functions (e.g., `FFF` and `SFF`) are similar. However the performance in the `SFF` basis is slightly better than that in the `FFF` basis.

Thus, FFTK and Tarang scale nearly up to 196608 cores. Thus these codes are capable of simulating turbulence at very high-resolution. FFTK would also be useful for other applications, e.g., image processing, density functional theory, etc.

7.2 Summary of DNS results on hydrodynamic turbulence

Using Tarang we performed spectral simulation of hydrodynamic turbulence on 4096^3 grid using 65536 cores of Cray XC40. The numerical data was used to demonstrate that Pao's model [62]—$E_u(k) \sim k^{-5/3} \exp(-k^{4/3})$—describes the energy spectrum in the inertial and dissipative ranges quite well. We also show that the energy flux computed using numerical simulation matches with the predictions of Pao. However, the numerical results, specially the energy flux, differ from the predictions of Pope [67]. These results are described in Verma *et al.* [97].

Using simulation data, we also compute the shell-to-shell energy transfers. We report that the shell-to-shell energy transfers are forward and local, in both, the inertial range and in the dissipation range.

7.3 Summary of DNS results on turbulent thermal convection

Using Tarang we simulated turbulent thermal convection on 4096^3 grid using 65536 cores of Cray XC40. We chose Pr = 1 and Ra $= 1.1 \times 10^{11}$. Using numerical data we compute the energy and entropy spectra, their respective fluxes, ring spectrum, shell-to-shell and ring-to-ring energy transfers[1]. Some of the key results obtained using numerical simulations are

1. The kinetic energy spectrum $E_u(k) \sim k^{-5/3}$, and the kinetic energy flux $\Pi_u(k) \sim$ constant. In addition, the shell-to-shell kinetic energy transfer is local and forward, and the ring spectrum is quite isotropic. Hence we conclude that physics of turbulent thermal convection is very similar to hydrodynamic turbulence [98].
2. The entropy flux $\Pi_\theta(k)$ is constant in the inertial range. But the entropy exhibits bi-spectrum [98].

7.4 Implications and future directions

A well-designed code like Tarang is very useful for DNS of turbulent flows. Maintaining a single code for many applications saves time and manpower. We could also optimise the code once, and it reflects on many other applications.

Though we have been successfully ran Tarang for many applications and shown its good scaling up to 196608 processors, many features need to be added to Tarang in future. They are

1. Inclusion of no-slip basis function using Chebyshev polynomials. Chebyshev polynomials are much more complex than sin/cos functions. But algorithm for this is well laid out [6] that could be implemented with care

[1] These analysis were performed by various members of the group [98]. My significant participation was towards the energy and entropy spectra and flux computations.

2. Enable Tarang for hybrid parallelisation with OpenMP and MPI. For multicore architecture with many cores (> 20), hybrid parallelisation could yield better efficiency. This is again a tedious task considering the size of the code.

3. Detailed profiling of the code. We need to identify the critical bottlenecks in the code. Then we need to speedup these segments of the code for a better efficiency. This is a laborious exercise. We plan to employ softwares like ITUNE and SCALASCA for this exercise.

4. User-friendly GUI-based reading of parameter and post-processing. Such features helps the users.

5. Inclusion of more solvers such as magnetoconvection, etc. These features will make the code richer, and it will attract more users. This exercise is somewhat easy due to Object-oriented design of Tarang.

Code development is an open-ended process. We hope to make Tarang even richer by addressing the above issues in near future.

Appendices

Appendix A

Transpose-free Fast Fourier Transform

In this appendix we describe how we avoid local transpose in FFTK to save communication time. The results presented here are taken from [12]. For simplicity we illustrate this procedure using slab decomposition with complex data of size $n_0 \times n_1 \times (n_2/2 + 1)$. The corresponding real space data is of the size $n_0 \times n_1 \times n_2$.

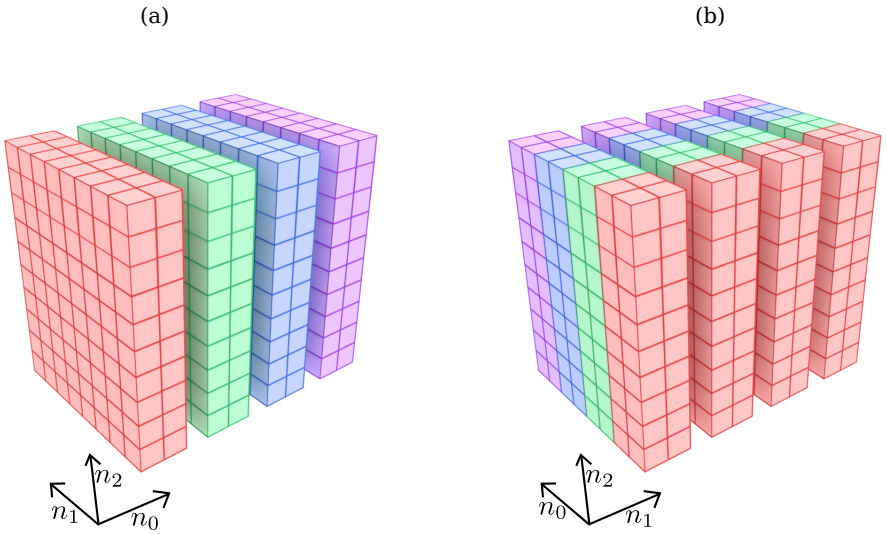


FIGURE A.1: Data division for FFT with transpose: (a) Complex data of size $n_0 \times n_1 \times (n_2/2 + 1)$ in Fourier space. (b) Real data of size $n_1 \times n_0 \times n_2$ in real space. Note that the axes n_0 and n_1 are exchanged during the transpose. Taken from Chatterjee *et al.*. [12].

