Commun. Comput. Phys.
doi: 10.4208/cicp.191209.111110s

Vol. **9**, No. 5, pp. 1152-1164
May 2011

A Memory-Saving Algorithm for Spectral Method of Three-Dimensional Homogeneous Isotropic Turbulence

Qing-Dong Cai[1] and Shiyi Chen[1,2,*]

[1] *LTCS and CAPT, Department of Mechanics and Aerospace Engineering, College of Engineering, Peking University, Beijing 100871, China.*
[2] *Department of Mechanical Engineering, The Johns Hopkins University, Baltimore, Maryland 21218, USA.*

Received 19 December 2009; Accepted (in revised version) 11 November 2010

Available online 14 January 2011

Abstract. Homogeneous isotropic turbulence has been playing a key role in the research of turbulence theory. And the pseudo-spectral method is the most popular numerical method to simulate this type of flow fields in a periodic box, where fast Fourier transform (FFT) is mostly effective. However, the bottle-neck in this method is the memory of computer, which motivates us to construct a memory-saving algorithm for spectral method in present paper. Inevitably, more times of FFT are needed as compensation. In the most memory-saving situation, only 6 three-dimension arrays are employed in the code. The cost of computation is increased by a factor of 4, and that 38 FFTs are needed per time step instead of the previous 9 FFTs. A simulation of isotropic turbulence on 2048^3 grid can be implemented on a 256G distributed memory clusters through this method.

AMS subject classifications: 80M22, 76F05, 76F65, 65T50

Key words: Spectral method, homogeneous isotropic turbulence, DNS, FFT.

1 Introduction

Homogeneous and isotropic (HI) turbulence has been a paradigm theoretically since it was introduced by G. I. Taylor [1] and it assumed even greater importance after the fundamental work of Kolmogorov [2–4]. Due to predicted "universality" at small-scales, turbulent motions smaller than those at lengths where production occurs are expected to be approximately HI in many applications. Nevertheless, even this simplest state of turbulence has resisted intensive efforts at complete understanding. Large scale direct

*Corresponding author. *Email addresses:* caiqd@pku.edu.cn (Q.-D. Cai), syc@coe.pku.edu.cn (S. Chen)

numerical simulations (DNS) of HI turbulence are urgently needed for the understanding of the small scale behaviors in turbulence. It is well know that the most popular numerical method for the simulation of HI turbulence is pseudo-spectral method [5,6]. The simulation scales from 64^3 [7], 128^3 [8–10] to 512^3 [11–15]. NEC's Earth Simulator at 36 teraflops led the TOP500 super computing list for two and a half years, from June 2002 until September 2004, when it was overtaken by IBM's Blue Gene. The landmark 4096^3 simulation performed on the Earth Simulator [16–20] remains the highest-resolution computation of three-dimensional (3D) homogeneous isotropic turbulence conducted to date.

The rapid progress of super computers has made the simulation of turbulence more and more powerful although it is still far from the resolution of Kolomogorov scale. With spectral method, the major problem is the large demand of memory instead of CPU time. For example, in order to simulate a three dimensional incompressible turbulence on a N^3 mesh, now the best code needs 12.5 real arrays of size N^3, which may still need another one in 3D fast Fourier transform (FFT) as a work array [11]. After that, the total memory in bytes will be $4\times12.5N^3=50N^3$ (assuming only single precision real number is used). If $N=1024$, the total amount will be about 54G, if $N=2048$, this amount will be up to about 429G. This makes it impossible to simulate a 2048^3 size flow on a 256G memory machine (the clusters we used have 256G memory, which is a popular configuration for university machines). In order to do this, we must reduce the number of arrays from 12.5 to at least to 7, then $4\times7N^3=28N^3$, which will be about 241G when $N=2048$. If additional array is needed in FFT program, this amount will be 275G, which is a little large. This means we must design a method with only use at most 7 arrays of size N^3 in order to finish this task. From the published papers, such as [15] and others, the number of arrays is more than ten. Through precise design, a memory-saving algorithm for spectral method of three-dimensional homogeneous isotropic turbulence is present, and the coding details are listed in Appendix.

The outline of the paper is as follows: in the next part, the standard algorithm of pseudo-spectral method will be introduced briefly, and the idea of our new algorithm will also be presented as well. Subsequently in the third part, we show the numerical results and performance. Finally, the summary and discussions are given in the forth part.

2 Numerical method

Almost all DNS of fluid turbulence are implemented based on the pseudo-spectral method, which solves the Navier-Stokes equation directly in the Fourier space. We review briefly the standard algorithm [5,6], and then point out the basic idea of our new algorithm. The incompressible Navier-Stokes equation can be written as

$$\begin{cases} \nabla\cdot\mathbf{v}=0, \\ \dfrac{\partial\mathbf{v}}{\partial t}-\mathbf{v}\times\boldsymbol{\omega}=-\nabla P+\nu\Delta\mathbf{v}+\mathbf{f}, \end{cases} \tag{2.1}$$

where $\mathbf{v}$ is velocity, and ν is kinematic viscosity. $\mathbf{f}$ is an external force term to be specified, and $P=p+|\mathbf{v}|^2/2$, p is pressure, $\boldsymbol{\omega}=\nabla\times\mathbf{v}$ is vorticity vector of flow field. The Eq. (2.1) can be transformed to spectral space by Fourier transformation

