PRAMANA — journal of physics

Vol. 81, No. 4
October 2013
pp. 617–629

Benchmarking and scaling studies of pseudospectral code Tarang for turbulence simulations

MAHENDRA K VERMA[1,*], ANANDO CHATTERJEE[1], K SANDEEP REDDY[2], RAKESH K YADAV[1], SUPRIYO PAUL[3], MANI CHANDRA[1] and RAVI SAMTANEY[4]

[1]Department of Physics; [2]Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur 208 016, India
[3]Computational Fluid Dynamics Team, Centre for Development of Advanced Computing, Pune 411 007, India
[4]Department of Mechanical Engineering, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
*Corresponding author. E-mail: mkv@iitk.ac.in

MS received 12 April 2013; accepted 13 June 2013
DOI: 10.1007/s12043-013-0594-4; ***e*Publication:** 21 September 2013

Abstract. Tarang is a general-purpose pseudospectral parallel code for simulating flows involving fluids, magnetohydrodynamics, and Rayleigh–Bénard convection in turbulence and instability regimes. In this paper we present code validation and benchmarking results of Tarang. We performed our simulations on 1024^3, 2048^3, and 4096^3 grids using the HPC system of IIT Kanpur and Shaheen of KAUST. We observe good 'weak' and 'strong' scaling for Tarang on these systems.

Keywords. Pseudospectral method; direct numerical simulations; high-performance computing.

PACS Nos 47.27.ek; 47.11.Kb; 47.27.E–

1. Introduction

A typical fluid flow is random or chaotic in the turbulent and instability regimes. Therefore, we need to employ accurate numerical schemes for simulating such flows. A pseudospectral algorithm [1,2] is one of the most accurate methods for solving fluid flows, and it is employed for performing direct numerical simulations of turbulent flows, as well as for critical applications like weather predictions and climate modelling. Yokokawa *et al* [3,4], Donzis *et al* [5], and Pouquet *et al* [6] have performed spectral simulations on some of the largest grids (e.g., 4096^3).

We have developed a general-purpose flow solver named Tarang (synonym for waves in Sanskrit) for turbulence and instability studies. Tarang is a parallel and modular code written in the object-oriented language C++. Using Tarang, we can solve incompressible flows involving pure fluid, Rayleigh–Bénard convection, passive and active scalars,

magnetohydrodynamics, liquid metals, etc. Tarang is an open-source code and it can be downloaded from http://turbulence.phy.iitk.ac.in. In this paper we shall describe some details of the code, scaling results, and code validation performed on Tarang.

2. Salient features of Tarang

The basic steps of Tarang follow the standard procedure of pseudospectral method [1,2]. However, we took several design decisions that help us scale the code, as well as solve large range of problems. Some of the design issues considered by us are:

(a) We chose an object-oriented structure for Tarang with C++ as the programing language. The modularity of the code helps us introduce new solvers very easily. Also, the basic functions, e.g., transforms, input–output, could be changed without affecting the other parts of the code. As a result, we have more than a dozen modules implemented in Tarang.
(b) Grids as large as 4096^3 take huge memory space. So, researchers have attempted to use single-precision calculations rather than double-precision calculations to save memory. Tarang allows the users to use single/double precision using a simple switch. In the present paper, we present our results for single precision. Note however that the results of single-precision and double-precision calculations are very similar (within a percent), and are consistent with the results of Yokokawa *et al* [3,4].

The basic numerical structure of Tarang follows standard pseudospectral algorithm. The Navier–Stokes and related equations are numerically solved given an initial condition of the fields. The fields are time-stepped using one of the time integrators. 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) [1,2]. 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 can involve Fourier functions, sines and cosines, Chebyshev polynomials, spherical harmonics, or a combination of these functions depending on the boundary conditions. For details, the reader is referred to Boyd [1] and Canuto *et al* [2]. Some of the specific algorithmic choices made in Tarang are as follows:

(a) In the turbulent regime, the two relevant time-scales, the large-eddy turnover time and the small-scale viscous time, are very different (orders of magnitude apart). To handle this, we use the 'exponential trick' that absorbs the viscous term using a change of variable [2].
(b) We use the fourth-order Runge–Kutta scheme for time stepping. The code however has an option to use the Euler and the second-order Runge–Kutta schemes as well.
(c) The code provides an option for dealiasing the fields. The 3/2 rule is used for dealiasing [2].
(d) The wavenumber components k_i are

$$k_i = \frac{2\pi}{L_i} n_i, \tag{1}$$

where L_i is the box dimension in the ith direction and n_i is an integer. We use parameters

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

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

The parallel implementation of Tarang involved parallelization of the spectral transforms and the input–output operations, as described below.