The usual FFT implementation involving transpose is illustrated below. The complex data is divided along n_0. If there are p processors, then each processor has $(n_0/p) \times n_1 \times (n_2/2 + 1)$ complex data. See Fig. A.1(a) for an illustration. A typical

inverse transform involves three steps:

1. Perform two-dimensional inverse transforms (complex-to-real `c2r`) on n_0/p planes each having data of size $n_1 \times (n_2/2+1)$.

2. Perform transpose on the array along n_0-n_1 axis. This operation involves local transpose and `MPI_Alltoall` operations (to be described below).

3. After the data transfer, the data along the n_0 axis resides in the respective processors. Now in each processor, we perform one-dimensional real-to-real (`r2r`) inverse transforms on $(n_1/p) \times n_2$ column each having data of size n_0.

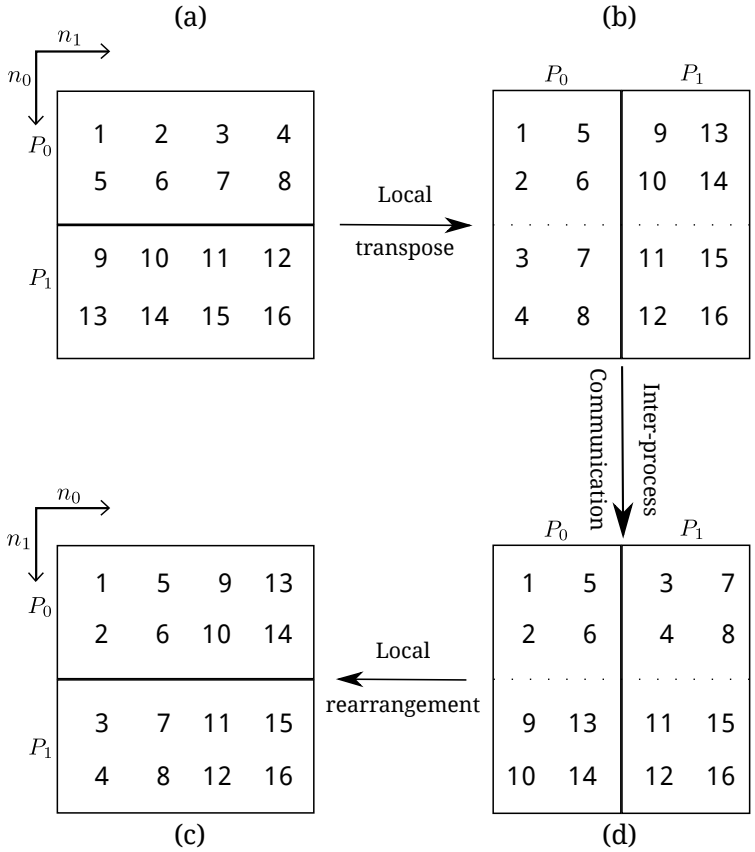


FIGURE A.2: The standard transpose procedure in during a FFT. It involves two local transposes and a `MPI_Alltoall`. Taken from Chatterjee *et al.* [12].

In Fig. A.2, we illustrate this transpose operation using a simple example involving 16 data points and 2 processors. In the first step, the local data is transposed, as illustrated in Fig. A.2(a,b). In the example, in process P_0, the data [[1,2,3,4],[5,6,7,9]] gets transformed to [[1,5],[2,6],[3,7],[4,8]]. After the local transpose, chunks of data are transferred among the processors using `MPI_Alltoall` function (Fig. A.2(b) to Fig. A.2(c)).

In this process, the blocks [[3,7],[4,8]] and [[9,13],[10,14]] are exchanged between P_0 and P_1. After this data transfer, there is another local transpose that transforms the data from Fig. A.2(c) to Fig. A.2(d).

This complete operation is called “transpose" because it is similar to matrix transpose. After transpose, we are ready for FFT operations along the n_0 axis. Note that the data along the rows of Fig. A.2(d) are consecutive, that makes it convenient for the FFT operation. We remark that the popular FFTW library employs the above procedure involving transpose.

An advantage of the above scheme is that the FFT is performed on consecutive data sets that minimises cache misses. However, the aforementioned FFT involves two local transposes, which are quite expensive. To avoid this, we have devised a FFT which is based on transpose-free data transfer. This process is described below.

In the transpose-free procedure, we replace the transpose operations (item 2 in the above list) with transpose-free inter-processor communication. We employ `MPI_Type_vector` and `MPI_Type_create_resized` to select strided data to be exchanged among the processors. We illustrate the communication process in Fig. A.3. Before communication, the processor P_0 contains data [[1,2,3,4],[5,6,7,9]], while the processor P_1 contains [[9, 10, 11, 12], [13, 14, 15, 16]]. During the communication, the data block [[3, 4], [7, 8]] is to be transferred from P_0 to P_1, and the data block [[9, 10], [13, 14]] is to be transferred from P_1 to P_0. Note that the data to be transferred are not consecutive, hence we need the MPI functions such as `MPI_Type_vector` and `MPI_Type_create_resized` to create strided-data sets. After the corresponding data has been transferred, P_0 contains [[1,2],[5,6],[9,10], [13,14]], while P_1 contains [[3,4], [7,8], [11,12], [15,16]], as desired. For these operations, special MPI functions `MPI_Type_vector` and `MPI_Type_create_resized` come quite handy.