$$\frac{\partial\hat{\mathbf{v}}(\mathbf{k},t)}{\partial t}=\mathbf{P}(\mathbf{k})\cdot\widehat{(\mathbf{v}\times\boldsymbol{\omega})}-\nu k^2\hat{\mathbf{v}}(\mathbf{k},t)+\mathbf{P}(\mathbf{k})\cdot\hat{\mathbf{f}}, \tag{2.2}$$

where overcarets denote Fourier coefficients, $\widehat{(\mathbf{v}\times\boldsymbol{\omega})}$ indicating that the cross times term $\mathbf{v}\times\boldsymbol{\omega}$ is finished in physical space, then the new quantity is transformed to spectral space, $\mathbf{k}$ is wave vector, and $k=|\mathbf{k}|$ is wave number. The tensor $\mathbf{P}(\mathbf{k})$ is divergence free projector in spectral space

$$P_{ij}(\mathbf{k})=\delta_{ij}-\frac{k_ik_j}{k^2}. \tag{2.3}$$

The incompressible condition becomes $\mathbf{k}\cdot\hat{\mathbf{v}}(\mathbf{k},t)=0$ in spectral space. Eq. (2.2) is widely used in pseudo-spectral method for simulation of isotropic turbulence [5,6,11]. We didn't try to change the spectral method itself, however, only intending to realize it in a different way in order to save memory as much as possible. Certainly, it may need more CPU time as compensation. The ordinary method usually needs three arrays for velocity vector, three arrays for vorticity vector and another three for cross times term in Eq. (2.2). With some work arrays, there are 12.5 N^3 arrays all together. The computation procedure to advance one step from $\hat{\mathbf{v}}^n$ to $\hat{\mathbf{v}}^{n+1}$ is:

1. obtain $\hat{\boldsymbol{\omega}}^n$ according to $\hat{\boldsymbol{\omega}}^n=i\mathbf{k}\times\hat{\mathbf{v}}^n$;
2. transform velocity and vorticity in spectral space to physical space $\hat{\mathbf{v}}^n\to\mathbf{v}^n$, $\hat{\boldsymbol{\omega}}^n\to\boldsymbol{\omega}^n$;
3. compute $\mathbf{v}\times\boldsymbol{\omega}$ and transform to spectral space to obtain $\widehat{\mathbf{v}\times\boldsymbol{\omega}}$ and conducting projection;
4. integrate Eq. (2.2) to obtain $\hat{\mathbf{v}}^{n+1}$ and do projection.

This is a general computational procedure to implement pseudo-spectral method, with 9 FFTs in every time step. We label this as 12.5A method according to the number of N^3 arrays used in the code. In order to save memory, first we notice that not all components of velocity and vorticity vector are needed because of divergence free condition in Fourier space. This allows us to use only two arrays for velocity, as we use $\hat{v}_x$ and $\hat{v}_z$ for convenience, and $\hat{v}_y$ can be gained if needed by

$$\hat{v}_y(\mathbf{k},t)=-\frac{[k_x\hat{v}_x(\mathbf{k},t)+k_z\hat{v}_z(\mathbf{k},t)]}{k_y}. \tag{2.4}$$

To avoid the trouble at $k_y=0$, the velocity component $\hat{v}_y$ at $k_y=0$ plane needed to be saved separately. In this way, Eq. (2.2) becomes

$$\begin{cases}\dfrac{\partial\hat{v}_x(\mathbf{k},t)}{\partial t}=\left(\mathbf{P}(\mathbf{k})\cdot\widehat{(\mathbf{v}\times\boldsymbol{\omega})}\right)_x-\nu k^2\hat{v}_x(\mathbf{k},t)+\left(\mathbf{P}(\mathbf{k})\cdot\hat{\mathbf{f}}\right)_x,\\ \dfrac{\partial\hat{v}_z(\mathbf{k},t)}{\partial t}=\left(\mathbf{P}(\mathbf{k})\cdot\widehat{(\mathbf{v}\times\boldsymbol{\omega})}\right)_z-\nu k^2\hat{v}_z(\mathbf{k},t)+\left(\mathbf{P}(\mathbf{k})\cdot\hat{\mathbf{f}}\right)_z.\end{cases} \tag{2.5}$$

The subscript x and z denote the index of component. The most troublesome term in the above equations is $\mathbf{P}(\mathbf{k})\cdot(\widehat{\mathbf{v}\times\boldsymbol{\omega}})$, which can be written out as

$$\left[\mathbf{P}(\mathbf{k})\cdot(\widehat{\mathbf{v}\times\boldsymbol{\omega}})\right]_i = P_{ij}(\widehat{\mathbf{v}\times\boldsymbol{\omega}})_j, \tag{2.6}$$

in addition to

$$(\widehat{\mathbf{v}\times\boldsymbol{\omega}})_1 = \widehat{v_y\omega_z} - \widehat{v_z\omega_y}, \tag{2.7a}$$

$$(\widehat{\mathbf{v}\times\boldsymbol{\omega}})_2 = \widehat{v_z\omega_x} - \widehat{v_x\omega_z}, \tag{2.7b}$$

$$(\widehat{\mathbf{v}\times\boldsymbol{\omega}})_3 = \widehat{v_x\omega_y} - \widehat{v_y\omega_x}. \tag{2.7c}$$