3. Parallelization strategy

A pseudospectral code involves forward and inverse transforms between the spectral and real space. In a typical pseudospectral code, these operations take ∼80% of the total time. Therefore, we use one of the most efficient parallel FFT routines, FFTW (fastest Fourier transform in the west) [7], in Tarang. We adopt FFTW's strategy for dividing the arrays. If p is the number of available processors, we divide each of the arrays into p 'slabs'. For example, a complex array $A(N_1, N_2, N_3/2 + 1)$ is split into $A(N_1/p, N_2, N_3/2 + 1)$ segments, each of which is handled by a single processor. This division is called 'slab decomposition'. The other time-consuming tasks in Tarang are the input and output (I/O) operations of large datasets and the element-by-element multiplication of arrays. The datasets in Tarang are massive, for example, the data size of a 4096^3 fluid simulation is of the order of 1.5 terabytes. For I/O operations, we use an efficient and parallel library named hierarchical data format-5 (HDF5). The third operation, element-by-element multiplication of arrays, is handled by individual processors in a straightforward manner.

Tarang has been organized in a modular fashion, and so the spectral transforms and I/O operations were easily parallelized. For a periodic box, we use the parallel FFTW library itself. However, for the mixed transforms (e.g., sine transform along x and Fourier transform along yz planes), we parallelize the transforms ourselves using one- and two-dimensional FFTW transforms.

An important aspect of any parallel simulation code is its scalability. We tested the scaling of FFTW and Tarang by performing simulations on 1024^3, 2048^3, and 4096^3 grids with variable number of processors. The simulations were performed on the HPC system of IIT Kanpur and Shaheen supercomputer of King Abdullah University of Science and Technology (KAUST). The HPC system has 368 compute nodes connected via a 40 Gbps Qlogic Infiniband switch with each node containing dual Intel Xeon Quad-core C5570 processor and 48GB of RAM. Its peak performance (Rpeak) is ∼34 teraflops (tera floating point operations per second). Shaheen on the other hand is a 16-rack IBM BlueGene/P system with 65536 cores and 65536 GB of RAM. Shaheen's peak performance is ∼222 teraflops.

For parallel FFT with slab decomposition, we compute the time taken per step (forward+backward transform) on Shaheen for several large N^3 grids. The results

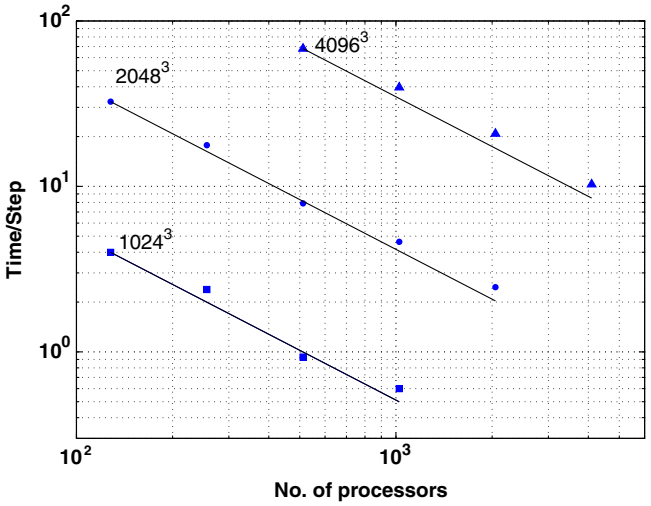

Figure 1. Scaling of parallel FFT on Shaheen for 1024^3, 2048^3, and 4096^3 grids with single-precision computation. The straight lines represent the ideal linear scaling.

displayed in figure 1 demonstrate an approximate linear scaling (called 'strong scaling'). Using the fact that each forward plus inverse FFT involves $5N^3 \log N^3$ operations for single-precision computations [7], the average FFT performance per core on Shaheen is ∼0.3 gigaflops, which is only 8% of its peak performance. Similar efficiency is observed for the HPC system as well, the cores of which have a rating of ∼12 gigaflops. The aforementioned loss of efficiency is consistent with the other FFT libraries, e.g, p3dfft [8]. Also note that an increase in the data size and the number of processors (resources) by the same amount takes approximately the same time (see figure 1). For example, FFT of a 1024^3 array using 128 processors, as well as that of a 2048^3 array on 1024 processors, takes ∼4 s as shown in table 1. Thus, our implementation of FFT shows good 'weak scaling' as well.