The data structure before and after the interprocess communication are shown in Fig. A.4(a,b) respectively. Here the data axes are not exchanged, however the columnar data along the n_0 axis are not contiguous. For example, the data of column [1,5,9,13] of Fig. A.3(b) are staggered by 1. As a result, FFT along the n_1 axis involves consecutive data, but not along n_0. The latter FFT however can be performed using strided FFTW functions.

Now let us briefly compare the performances of the two methods. The transpose-free scheme avoids local transpose, hence it saves some communication time compared to the usual FFT. A flip side of the transpose-free scheme is that it needs strided FFT that is prone to cache misses because the data are not contagious. Note, however,

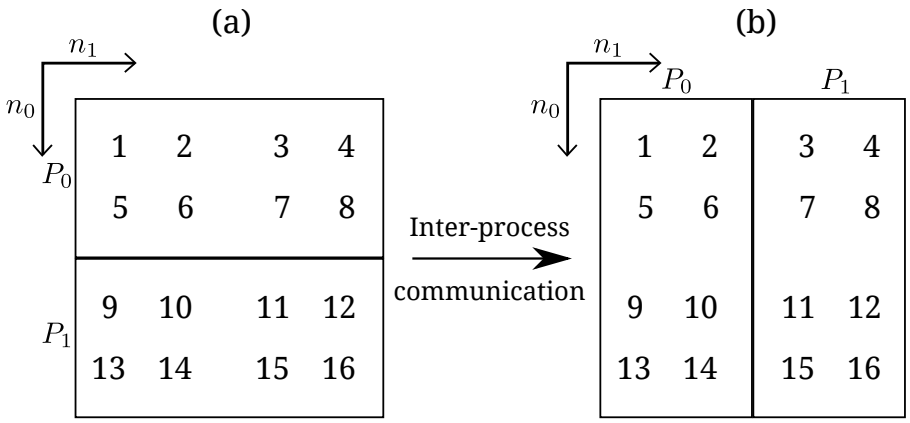


FIGURE A.3: Transpose using strided `MPI_Isend/MPI_Recv` that does not require a local transpose. This is employed in the transpose-free FFT. Taken from Chatterjee *et al.* [12].

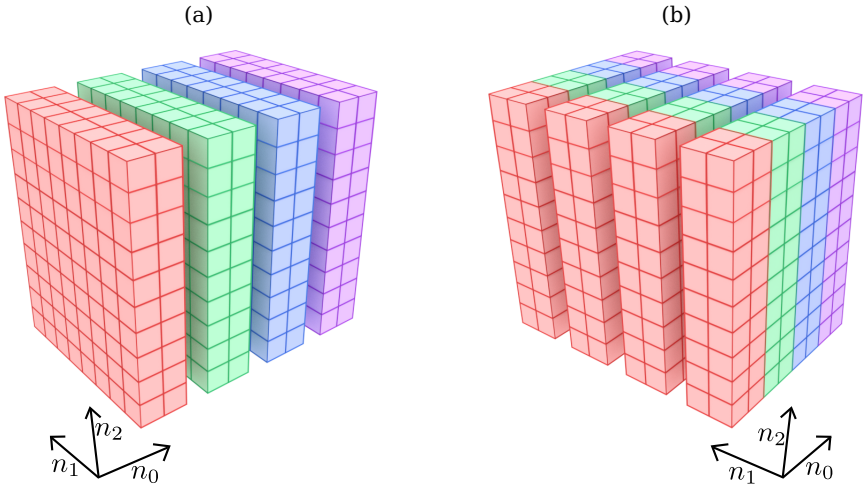


FIGURE A.4: Data division for a transpose-free FFT: (a) Complex data of size $n_0 \times n_1 \times (n_2/2+1)$ in Fourier space. (b) Real data of size $n_0 \times n_1 \times n_2$ in real space. Note that there is no exchange of axes here. Compare it with Fig. A.1. Taken from Chatterjee *et al.* [12].