Thus we have

$$\left[\mathbf{P}(\mathbf{k})\cdot(\widehat{\mathbf{v}\times\boldsymbol{\omega}})\right]_x = \left(1-\frac{k_x^2}{k^2}\right)(\widehat{v_y\omega_z}-\widehat{v_z\omega_y}) - \frac{k_xk_y(\widehat{v_z\omega_x}-\widehat{v_x\omega_z})}{k^2} - \frac{k_xk_z(\widehat{v_x\omega_y}-\widehat{v_y\omega_x})}{k^2}, \tag{2.8a}$$

$$\left[\mathbf{P}(\mathbf{k})\cdot(\widehat{\mathbf{v}\times\boldsymbol{\omega}})\right]_z = -\frac{k_xk_z(\widehat{v_y\omega_z}-\widehat{v_z\omega_y})}{k^2} - \frac{k_yk_z(\widehat{v_z\omega_x}-\widehat{v_x\omega_z})}{k^2} + \left(1-\frac{k_z^2}{k^2}\right)(\widehat{v_x\omega_y}-\widehat{v_y\omega_x}). \tag{2.8b}$$

All terms must be calculated one by one to avoid keeping them all to save memory space, however, more FFT calculations are needed as we have to use the velocity components both in physical and spectral space. We only assign one array for vorticity. When we want to use it anytime, we compute it in spectral space by

$$\hat{\boldsymbol{\omega}}(\mathbf{k},t) = i\mathbf{k}\times\hat{\mathbf{v}}(\mathbf{k},t), \tag{2.9}$$

which are

$$\hat{\omega}_x = i(k_y\hat{v}_z - k_z\hat{v}_y), \qquad \hat{\omega}_y = i(k_z\hat{v}_x - k_x\hat{v}_z), \qquad \hat{\omega}_z = i(k_x\hat{v}_y - k_z\hat{v}_x), \tag{2.10}$$

with an inverse FFT applied to it if we need vorticity in physical space.

The equation for $\hat{v}_y$ is not listed above because we need not to solve all v_y equation at present, but only those whose wave vectors are at $k_y=0$ plane are solved by:

$$\frac{\partial\hat{v}_y(k_x,0,k_z,t)}{\partial t} = \left(\mathbf{P}(\mathbf{k})\cdot(\widehat{\mathbf{v}\times\boldsymbol{\omega}})\right)_y - \nu k^2\hat{v}_y(k_x,0,k_y,t) + \left(\mathbf{P}(\mathbf{k})\cdot\hat{\mathbf{f}}\right)_y, \tag{2.11}$$

and in this plane, the convection term which is analogue of Eq. (2.8) has only one term left

$$\left[\mathbf{P}(\mathbf{k})\cdot(\widehat{\mathbf{v}\times\boldsymbol{\omega}})\right]_y = (\widehat{v_z\omega_x}-\widehat{v_x\omega_z}), \quad \text{when } k_y=0. \tag{2.12}$$

The time integration of Eqs. (2.5) and (2.11) is achieved by second-order Runge-Kutta method which have 2nd-order accuracy in time, which is more stable than Crank-Nicholson scheme for model equation. For Eq. (2.5), we have

$$\begin{cases} \hat{v}_x^{\frac{1}{2}} = \hat{v}_x^n + \dfrac{1}{2}\Delta t R_x(\hat{\mathbf{v}}^n), \quad \hat{v}_z^{\frac{1}{2}} = \hat{v}_z^n + \dfrac{1}{2}\Delta t R_z(\hat{\mathbf{v}}^n), \\ \hat{v}_x^{n+1} = \hat{v}_x^n + \Delta t R_x(\hat{\mathbf{v}}^{\frac{1}{2}}), \quad \hat{v}_z^{n+1} = \hat{v}_z^n + \Delta t R_z(\hat{\mathbf{v}}^{\frac{1}{2}}), \end{cases} \tag{2.13}$$

where $R_x(\cdot)$ and $R_z(\cdot)$ stand for the right-hand-side term of Eq. (2.5), respectively. And they must be computed on a general basis according to the expansion in Eq. (2.8), which will be the most time consuming part in the whole computation. The first step in (2.13) need four arrays to save $\hat{v}_x$,$\hat{v}_z$,$\hat{v}_x^{1/2}$ and $\hat{v}_z^{1/2}$. Another two arrays are needed to save a certain vorticity component and $\hat{v}_y$, the first of which is also used to save the product of this vorticity component with a velocity component appeared in Eq. (2.8), such as $v_y\omega_z$, $v_z\omega_y$, etc. At the second step of (2.13), the new results can be directly added to the arrays stored as $\hat{v}_x$ and $\hat{v}_z$ because the results in n-time level are no longer used further, such that no additional array is needed. In the whole computation, six N^3 size arrays are used. As we mentioned in the above analysis, the v_y array can be dropped off. Obviously, this will need more computation between velocity components if we only save two of the three velocity components at the same time. Since our objective is to use less than seven arrays, we keep this one to make the code much easier and more efficient. In fact, we use it as a work array to store the product of velocity component and vorticity component. Without this array, we must use the vorticity array itself to store this product, which will require more computations as when we need to compute vorticity, all velocity components must be transformed to spectral space, but the product is computed in physical space. 19 FFTs are needed in each half step, however, with one more work array, 13 FFTs are still needed. Those two methods above will be cited as 6A method and 7A method respectively. They use 38 and 26 FFTs per time step. This then explains why the computation can be 26/9 to 38/9 or 2.9 to 4.2 or roughly 3 to 4 times more expensive.