We also test the scaling of Tarang on Shaheen and the HPC system. Figures 2 and 3 exhibit the scaling results of fluid simulations performed on these systems. Figure 4 shows the scaling results for magnetohydrodynamics (MHD) simulation on Shaheen. These plots demonstrate strong scaling of Tarang, consistent with the aforementioned FFT

Table 1. Weak scaling of FFT using slab decomposition performed on Shaheen.

Grid size	No. of processors	Time/step (s)	% Loss
1024^3	128	3.99	
2048^3	1024	4.62	15.79
2048^3	512	7.89	
4096^3	4096	10.28	30.29

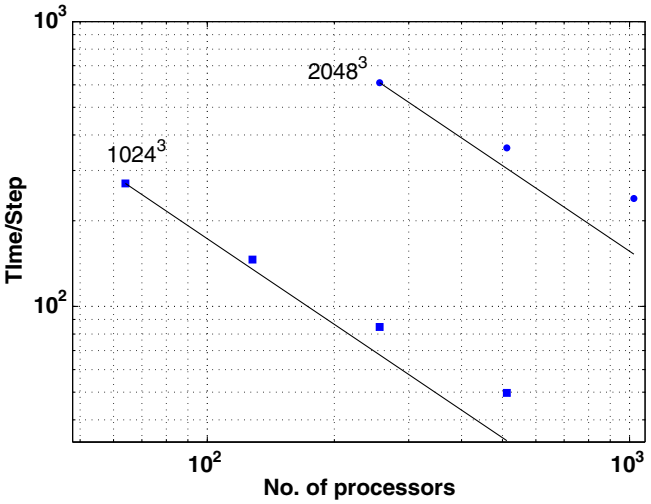

Figure 2. Scaling of Tarang's fluid solver on Shaheen for 1024^3 and 2048^3 grids with single-precision computation. The straight lines represent the ideal linear scaling.

scaling. Sometimes we observe a small loss of efficiency when $N = p$. Approximate weak scaling for the fluid turbulence solver of Tarang performed on both Shaheen and the HPC system are shown in table 2 and table 3 respectively. Approximate weak scaling for MHD solver of Tarang performed on Shaheen is shown in table 4.

A critical limitation of the 'slab decomposition' is that the number of processors cannot be more than N_1. This limitation can be overcome in a new scheme called 'pencil decomposition' in which the array $A(N_1, N_2, N_3/2 + 1)$ is split into

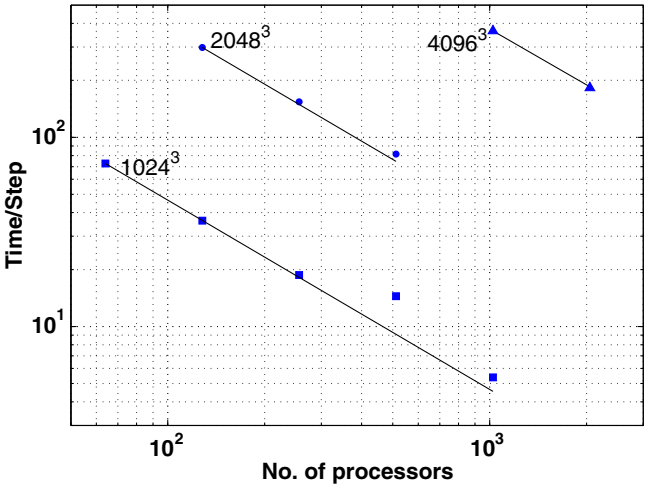

Figure 3. Scaling of Tarang's fluid solver on the HPC system of IIT Kanpur for 1024^3, 2048^3, and 4096^3 grids with single-precision computation. The straight lines represent the ideal linear scaling.