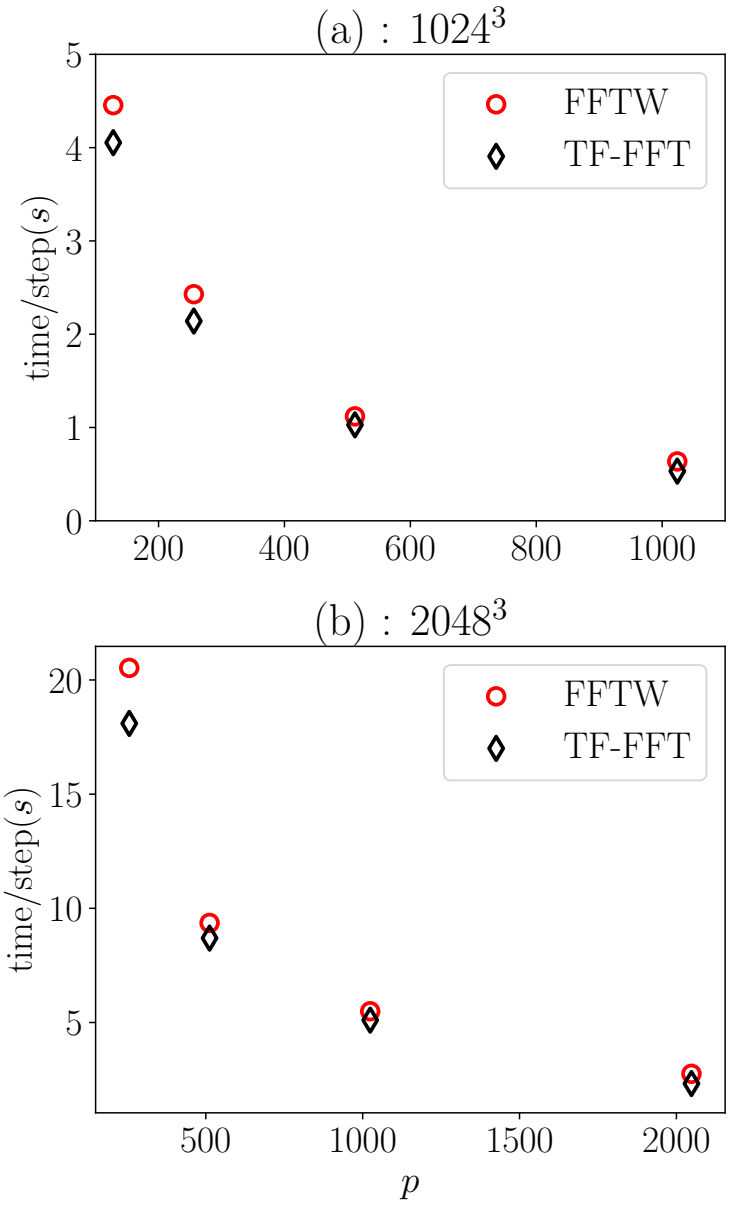


FIGURE A.5: Comparison between FFTs with transpose and without transpose for (a) 1024^3 grid, and (b) 2048^3 grid. Transpose-free FFT is marginally superior than the one with transpose. Taken from Chatterjee *et al.* [12].

that intelligent cache prefetch algorithms [2] could helps in efficient implementation of strided FFT.

To compare the efficiencies of the aforementioned FFT schemes, we performed FFTs using both the schemes. Since FFTW involves local transposes, we use this as one of the benchmark programs. We wrote a transpose-free FFT function as the other benchmark program. The tests were performed on IBM BlueGene/P (Shaheen I) of KAUST for a pair of forward and inverse transforms on 1024^3 and 2048^3 grids.

In Fig. A.5(a,b) we present the results for the 1024^3 and 2048^3 grids. In the figure the time taken by FFTW and transpose-free FFT are shown by red circle and black diamonds respectively. We observe that the transpose-free FFT is 10% to 16% more efficient for 1024^3 data, and 5% to 14% more efficient for 2048^3 data. The gain by the transpose-free FFT decreases as the number of processors are increased. The difference in time is a sum of two factors: (a) gain by avoidance of local transpose, and (b) loss due to strided FFT. We need a more detailed diagnostics to analyse the two algorithms. For example, we need to separately compute the computation and communication time. Also, it will be useful to estimate the time for the collection of the strided data, as well as that of the strided FFT. These works will be performed in future.

The FFT operations in FFTK, which has pencil decomposition, in transpose-free. The only difference between the slab-based FFT described in this appendix and the pencil-based FFT is that the data exchange in pencil-based FFT takes place among the respective communicators. For example, among the `MPI_COM_ROW` and `MPI_COM_COL` communicators of Fig. 3.2.

Appendix B

Datatypes for the H5SI library

The following datatypes are available for the H5SI library:

">f32"	Big endian 32 bit floating point number
"<f32"	Little endian 32 bit floating point number
">f64"	Big endian 64 bit floating point number
"<f64"	Little endian 64 bit floating point number
">i8"	Big endian 8 bit integer number
"<i8"	Little endian 8 bit integer number
">i16"	Big endian 16 bit integer number
"<i16"	Little endian 16 bit integer number
">i32"	Big endian 32 bit integer number
"<i32"	Little endian 32 bit integer number
">i64"	Big endian 64 bit integer number
"<i64"	Little endian 64 bit integer number
">u8"	Big endian 8 bit unsigned integer number
"<u8"	Little endian 8 bit unsigned integer number
">u16"	Big endian 16 bit unsigned integer number
"<u16"	Little endian 16 bit unsigned integer number

">u32"	Big endian 32 bit unsigned integer number
"<u32"	Little endian 32 bit unsigned integer number
">u64"	Big endian 64 bit unsigned integer number
"<u64"	Little endian 64 bit unsigned integer number
">b8"	Big endian 8 bit boolean
"<b8"	Little endian 8 bit boolean
">b16"	Big endian 16 bit boolean
"<b16"	Little endian 16 bit boolean
">b32"	Big endian 32 bit boolean
"<b32"	Little endian 32 bit boolean
">b64"	Big endian 64 bit boolean
"<b64"	Little endian 64 bit boolean
"S"	String
"a"	String
"char"	Native character
"schar"	Native short character
"uchar"	Native unsigned character
"short"	Native short integer
"ushort"	Native unsigned short integer
"int"	Native integer
"uint"	Native unsigned integer
"long"	Native long integer
"ulong"	Native unsigned long integer
"llong"	Native long long integer
"ullong"	Native unsigned long long integer
"float"	Native float
"double"	Native double
"ldouble"	Native long double

"b8"	Native 8 bit boolean
"b16"	Native 16 bit boolean
"b32"	Native 32 bit boolean
"b64"	Native 64 bit boolean
"cfloat"	Pair of native float
"cdouble"	forward Pair of native double

Bibliography

[1] BARTELLO, P., AND TOBIAS, S. M. Sensitivity of stratified turbulence to the buoyancy Reynolds number. *J. Fluid Mech. 725* (June 2013), 1–22.

[2] BERG, S. G. Cache prefetching. *Technical Report, UW-CSE* (Feb 2004).

[3] BIFERALE, L., BONACCORSO, F., MAZZITELLI, I. M., VAN HINSBERG, M. A. T., LANOTTE, A. S., MUSACCHIO, S., PERLEKAR, P., AND TOSCHI, F. Coherent Structures and Extreme Events in Rotating Multiphase Turbulent Flows. *Phys. Rev. X 6* (Nov. 2016), 041036.