3 Numerical results and performance

The computer we used is an integrated cluster with 256 nodes, and each of them has two processors. The memory of every node is 1G, all together there are 256G memory.

In order to validate the new code, we compute a 512^3 problem with 32 nodes 64 processes, which takes about 22 seconds per step. The initial value is the results of 12.5A method which takes about 5 seconds per step, and a steady spectrum has been reached already. We compute more steps with 6A method, and draw the spectrum every 50 steps. Fig. 1 shows the steady spectrum in farther computation to present good agreement among those spectrum distributions.

As we have pointed out, 6A method is time-consuming, because more FFTs are needed. We compare the computational time at different grid size in Fig. 2. All computations used 4 or 8 processors, which may come from 4 or 8 different nodes. If we use 4 nodes, the total memory is 4GB, it will be 8G when 8 nodes are used altogether. The 6A method takes more time than that of 12.5A one, and 8-node computation is a little faster. The abnormal time cost of 12.5A method at grid size 512 by using 4 node is because of the large demand of memory (about 6.75GB), which has exceeded 4GB, the swap space has been used. It is also noticed that the computation with 4 nodes (when the whole 4GB memory is available) need more time than that using 8 nodes.

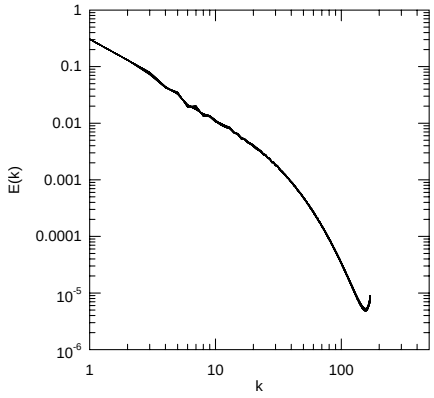

Figure 1: Energy spectrum on 512^3 grid, the computation is implemented on 32 nodes with 64 processes. First, a steady spectrum is gained by 12.5A method, then 6A method is implemented to forward 1500 steps with the same time step $\Delta t=0.0004$, outputting the spectrum every 50 steps, we get 30 distributions of energy spectrum. Draw them together, the consistence means the spectrum is still stable. If we compute from the same initial condition with only 6A method, we will get the same results.

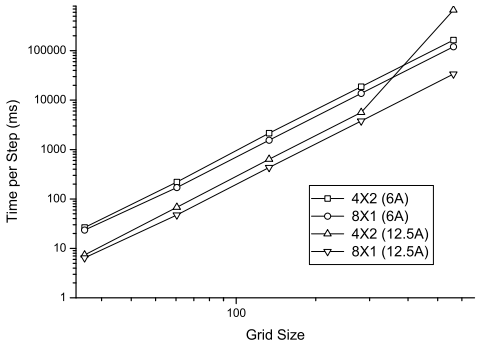

Figure 2: Computational time per step for different grid size. The computation is with $n=4$ or 8 nodes and $np=8$ processes. The 12.5A method is more efficient than 6A one as expected. For 512^3 grid, the 12.5A method needs much more time because of inadequacy of memory, and at the same time, swap space is used. As for 6A method, the memory is still enough.

The purpose to develop this 6A method is to reduce the demand of memory in classical spectral method. We can solve 512^3 problem with only 4 nodes (every node starts two processors), whose total memory is 4GB. The needed memory is $512^3\cdot 4\cdot(6+1)=3.76$GB (the extra one is used in FFT). From the report of machine, the memory we used is 3.84GB, which almost reach the top limit of 4 nodes (4GB). The 1024^3 simulation will need 32 nodes, and 2048^3 simulation must use all 256 nodes. Table 1 shows the computational time per step for those three situations. Theoretically, those three cases have the same load on each node. Furthermore, the computational time should be same. However, the communication is quite different, as more time is needed when grid size becomes larger. In our computation, the 2048^3 case only runs about 20 steps to obtain the results about performance. The other two cases, 512^3 and 1024^3 ones, run to the final convergent results with steady spectrums. Fig. 3 shows the development procedures of spectrums in

Table 1: Computation time per step with 6A method.

grid size	512^3	1024^3	2048^3
number of nodes	4	32	256
number of process	8	64	512
time (second)	164	180	283

512^3 and 1024^3 simulations. The spectrums at low wave number are almost the same. More high wave number energy is resolved in 1024^3 case. The final convergent spectrum is shown in Fig. 4. According to above memory estimation, we may solve 4096^3 simulation with a 1.6TB-memory machine with present method, while the original spectral method must use at least 3.16TB-memory machine. Upon using the Earth Simulator in computations [16, 18], the authors had to reduce partly the precision of arithmetic from double to single in order to allocate sufficient memory for their 4096^3 simulation. If the proposed scheme is used, that compromise will not be necessary.