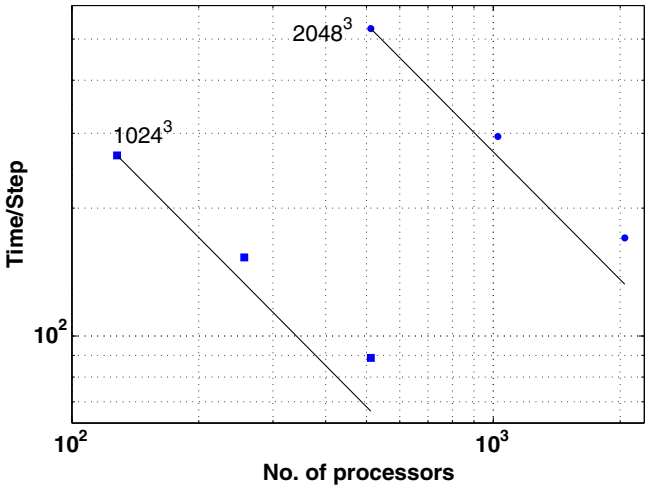

Figure 4. Scaling of Tarang's magnetohydrodynamic (MHD) solver on Shaheen for 1024^3 and 2048^3 grids with single-precision computation. The straight lines represent the ideal linear scaling.

Table 2. Weak scaling of the fluid solver of Tarang performed on Shaheen.

Grid size	No. of processors	Time/step (s)	% Loss
1024^3	64	270.00	
2048^3	512	360.17	33.39

Table 3. Weak scaling of the fluid solver of Tarang performed on HPC system.

Grid size	No. of processors	Time/step (s)	% Loss
1024^3	64	72.70	
2048^3	512	81.56	12.19
2048^3	256	153.93	
4096^3	2048	184.88	16.74

Table 4. Weak scaling of the MHD solver of Tarang performed on Shaheen.

Grid size	No. of processors	Time/step (s)	% Loss
1024^3	128	266.08	
2048^3	1024	294.99	10.86
1024^3	256	153.04	
2048^3	2048	170.16	11.18

$A(N_1/p_1, N_2/p_2, N_3/2+1)$ pencils where the total number of processors $p = p_1 \times p_2$ [8]. We are in the process of implementing 'pencil decomposition' on Tarang. In this paper we shall focus only on the 'slab decomposition'.

After the above discussion on parallelization of the code, we shall discuss code validation, and time and space complexities for simulations of fluid turbulence, Rayleigh–Bénard convection, and magnetohydrodynamic turbulence.

4. Fluid turbulence

The governing equations for incompressible fluid turbulence are

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}^u, \tag{3}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{4}$$

where $\mathbf{u}$ is the velocity field, p is the pressure field, ν is the kinematic viscosity, and $\mathbf{F}^u$ is the external forcing. For studies on homogeneous and isotropic turbulence, simulations are performed on high-resolution grids (e.g., 2048^3, 4096^3) with a periodic boundary condition. The resolution requirement is stringent due to $N \sim \mathrm{Re}^{3/4}$ relation; for $\mathrm{Re} = 10^5$, the required grid resolution is $\sim 5600^3$, which is quite challenging even for modern supercomputers.

Regarding the space complexity of a forced fluid turbulence simulation, Tarang requires 15 arrays (for $\mathbf{u}(\mathbf{k})$, $\mathbf{u}(\mathbf{r})$, $\mathbf{F}^u(\mathbf{k})$, $\mathbf{nlin}(\mathbf{k})$, and three temporary arrays), which translates to $\sim$120 gigabytes (8 terabytes) of memory for 1024^3 (4096^3) double-precision computations. Here $\mathbf{k}$ and $\mathbf{r}$ represent the wavenumbers and the real space coordinates respectively. The requirement is halved for a simulation with single precision. Regarding the time requirement, each numerical step of the fourth-order Runge–Kutta (RK4) scheme requires 9×4 FFT operations. The factor 9 is due to the three inverse and six forward transforms performed for each of the four RK4 iterates. Therefore, for every time-step, all the FFT operations require $36 \times 2.5 \times N^3 \log_2(N^3)$ multiplications for a single-precision simulation [7], which translates to $\sim$2.9 (185) terafloating-point operations for 1024^3 (4096^3) grids. The number of operations for double-precision computation is twice that of the above estimate. On 128 processors on HPC system, a fluid simulation with single precision takes $\sim$36 s (see figure 3), which corresponds to a per core performance of $\sim$0.68 gigaflops. This is only 6% of the peak performance of the cores, which is consistent with the efficiency of FFT operations discussed in §3. Also note that the solver also involves other operations, e.g., element-by-element array multiplication, but these operations take only a small fraction of the total time.

We can also estimate the total time required to perform a 4096^3 fluid simulation. A typical fluid turbulence would require five eddy turnover time with $\mathrm{d}t \approx 5 \times 10^{-4}$, which corresponds to 10^4 time-steps for the simulations. So the total floating point operations required for this single-precision simulation is 185×10^4 terafloating-point operations for the FFT itself. Assuming 5% efficiency for FFT, and FFTs share being 80% of the total time, the aforementioned fluid simulation will take $\sim$128 h on a 100 teraflop cluster.