[4] BOLGIANO, R. Turbulent spectra in a stably stratified atmosphere. *J. Geophys. Res. 64*, 12 (1959), 2226–2229.

[5] BORUE, V., AND ORSZAG, S. A. Turbulent convection driven by a constant temperature gradient. *J. Sci. Comput. 12*, 3 (1997), 305–351.

[6] BOYD, J. P. *Chebyshev and Fourier Spectral Methods*, 2nd revised ed. Dover Publications, New York, 2003.

[7] BRETHOUWER, G., BILLANT, P., BILLANT, P., LINDBORG, E., AND CHOMAZ, J.-M. Scaling analysis and simulation of strongly stratified turbulent flows. *J. Fluid Mech. 585* (Aug. 2007), 343–368.

[8] CAMUSSI, R., AND VERZICCO, R. Temporal statistics in high rayleigh number convective turbulence. *Eur. J. Mech. B. Fluids 23* (Jan. 2004), 427–442.

[9] CANUTO, C., HUSSAINI, M. Y., QUARTERONI, A., AND ZANG, T. A. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, Berlin Heidelberg, 1988.

[10] CANUTO, C., HUSSAINI, M. Y., QUARTERONI, A., AND ZANG, T. A. *Spectral methods: Fundamentals in Single Domains*. Scientific computation. Springer-Verlag, Berlin, 2006.

[11] CHAN, A., BALAJI, P., GROPP, W., AND THAKUR, R. Communication analysis of parallel 3d fft for flat cartesian meshes on large blue gene systems. In *High Performance Computing - HiPC 2008* (Berlin, Heidelberg, 2008), Springer Berlin Heidelberg, pp. 350–364.

[12] CHATTERJEE, A. G., VERMA, M. K., KUMAR, A., SAMTANEY, R., HADRI, B., AND KHURRAM, R. Scaling of a Fast Fourier Transform and a pseudo-spectral fluid solver up to 196608 cores. *J. Parallel Distrib. Comput. 113* (Mar. 2018), 77–91.

[13] CHING, E. S. C. Scaling laws in the central region of confined turbulent thermal convection. *Phys. Rev. E 75*, 5 (May 2007), 056302.

[14] CHING, E. S. C. *Statistics and Scaling in Turbulent Rayleigh-Bénard Convection*. Springer, Berlin, 2013.

[15] CHING, E. S. C., AND CHENG, W. C. Anomalous scaling and refined similarity of an active scalar in a shell model of homogeneous turbulent convection. *Phys. Rev. E 77*, 1 (Jan. 2008), 015303.

[16] CLAY, M., BUARIA, D., YEUNG, P., AND GOTOH, T. Gpu acceleration of a petascale application for turbulent mixing at high schmidt number using openmp 4.5. *Computer Physics Communications* (2018).

[17] COOLEY, J. W., AND TUKEY, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation 19*, 90 (1965), 297–301.

[18] CZECHOWSKI, K., BATTAGLINO, C., MCCLANAHAN, C., IYER, K., YEUNG, P.-K., AND VUDUC, R. On the communication complexity of 3d ffts and its implications for exascale. In *Proceedings of the 26th ACM International Conference on Supercomputing* (New York, NY, USA, 2012), ICS '12, ACM, pp. 205–214.

[19] CZECHOWSKI, K., BATTAGLINO, C., MCCLANAHAN, C., IYER, K., YEUNG, P. K., AND VUDUC, R. On the communication complexity of 3D FFTs and its implications for Exascale. In *Proceedings of the 26th ACM international conference on Supercomputing* (New York, New York, USA, June 2012), ACM, pp. 205–214.

[20] DALLAS, V., FAUVE, S., AND ALEXAKIS, A. Statistical Equilibria of Large Scales in Dissipative Hydrodynamic Turbulence. *Phys. Rev. Lett. 115*, 20 (Nov. 2015), 204501.

[21] DAR, G., VERMA, M. K., AND ESWARAN, V. Energy transfer in two-dimensional magnetohydrodynamic turbulence: formalism and numerical results. *Physica D 157*, 3 (Jan. 2001), 207–225.

[22] DAVIDSON, P. A. *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, Oxford, 2004.

[23] DEBLIQUY, O., VERMA, M. K., AND CARATI, D. Energy fluxes and shell-to-shell transfers in three-dimensional decaying magnetohydrodynamic turbulence. *Phys. Plasmas 12*, 4 (Apr. 2005), 042309.

[24] DOBLER, W., HAUGEN, N. E. L., YOUSEF, T. A., AND BRANDENBURG, A. Bottleneck effect in three-dimensional turbulence simulations. *Phys. Rev. E 68*, 2 (Aug. 2003), 026304.

[25] DOMARADZKI, J. A., AND ROGALLO, R. S. Local Energy Transfer and Nonlocal Interactions in Homogeneous, Isotropic Turbulence. *Phys. Fluids A 2*, 3 (Jan. 1990), 414–426.

[26] DONZIS, D. A., AND SREENIVASAN, K. R. Short-term forecasts and scaling of intense events in turbulence. *J. Fluid Mech. 647* (2010), 13–26.

[27] DONZIS, D. A., AND SREENIVASAN, K. R. The bottleneck effect and the Kolmogorov constant in isotropic turbulence. *J. Fluid Mech. 657* (June 2010), 171–188.

[28] DONZIS, D. A., SREENIVASAN, K. R., AND YEUNG, P. K. The Batchelor Spectrum for Mixing of Passive Scalars in Isotropic Turbulence. *Flow Turbul. Combust. 85*, 3-4 (July 2010), 549–566.