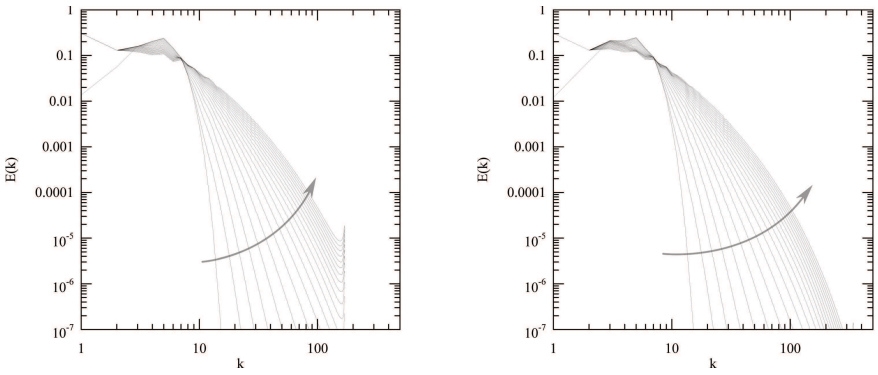

Figure 3: Spectral developments of 512^3 (left) and 1024^3 (right) simulations, the arrows show the direction of the spectrum developments.

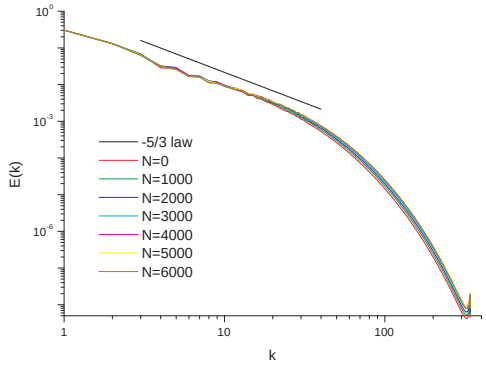

Figure 4: Spectral developments of 1024^3 simulation for the convergent result, the spectra reach a steady state after about 3000 steps. N is time step.

Table 2: Computation time per step on PKU-HP clusters with 512^3 mesh, 12.5A, 7A, and 6A methods are employed respectively.

	number of process	2×1	4×1	8×1	16×1	32×1	2×2	4×2	8×2	16×2
12.5A	time (second)	**302.8**	33.6	17.3	9.0	4.4	**292.0**	25.0	12.9	6.4
	Memory(MB)	3791	1900	972	529	320	1898×2	972×2	529×2	320×2
7A	time (second)	156.5	83.6	44.1	21.9	11.3	111.0	58.8	29.8	14.8
	Memory(MB)	2124	1065	554	321	215	1065×2	554×2	321×2	215×2
6A	time (second)	216.0	115.5	60.9	30.3	15.5	148.6	79.2	40.4	20.1
	Memory(MB)	1867	936	490	289	199	936×2	490×2	289×2	199×2

Table 3: Computation time per step on PKU-HP clusters with 1024^3 mesh, 12.5A, 7A, and 6A methods are employed respectively.

	number of process	16×1	32×1	64×1	16×2	32×2	64×2
12.5A	time (second)	81.6	40.2	20.1	**470.0**	28.4	13.7
	Memory(MB)	3669	1890	1022	1890×2	1022×2	692×2
7A	time (second)	201.5	102.2	51.3	134.0	67.2	32.4
	Memory(MB)	2004	1057	606	1057×2	606×2	484×2
6A	time (second)	280.2	141.2	71.7	182.8	92.2	44.3
	Memory(MB)	1747	928	541	928×2	541×2	442×2

We run our codes on PKU-HP clusters also, which have 128 nodes, each of which has two processors with 4G memory. The performances of three different methods, 12.5A, 7A, and 6A methods, are shown in Table 2 and Table 3 for 512^3 case and 1024^3 case respectively. 12.5A method is the original one [11], 7A and 6A methods are the new implementations of the original method, the first one has 7 N^3-arrays, the second has 6. Except for the three special cases (marked with boldface) for 12.5A method, the performance of parallel computations is very good, the time for every step is almost exactly bisected as the computational resource is doubled. Two bold cases in Table 2 show much longer time needed in computation with 12.5A method for 2×1 and 2×2 processors. 2×1 means that there are 2 nodes in the computation, and 1 process each node. 2×2 means there are 2 nodes in the computation, and 2 processes each node. From the reports about memory, we can find that the memory is almost reaching the top limit of 4G in each node. The usage of swap space makes the efficiency lower. For 1024^3 simulation, the same situation appears in 16×2 simulation. From Table 2 and Table 3, the required memory of the 6A method is only half of that of the 12.5A method, which permits us to simulate much larger cases with the same computer.

4 Conclusions and discussion

A new implementation algorithm of the pseudo-spectral method is proposed in this paper. The advantage of this approach lies in memory saving. In fact, by using second-order

Runge-Kutta time integration and careful programming, we found that only 6 arrays are needed for time evolution of the three-dimensional Navier-Stokes equations. We have coded the memory-saving pseudo-spectral method with six arrays while using Message Passing Interface and obtaining simulation results, which are identical to those from the standard 12.5-arrays code for system sizes of 512^3, 1024^3 and 2048^3. We have also simulated flows with 2048^3 mesh points using a machine with 256 GB memory, which normally have a memory requirement larger than 400 GB. Typically, the computational speed for the 6 array spectral code is 3-4 times slower than that for a 12.5 array calculation with the same computer resource. For this reason, we have developed different spectral codes, which can vary the number of arrays to cope with different memory and computational speed requirements.