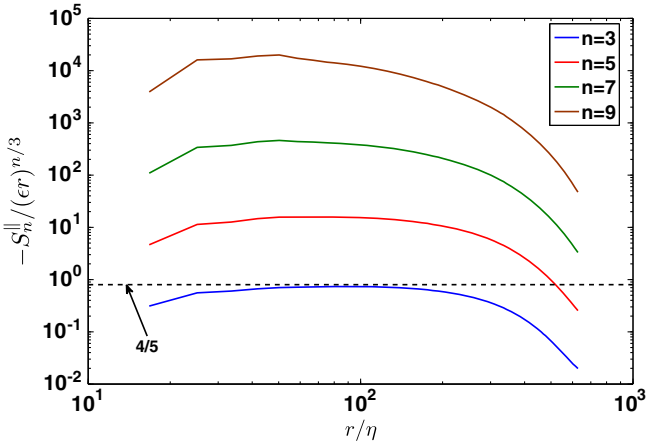

Figure 5. Plots of the normalized odd-order structure functions $-S_n^{\parallel}(r)/(\epsilon r)^{n/3}$ vs. r/η for a fluid simulation using Tarang. Here ϵ is the energy flux and η is the Kolmogorov scale.

We perform code validation of the fluid solver using Kolmogorov's theory for the third-order structure function [9], according to which

$$S_3^{\parallel}(r) = \langle \{u_{\parallel}(\mathbf{x}+\mathbf{r}) - u_{\parallel}(\mathbf{x})\}^3 \rangle = -\frac{4}{5}\epsilon r, \tag{5}$$

where ϵ is the energy flux in the inertial range and $\langle \cdots \rangle$ represents ensemble averaging (here spatial averaging). We compute the structure function $S_3^{\parallel}(r)$, as well as $S_5^{\parallel}(r)$, $S_7^{\parallel}(r)$, and $S_9^{\parallel}(r)$ for the steady-state dataset of a fluid simulation on a 1024^3 grid. The computed values of $S_q^{\parallel}(r)$ are illustrated in figure 5 that shows a good agreement with Kolmogorov's theory.

After the discussion on fluid solver, we move on to the module for solving Rayleigh–Bénard convection.

5. Rayleigh–Bénard convection

Rayleigh–Bénard convection (RBC) is an idealized model of convection in which fluid is confined between two plates that are separated by a distance d, and are maintained at temperatures T_0 and $T_0 - \Delta$. The equations for the fluid under Boussinesq approximations are

$$\partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{\nabla\sigma}{\rho_0} + \alpha g\theta\hat{z} + \nu\nabla^2\mathbf{u}, \tag{6}$$

$$\partial_t\theta + (\mathbf{u}\cdot\nabla)\theta = \frac{\Delta}{d}u_z + \kappa\nabla^2\theta, \tag{7}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{8}$$

where θ is the temperature fluctuation ($T = T_c + \theta$ with T_c as the conduction temperature profile), σ is the pressure fluctuations from the steady conduction state, $\hat{z}$ is the buoyancy direction, Δ is the temperature difference between the two plates, ν is the kinematic

viscosity, and κ is the thermal diffusivity. We solve the nondimensionalized equations, which are obtained using d as the length scale, κ/d as the velocity scale, and Δ as the temperature scale:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla\sigma + \mathrm{Ra}\,\mathrm{Pr}\,\theta\hat{z} + \mathrm{Pr}\,\nabla^2\mathbf{u}, \tag{9}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = u_3 + \nabla^2\theta. \tag{10}$$

Here the two important nondimensional parameters are the Rayleigh number $\mathrm{Ra} = \alpha g \Delta d^3/\nu\kappa$ and the Prandtl number $\mathrm{Pr} = \nu/\kappa$. At present, we can apply the free-slip boundary condition for the velocity fields at the horizontal plates, i.e.,

$$u_3 = \partial_z u_1 = \partial_z u_2 = 0, \quad \text{for } z = 0, 1, \tag{11}$$

and isothermal boundary condition on the horizontal plates

$$\theta = 0, \quad \text{for } z = 0, 1. \tag{12}$$

Periodic boundary conditions are applied to the vertical boundaries. Note that the free-slip boundary condition is not as common as no-slip boundary condition, still it is quite useful in many situations, e.g., the upper layer of the atmosphere, flow involving two immiscible liquid, etc.