[29] DONZIS, D. A., YEUNG, P. K., AND PEKUROVSKY, D. Turbulence simulations on $O(10^4)$ processors. In *Proc TeraGrid* (2008).

[30] DONZIS, D. A., YEUNG, P. K., AND SREENIVASAN, K. R. Dissipation and enstrophy in isotropic turbulence: Resolution effects and scaling in direct numerical simulations. *Phys. Fluids 20*, 4 (Apr. 2008), 045108.

[31] FALKOVICH, G. Bottleneck phenomenon in developed turbulence. *Phys. Fluids 6*, 4 (1994), 1411–1414.

[32] FRIGO, M., AND JOHNSON, S. G. The Design and Implementation of FFTW3. *Proceedings of the IEEE 93*, 2 (Feb. 2005), 216–231.

[33] FRISCH, U. *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge University Press, Cambridge, 1995.

[34] GOTOH, T., FUKAYAMA, D., AND NAKANO, T. Velocity field statistics in homogeneous steady turbulence obtained using a high-resolution direct numerical simulation. *Phys. Fluids 14*, 3 (2002), 1065–1081.

[35] GRANT, H. L., STEWART, R. W., AND MOILLIET, A. Turbulence spectra from a tidal channel. *J. Fluid Mech. 12*, 02 (1962), 241–268.

[36] GROSSMANN, S., AND LOHSE, D. Fourier-Weierstrass mode analysis for thermally driven turbulence. *Phys. Rev. Lett. 67*, 4 (1991), 445–448.

[37] HADRI, B., KORTAS, S., FEKI, S., KHURRAM, R., AND NEWBY, G. Overview of the KAUST's Cray X40 System–Shaheen II. In *CUG2015 Proceedings* (2015).

[38] ISHIHARA, T., YOKOKAWA, M., ITAKURA, K., AND UNO, A. Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box. *Phys. Fluids 15*, 2 (Feb. 2003), L21.

[39] ISHIHARA, T., YOKOKAWA, M., ITAKURA, K., AND UNO, A. Energy Spectrum in the Near Dissipation Range of High Resolution Direct Numerical Simulation of Turbulence. *J. Phys. Soc. Jpn. 74*, 5 (May 2005), 1464–1471.

[40] KACZOROWSKI, M., AND XIA, K.-Q. Turbulent flow in the bulk of Rayleigh–Bénard convection: small-scale properties in a cubic cell. *J. Fluid Mech. 722* (2013), 596–617.

[41] KERR, R. M. Rayleigh number scaling in numerical convection. *J. Fluid Mech. 310* (Jan. 1996), 139–179.

[42] KOLMOGOROV, A. N. Dissipation of Energy in Locally Isotropic Turbulence. *Dokl Acad Nauk SSSR 32*, 1 (1941), 16–18.

[43] KOLMOGOROV, A. N. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Dokl Acad Nauk SSSR 30*, 4 (1941), 301–305.

[44] KRAICHNAN, R. H. Inertial-range transfer in two-and three-dimensional turbulence. *J. Fluid Mech. 47* (1971), 525–535.

[45] KUMAR, A., CHATTERJEE, A. G., AND VERMA, M. K. Energy spectrum of buoyancy-driven turbulence. *Phys. Rev. E 90*, 2 (Aug. 2014), 023016.

[46] KUMAR, A. AND VERMA, M. K. Applicability of Taylor's hypothesis in thermally driven turbulence. R. Soc. open sci. 5, (2018) 172152.

[47] LESIEUR, M. *Turbulence in Fluids*. Springer-Verlag, Dordrecht, 2008.

[48] LESLIE, D. C. *Developments in the theory of turbulence*. Clarendon Press, Oxford, 1973.

[49] LINDBORG, E. The energy cascade in a strongly stratified fluid. *J. Fluid Mech. 550* (Mar. 2006), 207–242.

[50] LOHSE, D., AND MÜLLER-GROELING, A. Bottleneck effects in turbulence: Scaling phenomena in r versus p space. *Phys. Rev. Lett. 74*, 10 (Mar. 1995), 1747–1750.

[51] LOHSE, D., AND XIA, K.-Q. Small-scale properties of turbulent Rayleigh–Bénard convection. *Annu. Rev. Fluid Mech. 42*, 1 (2010), 335–364.

[52] L'VOV, V. S. Spectra of velocity and temperature-fluctuations with constant entropy flux of fully-developed free-convective turbulence. *Phys. Rev. Lett. 67*, 6 (Jan. 1991), 687–690.

[53] L'VOV, V. S., AND FALKOVICH, G. Conservation laws and two-flux spectra of hydrodynamic convective turbulence. *Physica D 57* (Jan. 1992), 85–95.

[54] MARTÍNEZ, D. O., CHEN, S., DOOLEN, G. D., KRAICHNAN, R. H., WANG, L.-P., AND ZHOU, Y. Energy spectrum in the dissipation range of fluid turbulence. *J. Plasma Phys. 57*, 1 (Jan. 1997), 195–201.

[55] MCCOMB, W. D. *The physics of fluid turbulence*. Clarendon Press, Oxford, 1990.

[56] MININNI, P. D., ALEXAKIS, A., AND POUQUET, A. Nonlocal interactions in hydrodynamic turbulence at high Reynolds numbers: The slow emergence of scaling laws. *Phys. Rev. E 77*, 3 (Mar. 2008), 036306.

[57] MININNI, P. D., ROSENBERG, D. L., REDDY, R., AND POUQUET, A. G. A hybrid MPI-OpenMP scheme for scalable parallel pseudospectral computations for fluid turbulence. *Parallel Computing 37*, 6-7 (July 2011), 316–326.