Appendix

Detailed description of the programming

In our code, we use six N^3 arrays and two N^2 arrays as follows:

```
complex, allocatable, dimension (:,:,:)   ::  VX,VZ,VXB,VZB
complex, allocatable, dimension (:,:)     ::  VY0,VYB0
complex, allocatable, dimension (:,:,:)   ::  WO,VY
```

N^2 arrays are used to store velocity component $\hat{v}_y$ at $k_y=0$ plane. The memory of N^2 arrays can be neglected compared with N^3 ones. In the following statement, we use VX$\leftarrow v_x$ to show that velocity component v_x is saved in array VX. The computation procedures of the 6A method to advance $\hat{\mathbf{v}}^n$ to $\hat{\mathbf{v}}^{n+1}$ (which is the implementation of Eq. (2.13)) are:

---

1. Known $\hat{\mathbf{v}}^n$: VX$\leftarrow\hat{v}_x^n$, VY$\leftarrow\hat{v}_y^n$, VZ$\leftarrow\hat{v}_z^n$, and VY0$\leftarrow\hat{v}_y^n(k_x,0,k_z)$;
2. VXB$\leftarrow\hat{v}_x^n$, VZB$\leftarrow\hat{v}_z^n$, and an N^2 array VYB0 is used to store $\hat{v}_y(k_x,0,k_z)$, then implement the first step of Eq. (2.13) as:
   (a) Note that the initial values of velocity $\hat{\mathbf{v}}^n$ are stored in arrays VXB, VZB and VYB0. The whole velocity information is saved in VX, VY and VZ also;
   (b) Viscous terms in Eqs. (2.5) and (2.11): VXB$\leftarrow$ VXB $-(\nu k^2\Delta t/2)*$ VX, VZB$\leftarrow$ VZB $-(\nu k^2\Delta t/2)*$ VZ, VYB0$\leftarrow$ VYB0 $-(\nu k^2\Delta t/2)*$ VY0,
   (c) Convection terms in Eqs. (2.5) and (2.11):
      i. Terms which contain ω_x in Eqs. (2.8) and (2.12):
         A. Obtain $\hat{\omega}_x$ according to Eq. (2.10), WO $\leftarrow\hat{\omega}_x$;
         B. FFT$_1$: $\hat{\omega}_x\rightarrow\omega_x$, WO $\leftarrow\omega_x$;
         C. Compute the contributions $R_x=k_xk_z\widehat{v_y\omega_x}/k^2$, and $R_z=-(1-k_z^2/k^2)\widehat{v_y\omega_x}$ in Eq. (2.8)
            - FFT$_2$: $\hat{v}_y\rightarrow v_y$, VY $\leftarrow v_y$;

- Compute $v_y\omega_x \rightarrow$ VY, FFT_3: $v_y\omega_x \rightarrow \widehat{v_y\omega_x}$, VY $\leftarrow \widehat{v_y\omega_x}$;
- Update the right side hand term:

$$VXB \leftarrow VXB + \Big(\frac{1}{2}\Delta t k_x k_z\Big)\frac{\widehat{v_y\omega_x}}{k^2},$$
$$VZB \leftarrow VZB - \Big(\frac{1}{2}\Delta t\Big(1-\frac{k_z^2}{k^2}\Big)\Big)\widehat{v_y\omega_x},$$

D. Compute the contributions $R_x = -k_x k_y \widehat{v_z\omega_x}/k^2$, and $R_z = -k_y k_z \widehat{v_z\omega_x}/k^2$ in Eq. (2.8), and $R_y^0 = \widehat{v_z\omega_x}$ in Eq. (2.12)
- FFT_4: $\hat{v}_z \rightarrow v_z$, VZ $\leftarrow v_z$;
- Compute $v_z\omega_x \rightarrow$ VY, FFT_5: $v_z\omega_x \rightarrow \widehat{v_z\omega_x}$, VY $\leftarrow \widehat{v_z\omega_x}$;
- Update the right side hand term:

$$VXB \leftarrow VXB - \Big(\frac{1}{2}\Delta t k_x k_y\Big)\frac{\widehat{v_z\omega_x}}{k^2},$$
$$VZB \leftarrow VZB - \Big(\frac{1}{2}\Delta t k_y k_z\Big)\frac{\widehat{v_z\omega_x}}{k^2},$$
$$VYB0 \leftarrow VYB0 + \Big(\frac{1}{2}\Delta t\Big)\widehat{v_z\omega_x},$$

ii. Terms which contain ω_y in Eq. (2.8)

A. Transforming the velocity component v_z to spectral space, FFT_6: $v_z \rightarrow \hat{v}_z$, VZ $\leftarrow \hat{v}_z$;

B. Compute $\hat{v}_y$ according to Eq. (2.4), VY $\leftarrow \hat{v}_y$;

C. Obtain $\hat{\omega}_y$ according Eq. (2.10), WO $\leftarrow \hat{\omega}_y$;

D. FFT_7: $\hat{\omega}_y \rightarrow \omega_y$, WO $\leftarrow \omega_y$;

E. Compute the contributions $R_x = -(1-k_x^2/k^2)\widehat{v_z\omega_y}$, and $R_z = k_x k_z \widehat{v_z\omega_y}/k^2$ in Eq. (2.8)
- FFT_8: $\hat{v}_z \rightarrow v_z$, VZ $\leftarrow v_z$;
- Compute $v_z\omega_y \rightarrow$ VY, FFT_9: $v_z\omega_y \rightarrow \widehat{v_z\omega_y}$, VY $\leftarrow \widehat{v_z\omega_y}$;
- Update the right side hand term:

$$VXB \leftarrow VXB - \Big(\frac{1}{2}\Delta t\Big(1-\frac{k_x^2}{k^2}\Big)\Big)\widehat{v_z\omega_y},$$
$$VZB \leftarrow VZB + \Big(\frac{1}{2}\Delta t k_x k_z\Big)\frac{\widehat{v_z\omega_y}}{k^2},$$