The number of arrays required for a RBC simulation is 18 (15 for fluids plus three for $\theta(\mathbf{k})$, $\theta(\mathbf{r})$, $\mathbf{nlin}^{\theta}$). Thus, the memory requirement for RBC is (18/15) times that for the fluid simulation. Regarding the time complexity, the number of FFT operations required per time-step is 13×4 FFT operations (4 inverse + 9 forward transforms per RK4 step). As a result, the total time requirement for a RBC simulation is (13/9) times the respective fluid simulation.

For code validation of Tarang's RBC solver, we compare the Nusselt number $\mathrm{Nu} = 1 + \langle u_3\theta\rangle$ computed using Tarang with that computed by Thual [10] for a two-dimensional simulation with the free-slip boundary condition. The analysis is performed for the steady-state dataset. The comparative results shown in table 5 illustrate excellent

Table 5. Verification of Tarang against Thual's [10] 2D RBC simulations. We compare Nusselt numbers (Nu) computed in our simulations on a 64^2 grid against Thual's simulations on 16^2 (THU1), 32^2 (THU2), and 64^2 (THU3) grids. All Nu values tabulated here are for $\mathrm{Pr} = 6.8$.

r	THU1	THU2	THU3	Tarang
2	2.142	–	–	2.142
3	2.678	–	–	2.678
4	3.040	3.040	–	3.040
6	3.553	3.553	–	3.553
10	4.247	4.244	–	4.243
20	5.363	5.333	5.333	5.333
30	6.173	6.105	6.105	6.105
40	6.848	6.742	6.740	6.740
50	7.441	7.298	7.295	7.295

agreement between the two runs. We also compute the Nusselt number for a three-dimensional flow with Pr = 6.8 and observe that Nu = $(0.27 \pm 0.04)(\mathrm{PrRa})^{0.27\pm0.01}$ [11], which is in good agreement with earlier experimental and numerical results.

Using the RBC module of Tarang, we also studied the energy spectra and fluxes of the velocity and temperature fields [12], the Nusselt number scaling [11], and chaos and bifurcations near the onset of convection [13,14].

In the next section we shall discuss the results of the MHD module of Tarang.

6. Magnetohydrodynamic turbulence and dynamo

The equations for the incompressible MHD turbulence [15] are

$$\partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + (\mathbf{B}\cdot\nabla)\mathbf{B} + \nu\nabla^2\mathbf{u} + \mathbf{F}^u, \tag{13}$$

$$\partial_t \mathbf{B} + (\mathbf{u}\cdot\nabla)\mathbf{B} = (\mathbf{B}\cdot\nabla)\mathbf{u} + \eta\nabla^2\mathbf{B} + \mathbf{F}^B, \tag{14}$$

$$\nabla\cdot\mathbf{u} = \nabla\cdot\mathbf{B} = 0, \tag{15}$$

where $\mathbf{u}$, $\mathbf{B}$, and p are the velocity, magnetic, and pressure (thermal+magnetic) fields respectively, ν is the kinematic viscosity, and η is the magnetic diffusivity. The $\mathbf{F}^u$ and $\mathbf{F}^B$ are the external forcing terms for the velocity and magnetic fields respectively. Typically, $\mathbf{F}^B = 0$, but Tarang implements $\mathbf{F}^B$ for generality. The magnetic field $\mathbf{B}$ can be separated into its mean $\mathbf{B}_0$ and fluctuations $\mathbf{b}$: $\mathbf{B} = \mathbf{B}_0 + \mathbf{b}$. The number of nonlinear terms in the above equations is four whose computation requires 27 FFTs. However, the number of FFT computations in terms of the Elsasser variables $\mathbf{z}^\pm = \mathbf{u} \pm \mathbf{b}$ is only 15, thus saving significant computing time. We use the relations

$$(\mathbf{u}\cdot\nabla)\mathbf{u} - (\mathbf{B}\cdot\nabla)\mathbf{B} = (\mathbf{z}^-\cdot\nabla)\mathbf{z}^+ + (\mathbf{z}^+\cdot\nabla)\mathbf{z}^-, \tag{16}$$

$$(\mathbf{u}\cdot\nabla)\mathbf{B} - (\mathbf{B}\cdot\nabla)\mathbf{u} = (\mathbf{z}^-\cdot\nabla)\mathbf{z}^+ - (\mathbf{z}^+\cdot\nabla)\mathbf{z}^- \tag{17}$$