[58] MISHRA, P. K., AND VERMA, M. K. Energy spectra and fluxes for Rayleigh-Bénard convection. *Phys. Rev. E 81*, 5 (May 2010), 056316.

[59] NATH, D., PANDEY, A., KUMAR, A., AND VERMA, M. K. Near isotropic behavior of turbulent thermal convection. *Phys. Rev. Fluids 1* (Oct. 2016), 064302.

[60] OBUKHOV, A. M. On influence of buoyancy forces on the structure of temperature field in a turbulent flow. *Dokl Acad Nauk SSSR 125* (1959), 1246.

[61] ORSZAG, S. A. Comparison of pseudospectral and spectral approximation. *Studies in Applied Mathematics 51*, 3 (1972), 253–259.

[62] PAO, Y.-H. Structure of Turbulent Velocity and Scalar Fields at Large Wavenumbers. *Phys. Fluids 8*, 6 (1965), 1063–1075.

[63] PEKUROVSKY, D. P3DFFT: A Framework for Parallel Computations of Fourier Transforms in Three Dimensions. *Siam J. Sci. Comput. 34*, 4 (Jan. 2012), C192–C209.

[64] PEKUROVSKY, D. P3dfft: A framework for parallel computations of fourier transforms in three dimensions. *SIAM Journal on Scientific Computing 34*, 4 (Aug. 2012), C192–C209.

[65] PIPPIG, M., AND POTTS, D. Scaling parallel fast fourier transform on bluegene/p. In *Jülich BlueGeneP Scaling Workshop* (2010), Jülich BlueGene/P Scaling Workshop.

[66] PIPPIG, M., AND POTTS, D. Scaling parallel fast fourier transform on bluegene/p. In *Jülich BlueGene/P Scaling Workshop* (2010), pp. 27–30.

[67] POPE, S. B. *Turbulent Flows*. Cambridge University Press, Cambridge, 2000.

[68] PROCACCIA, I., AND ZEITAK, R. Scaling exponents in nonisotropic convective turbulence. *Phys. Rev. Lett. 62*, 18 (1989), 2128–2131.

[69] REDDY, K. S., AND VERMA, M. K. Strong anisotropy in quasi-static magnetohydrodynamic turbulence for high interaction parameters. *Phys. Fluids 26* (2014), 025109.

[70] RICHARDS, D. F., GLOSLI, J. N., CHAN, B., DORR, M. R., DRAEGER, E. W., FATTEBERT, J.-L., KRAUSS, W. D., SPELCE, T., STREITZ, F. H., SURH, M. P., AND GUNNELS, J. A. Beyond homogeneous decomposition: Scaling long-range forces on massively parallel systems. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (New York, NY, USA, 2009), SC '09, ACM, pp. 60:1–60:12.

[71] RILEY, J. J., AND LINDBORG, E. Recent progress in stratified turbulence. In *Ten Chapters in Turbulence*. Cambridge University Press, 2010.

[72] RINCON, F. Anisotropy, inhomogeneity and inertial-range scalings in turbulent convection. *J. Fluid Mech. 563* (Jan. 2006), 43–69.

[73] RORAI, C., MININNI, P. D., AND POUQUET, A. G. Stably stratified turbulence in the presence of large-scale forcing. *Phys. Rev. E 92*, 1 (2015), 013003.

[74] ROSENBERG, D. L., POUQUET, A. G., MARINO, R., AND MININNI, P. D. Evidence for Bolgiano-Obukhov scaling in rotating stratified turbulence using high-resolution direct numerical simulations. *Phys. Fluids 27*, 5 (May 2015), 055105.

[75] RUBINSTEIN, R. Renormalization group theory of Bolgiano scaling in Boussinesq turbulence. Tech. Rep. ICOM-94-8; CMOTT-94-2, 1994.

[76] SADDOUGHI, S. G., AND VEERAVALLI, S. V. Local isotropy in turbulent boundary layers at high Reynolds number. *J. Fluid Mech. 268* (1994), 333–372.

[77] SCHEEL, J. D., AND SCHUMACHER, J. Local boundary layer scales in turbulent Rayleigh–Bénard convection. *J. Fluid Mech. 758* (Oct. 2014), 344–373.

[78] SCHEEL, J. D., AND SCHUMACHER, J. Predicting transition ranges to fully turbulent viscous boundary layers in low prandtl number convection flows. *Phys. Rev. Fluids 2* (Dec 2017), 123501.

[79] SCHUMACHER, J., BANDARU, V., PANDEY, A., AND SCHEEL, J. D. Transitional boundary layers in low-Prandtl-number convection. *Phys. Rev. Fluids 1*, 8 (Dec. 2016), 084402.

[80] ŠKANDERA, D., BUSSE, A., AND MÜLLER, W.-C. Scaling properties of convective turbulence. In *High Performance Computing in Science and Engineering, Garching/Munich 2007* (Berlin, Heidelberg, 2009), S. Wagner, M. Steinmetz, A. Bode, and M. Brehm, Eds., Springer Berlin Heidelberg, pp. 387–396.

[81] SREENIVASAN, K. R. On the universality of the Kolmogorov constant. *Phys. Fluids 7*, 11 (Nov. 1995), 2778–2784.

[82] STEVENS, R. J. A. M., LOHSE, D., AND VERZICCO, R. Prandtl and Rayleigh number dependence of heat transport in high Rayleigh number thermal convection. *J. Fluid Mech. 688* (2011), 31–43.

[83] TEACA, B., VERMA, M. K., KNAEPEN, B., AND CARATI, D. Energy transfer in anisotropic magnetohydrodynamic turbulence. *Phys. Rev. E 79*, 4 (Apr. 2009), 046312.