F. Compute the contributions $R_x = -k_x k_z \widehat{v_x\omega_y}/k^2$, and $R_z = (1-k_z^2/k^2)\widehat{v_x\omega_y}$ in Eq. (2.8)
- FFT_{10}: $\hat{v}_x \rightarrow v_x$, VX $\leftarrow v_x$;
- Compute $v_x\omega_y \rightarrow$ VY, FFT_{11}: $v_x\omega_y \rightarrow \widehat{v_x\omega_y}$, VY $\leftarrow \widehat{v_x\omega_y}$;
- Update the right side hand term:

$$VXB \leftarrow VXB - \Big(\frac{1}{2}\Delta t k_x k_z\Big)\frac{\widehat{v_x\omega_y}}{k^2},$$
$$VZB \leftarrow VZB + \Big(\frac{1}{2}\Delta t\Big(1-\frac{k_z^2}{k^2}\Big)\Big)\widehat{v_x\omega_y},$$

iii. Terms which contain ω_z in Eq. (2.8)

A. Transform the velocity component v_x to spectral space, $\text{FFT}_{12}: v_x \to \hat{v}_x$, VX $\leftarrow \hat{v}_x$;

B. Transform the velocity component v_z to spectral space, $\text{FFT}_{13}: v_z \to \hat{v}_z$, VZ $\leftarrow \hat{v}_z$;

C. Compute $\hat{v}_y$ according to Eq. (2.4), VY $\leftarrow \hat{v}_y$;

D. Obtain $\hat{\omega}_z$ according Eq. (2.10), WO $\leftarrow \hat{\omega}_z$;

E. FFT_{14}: $\hat{\omega}_z \to \omega_z$, WO $\leftarrow \omega_z$;

F. Compute the contributions $R_x = (1-k_x^2/k^2)\widehat{v_y\omega_z}$, and $R_z = -k_xk_z\widehat{v_y\omega_z}/k^2$ in Eq. (2.8)

- FFT_{15}: $\hat{v}_y \to v_y$, VY $\leftarrow v_y$;
- Compute $v_y\omega_z \to$ VY, FFT_{16}: $v_y\omega_z \to \widehat{v_y\omega_z}$, VY $\leftarrow \widehat{v_y\omega_z}$;
- Update the right side hand term:

$$VXB \leftarrow VXB + \Big(\frac{1}{2}\Delta t\Big(1-\frac{k_x^2}{k^2}\Big)\Big)\widehat{v_y\omega_z},$$
$$VZB \leftarrow VZB - \Big(\frac{1}{2}\Delta t k_x k_z\Big)\frac{\widehat{v_y\omega_z}}{k^2},$$

G. Compute the contributions $R_x = k_xk_y\widehat{v_x\omega_z}/k^2$, $R_z = k_yk_z\widehat{v_x\omega_z}/k^2$ in Eq. (2.8), and $R_y^0 = -\widehat{v_x\omega_z}$ in Eq. (2.12)

- FFT_{17}: $\hat{v}_x \to v_x$, VX $\leftarrow v_x$;
- Compute $v_x\omega_z \to$ VY, FFT_{18}: $v_x\omega_z \to \widehat{v_x\omega_z}$, VY $\leftarrow \widehat{v_x\omega_z}$;
- Update the right side hand term:

$$VXB \leftarrow VXB + \Big(\frac{1}{2}\Delta t k_x k_y\Big)\frac{\widehat{v_x\omega_z}}{k^2},$$
$$VZB \leftarrow VZB + \Big(\frac{1}{2}\Delta t k_y k_z\Big)\frac{\widehat{v_x\omega_z}}{k^2},$$
$$VYB0 \leftarrow VYB0 - \Big(\frac{1}{2}\Delta t\Big)\widehat{v_x\omega_z},$$

(d) Restore velocity field: FFT_{19}: $v_x^n \to \hat{v}_x^n$, VX $\leftarrow \hat{v}_x^n$;

(e) Compute $\hat{v}_y^n$ according to Eq. (2.4), VY $\leftarrow \hat{v}_y^n$;

3. After first half step, we got $\hat{\mathbf{v}}^{1/2}$, VXB $\leftarrow \hat{v}_x^{1/2}$, VZB $\leftarrow \hat{v}_z^{1/2}$, VYB0 $\leftarrow \hat{v}_y^{1/2}$, and $\hat{\mathbf{v}}^n$ is still saved at VX $\leftarrow \hat{v}_x^n$, VZ $\leftarrow \hat{v}_z^n$, VY $\leftarrow \hat{v}_y^n$, VY0 $\leftarrow \hat{v}_y^n(k_x,0,k_z)$. The implementation of the second half step of Eq. (2.13) is almost the same with that of the first half step. The difference is that change $\Delta t/2$ to Δt, all quantities at n time level are substituted by that at $n+1/2$ level. And all update procedures should be done to VX, VZ and VY0 directly.