to compute the nonlinear terms. Thus, the time requirement for a MHD simulation would be around 15/9 times that for the fluid simulation. In figure 4 we plot the time taken per step for different sets of processors on Shaheen. The results are consistent with the above estimates. Regarding the space complexity, an MHD simulation requires 27 arrays for storing $\mathbf{u}(\mathbf{k})$, $\mathbf{B}(\mathbf{k})$, $\mathbf{B}(\mathbf{r})$, $\mathbf{u}(\mathbf{r})$, $\mathbf{F}^u(\mathbf{k})$, $\mathbf{F}^B(\mathbf{k})$, $\mathbf{nlin}^u(\mathbf{k})$, $\mathbf{nlin}^B(\mathbf{k})$, and three temporary fields. Hence, the memory requirement for a MHD simulation is 27/15 times that of a fluid simulation.

We perform code validation of Tarang's MHD module using the results of Breyiannis and Valougeorgis's [16] lattice kinetic simulations of three-dimensional decaying MHD. Following Breyiannis and Valougeorgis, we solve the MHD equations inside a cube with periodic boundary conditions on all directions, and with a Taylor–Green vortex (given below) as an initial condition,

$$\mathbf{u} = [\sin(x)\cos(y)\cos(z), -\cos(x)\sin(y)\cos(z), 0], \tag{18}$$

$$\mathbf{B} = [\sin(x)\sin(y)\cos(z), \cos(x)\cos(y)\cos(z), 0]. \tag{19}$$

This Taylor–Green vortex is then allowed to evolve freely. The simulation box is discretized using 32^3 grid points.

The results of this test case for different parameter values ($\nu = \eta = 0.01, 0.05, 0.1$) are presented in figure 6. The top and bottom panels exhibit the time evolution of the total kinetic and magnetic energies respectively. Tarang's datapoints, illustrated using blue dots, are in excellent agreement with Breyiannis and Valougeorgis' results [16], which is represented by solid lines. We thus verify the MHD module of Tarang.

We have used Tarang to perform extensive simulations of dynamo transition under the Taylor–Green forcing [17,18]. Using Tarang, we have also computed the magnetic and

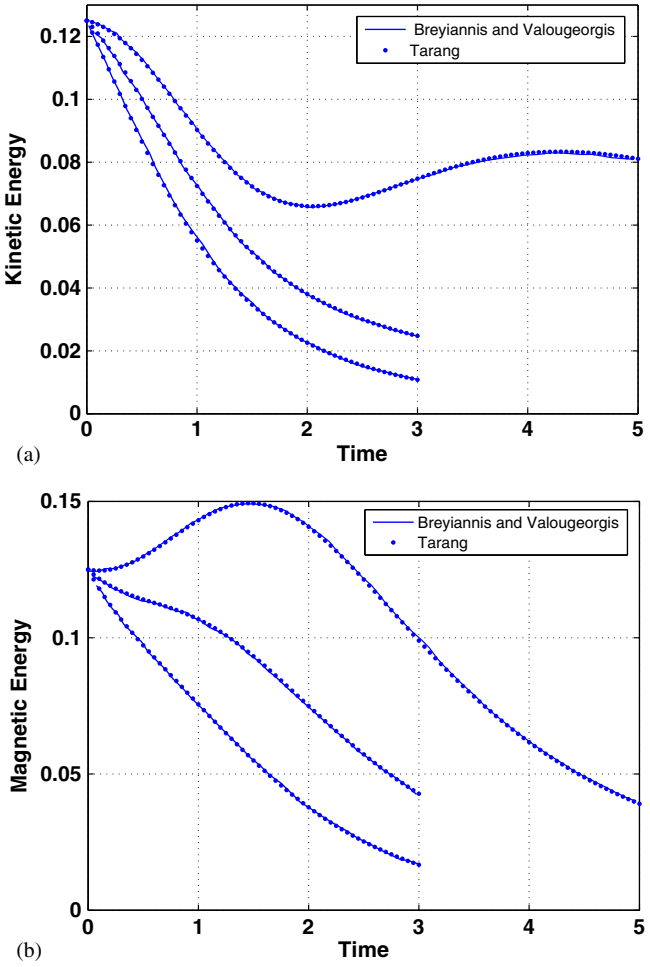

Figure 6. Time evolution of total kinetic energy (**a**) and total magnetic energy (**b**) for a decaying MHD simulation with Taylor–Green vortex as an initial condition. Blue dots are Tarang's datapoints, while the solid lines are the lattice simulation results of Breyiannis and Valougeorgis [16]. The three different curves reported here are for $\nu = \eta = 0.01, 0.05, 0.1$ from top to bottom.

kinetic energy spectra, various energy fluxes [15], and shell-to-shell energy transfers for MHD turbulence; these results would be presented in a subsequent paper.