[84] VAN DER POEL, E. P., VERZICCO, R., GROSSMANN, S., AND LOHSE, D. Plume emission statistics in turbulent rayleighâĂŞbÃľnard convection. *Journal of Fluid Mechanics 772* (Jun 2015), 5âĂŞ15.

[85] VASHISHTHA, S., CHATTERJEE, A., KUMAR, A., AND VERMA, M. K. Large eddy simulations using recursive renormalization-group based eddy viscosity. *ArXiv:1712.03170* (Dec. 2017).

[86] VELDHUIZEN, T. Blitz++ user guide: A c++ class library for scientific computing. Tech. rep., Tech. rep., available at: http://blitz.sourceforge.net/resources/blitz-0.9.pdf (last access: Nov. 2017), 2006.

[87] VAN DER POEL, E. P., OSTILLA-MÓNICO, R., DONNERS, J., AND VERZICCO, R.. A pencil distributed finite difference code for strongly turbulent wall-bounded flows. *Computers & Fluids 116*, (2015), 10–16.

[88] VERMA, M. K. Statistical theory of magnetohydrodynamic turbulence: recent results. *Phys. Rep. 401*, 5 (Nov. 2004), 229–380.

[89] VERMA, M. K. Variable enstrophy flux and energy spectrum in two-dimensional turbulence with Ekman friction. *EPL 98* (2012), 14003.

[90] VERMA, M. K. Anisotropy in Quasi-Static Magnetohydrodynamic Turbulence. *Rep. Prog. Phys. 80*, 8 (May 2017), 087001.

[91] VERMA, M. K. *Physics of Buoyant Flows: From Instabilities to Turbulence*, World Scientific, Singapore, 2018.

[92] VERMA, M. K., AYYER, A., DEBLIQUY, O., KUMAR, S., AND CHANDRA, A. V. Local shell-to-shell energy transfer via nonlocal interactions in fluid turbulence. *Pramana-J. Phys. 65*, 2 (Jan. 2005), 297–310.

[93] VERMA, M. K., CHATTERJEE, A. G., YADAV, R. K., PAUL, S., CHANDRA, M., AND SAMTANEY, R. Benchmarking and scaling studies of pseudospectral code Tarang for turbulence simulations. *Pramana-J. Phys. 81*, 4 (Sept. 2013), 617–629.

[94] VERMA, M. K., AND DONZIS, D. A. Energy transfer and bottleneck effect in turbulence. *J. Phys. A: Math. Theor. 40*, 16 (Mar. 2007), 4401–4412.

[95] VERMA, M. K., KUMAR, A., AND CHATTERJEE, A. G. Energy Spectrum and Flux of Buoyancy-Driven Turbulence . *Phys. Focus 25* (Feb. 2015), 45.

[96] VERMA, M. K., KUMAR, A., AND CHATTERJEE, A. G. Energy Spectrum and Flux of Buoyancy-Driven Turbulence. In *Proceedings of the Advances in Computation, Modeling and Control of Transitional and Turbulent Flows* (2015), T. K. Sengupta, S. K. Lele, K. R. Sreenivasan, and P. A. Davidson, Eds., pp. 442–451.

[97] VERMA, M. K., KUMAR, A., KUMAR, P., BARMAN, S., CHATTERJEE, A. G., AND SAMTANEY, R. Energy fluxes and spectra for turbulent and laminar flows. *arXiv:1705.04917* (May 2017).

[98] VERMA, M. K., KUMAR, A., AND PANDEY, A. Phenomenology of buoyancy-driven turbulence: recent results. *New J. Phys. 19* (2017), 025012.

[99] VERMA, M. K., AND REDDY, K. S. Modeling quasi-static magnetohydrodynamic turbulence with variable energy flux. *Phys. Fluids 27*, 2 (Feb. 2015), 025114.

[100] VERZICCO, R., AND CAMUSSI, R. Numerical experiments on strongly turbulent thermal convection in a slender cylindrical cell. *J. Fluid Mech. 477* (Jan. 2003), 19–49.

[101] YAKHOT, V., AND ORSZAG, S. A. Renormalization group analysis of turbulence. I. Basic theory. *J. Sci. Comput. 1*, (Mar. 1986), 3–51.

[102] YEUNG, P. K., AND BRASSEUR, J. G. The response of isotropic turbulence to isotropic and anisotropic forcing at the large scales. *Phys. Fluids A 3*, (1991), 884.

[103] YEUNG, P. K., DONZIS, D. A., AND SREENIVASAN, K. R. High-Reynolds-number simulation of turbulent mixing. *Phys. Fluids 17*, (Aug. 2005), 081703.

[104] YEUNG, P. K., AND SREENIVASAN, K. R. Spectrum of passive scalars of high molecular diffusivity in turbulent mixing. *Journal of Fluid Mechanics 716* (2013), R14.

[105] YEUNG, P. K., ZHAI, X. M., AND SREENIVASAN, K. R. Extreme events in computational turbulence. *PNAS 112*, 41 (Oct. 2015), 12633.

[106] YEUNG, P. K., AND ZHOU, Y. Universality of the Kolmogorov constant in numerical simulations of turbulence. *Phys. Rev. E 56*, 2 (Aug. 1997), 1746–1752.

[107] YOKOKAWA, M., ITAKURA, K., UNO, A., AND ISHIHARA, T. 16.4-Tflops Direct Numerical Simulation of Turbulence by a Fourier Spectral Method on the Earth Simulator. In *ACM/IEEE 2002 Conference* (2002), IEEE.

[108] ZHOU, Y. Degrees of locality of energy transfer in the inertial range. *Phys. Fluids 5* (1993), 1092–1094.