4. Eventually, we get velocity $\hat{\mathbf{v}}^{n+1}$: VX $\leftarrow \hat{v}_x^{n+1}$, VZ $\leftarrow \hat{v}_z^{n+1}$, VY $\leftarrow \hat{v}_y^{n+1}$, VY0 $\leftarrow \hat{v}_y^{n+1}(k_x,0,k_z)$.

There are 38 FFTs (19 in half step 2 and another 19 in step 3) in one time step. If we have one more N^3 array WT, before FFT_4, insert WT$\leftarrow \hat{v}_z$, then FFT_6 and FFT_8 are not in need

anymore. Before FFT_{10}, insert VZ$\leftarrow \hat{v}_z$ (the source code is "VZ=WT", to store $\hat{v}_z$ in array VZ) and WT$\leftarrow \hat{v}_x$, then FFT_{12}, FFT_{13}, FFT_{17} and FFT_{19} are all neglectable. Such that there are 26 FFTs in one time step for 7A method. In a practical simulation, we can use 6A or 7A method according to the memory of computer. Obviously, if the memory permits us to use more arrays, we can design more effective algorithms.

Acknowledgments

We appreciate Prof. Lin-Bo Zhang at LSSC for providing the super computer to us. Also we acknowledge CCSE of Peking University to provide HP clusters. Finally, we thank the support from National Natural Science Funds for Distinguished Young Scholar group under Grant No. 10921202 and National Climb Plan under Grant No. 2009CB724100.

References

[1] G. I. Taylor, Statistical theory of turbulence, Proc. Roy. Soc. London. A., 151 (1935), 421–444.
[2] A. N. Kolmogorov, The local structure of turbulence in incompressible viscous fluid for very large Reynolds number, Dokl. Acad. Nauk. SSSR., 30 (1941), 301–305. also: Turbulence and stochastic process: Kolmogorov's ideas 50 years on, Pro. Math. Phys. Sci., 434 (1991), 9–13.
[3] A. N. Kolmogorov, On degeneration of isotropic turbulence in an incompressible viscous liquid, Dokl. Acad. Nauk. SSSR., 31 (1941), 538–540.
[4] A. N. Kolmogorov, Dissipation of energy in locally isotropic turbulence, Dokl. Acad. Nauk. SSSR., 32 (1941), 16–19. also: Turbulence and stochastic process: Kolmogorov's ideas 50 years on, Pro. Math. Phys. Sci., 434 (1991), 15–17.
[5] S. A. Orszag, Numerical simulation of incompressible flows within simple boundaries: I. Galerkin (spetral) representations, Stu. Appl. Math., 50 (1971), 293–327.
[6] C. Canuto, M. Y. Hussaini, A. Quarteron, and T. A. Zang, Spectral Methods in Fluid Dynamics, Springer-Verlag, 1987.
[7] E. D. Siggia, Numerical study of small-scale intermittency in three-dimensional turbulence, J. Fluid. Mech., 107 (1981), 375–406.
[8] R. M. Kerr, Higher-order derivative correlations and the alignment of small-scale structures in isotropic numerical turbulence, J. Fluid. Mech., 153 (1985), 31–58.
[9] R. M. Kerr, Velocity, scalar and transfer spectra in numerical turbulence, J. Fluid. Mech., 211 (1990), 309–332.
[10] Z.-S. She, E. Jackson, and S. A. Orszag, Intermittent vortex structures in homogeneous isotropic turbulence, Nature., 344 (1990), 226–228.
[11] S. Chen, and X. Shan, High-resolution turbulent simulations using the connection machine-2, Comput. Phys., 6 (1992), 643–646.
[12] S. Chen, G. D. Doolen, R. H. Kraichnan, and Z. S. She, On statistical correlations between velocity increments and locally averaged dissipation in homogeneous turbulence, Phys. Fluids. A., 5 (1993), 458–463.
[13] L. P. Wang, S. Chen, J. G. Brasseur, and J. C. Wyngaard, Examination of hypotheses in the Kolomogorov refined turbulence theory through high-resolution simulations, part 1. velocity field, J. Fluid. Mech., 309 (1996), 113–156.

[14] P. K. Yeung, and Y. Zhou, Universality of the Kolmogorov constant in numerical simulations of turbulence, Phys. Rev. E., 56 (1997), 1746–1752.
[15] P. K. Yeung, and Y. Zhou, Numerical study of rotating turbulence with external forcing, Phys. Fluids., 10 (1998), 2895–2909.
[16] 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, in: Conference on High Performance Networking and Computing, Proceedings of the 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland, 2002, 1–17.
[17] T. Ishihara, K. Yoshida, and Y. Kaneda, Anisotropic velocity correlation spectrum at small scales in a homogeneous turbulent shear flow, Phys. Rev. Lett., 88 (2002), 154501.
[18] Y. Kaneda, T. Ishihara, M. Yokokawa, K. Itakura, and A. Uno, Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box, Phys. Fluids., 15 (2003), L21–L24.
[19] T. Gotoh, D. Fukayama, and T. Nakano, Velocity field statistics in homogeneous steady turbulence obtained using a high-resolution direct numerical simulation, Phys. Fluids., 14 (2002), 1065–1081.
[20] Y. Kaneda, and T. Ishihara, High-resolution direct numerical simulation of turbulence, J. Turbul., 7 (2006), N20.