In addition to the fluid, MHD, and Rayleigh–Bénard convection solvers, Tarang has modules for simulating rotating turbulence, passive and active scalars, liquid metal flows, rotating convection [19], and Kolmogorov flow.

7. Conclusions

In this paper we describe the salient features and code validation of Tarang. Tarang passes several validation tests performed for fluid, Rayleigh–Bénard convection, and magnetohydrodynamic solvers. We also report scaling analysis of Tarang and show that it exhibits excellent strong and weak scaling up to several thousand processors. Tarang has been used for studying Rayleigh–Bénard convection, dynamo, and magnetohydrodynamic turbulence. It has been ported to various computing platforms including the HPC system of IIT Kanpur, Shaheen of KAUST, PARAM YUVA of the Centre for Advanced Computing (Pune), and EKA of the Computational Research Laboratory (Pune).

Acknowledgement

Tarang simulations were performed on Shaheen supercomputer of KAUST (through the project k97) and on the HPC system of IIT Kanpur, for which the authors thank the personnels of respective Supercomputing Centres, especially Abhishek and Brajesh Pande of IIT Kanpur. The authors are grateful to Sandeep Joshi and Late Dr V Sunderarajan (CDAC) who encouraged them to run Tarang on very large grids. The authors also thank Daniele Carati and his group at ULB Brussels for sharing the details of a pseudospectral code, and CDAC engineers and Arvind Mishra for their help at various stages. MKV acknowledges the support of Swaranajayanti fellowship and a research grant 2009/36/81-BRNS from Bhabha Atomic Research Centre.

References

[1] J P Boyd, *Chebyshev and Fourier spectral methods* (Dover Publishers, New York, 2001)
[2] C Canuto, M Y Hussaini, A Quarteroni and T A Zhang, *Spectral methods in fluid turbulence* (Springer-Verlag, Berlin, 1998)
[3] M Yokokawa, K Itakura, A Uno, T Ishihara and Y Kaneda, 16.4-TFlops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator, Tech. Rep., dspace.itri.aist.go.jp (2002)
[4] Y Kaneda, T Ishihara, M Yokokawa, K Itakura and A Uno, *Phys. Fluids* **15**, L21 (2003)
[5] Diego Donzis, K Sreenivasan and P Yeung, *Flow, Turbulence and Combustion* **85**, 549 (2010)
[6] A Pouquet, J Baerenzung, P D Mininni, D Rosenberg and S Thalabard, *J. Phys.: Conf. Ser.* **318(4)**, 042015 (2011)
[7] M Frigo and S G Johnson, *Proceedings of the IEEE* **93(2)**, 216 (2005). Special issue on *Program Generation, Optimization, and Platform Adaptation*
[8] Parallel three-dimensional fast Fourier transforms (P3DFFT) library, http://code.google.com/p/p3dfft (2008)

[9] A N Kolmogorov, *Dokl. Akad. Nauk SSSR* **30**, 9 (1941)
[10] O Thual, *J. Fluid Mech.* **240**, 229 (1992)
[11] M K Verma, P K Mishra, A Pandey and S Paul, *Phys. Rev. E* **85**, 016310 (2012)
[12] P K Mishra and M K Verma, *Phys. Rev. E* **81**, 056316 (2010)
[13] P Pal, P Wahi, S Paul, M K Verma, K Kumar and P K Mishra, *Europhys. Lett.* **87**, 54003 (2009)
[14] S Paul, P Pal, P Wahi and M K Verma, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **21**, 023118 (2011)
[15] M K Verma, *Phys. Rep.* **401**, 229 (2004)
[16] G Breyiannis and D Valougeorgis, *Comput. Fluids* **35(8)**, 920 (2006)
[17] R Yadav, M Chandra, M K Verma, S Paul and P Wahi, *Europhys. Lett.* **91**, 69001 (2010)
[18] Rakesh K Yadav, Mahendra K Verma and Pankaj Wahi, *Phys. Rev. E* **85**, 036301 (2012)
[19] Hirdesh K Pharasi, Rahul Kannan, Krishna Kumar and Jayanta K Bhattacharjee, *Phys. Rev. E* **84**, 047301 (2011)