An Asymptotic Parallel Linear Solver and Its Application to Direct Numerical Simulation for Compressible Turbulence

Mitsuo Yokokawa[1][0000−0003−3790−1243], Taiki Matsumoto[1], Ryo Takegami[1], Yukiya Sugiura[2], Naoki Watanabe[3], Yoshiki Sakurai[4][0009−0009−0575−0904], Takashi Ishihara[5][0000−0002−4520−6964], Kazuhiko Komatsu[6][0000−0003−4463−8359], and Hiroaki Kobayashi[7][0000−0002−3350−1413]

[1] Graduate School of System Informatics, Kobe University, Kobe, Japan
yokokawa@port.kobe-u.ac.jp
[2] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[3] Mizuho Research & Technologies, Ltd., Chiyoda-ku, Tokyo, Japan
[4] Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Japan
[5] Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama, Japan
[6] Cyberscience Center, Tohoku University, Sendai, Japan
[7] Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Abstract. When solving numerically partial differential equations such as the Navier-Stokes equations, higher-order finite difference schemes are occasionally applied for spacial descretization. Compact finite difference schemes are one of the finite difference schemes and can be used to compute the first-order derivative values with smaller number of stencil grid points, however, a linear system of equations with a tridiagonal or pentadiagonal matrix derived from the schemes have to be solved. In this paper, an asymptotic parallel solver for a reduce matrix, that obtained from the Mattor's method in a computation of the first-order derivatives with an eighth-order compact difference scheme under a periodic boundary condition, is proposed. The asymptotic solver can be applied as long as the number of grid points of each Cartesian coordinate in the parallelized subdomain is 64 or more, and its computational cost is lower than that of the Mattor's method. A direct numerical simulation code has also been developed using the two solvers for compressible turbulent flows under isothermal conditions, and optimized on the vector supercomputer SX-Aurora TSUBASA. The optimized code is 1.7 times faster than the original one for a DNS with 2048^3 grid points and the asymptotic solver achieves approximately a 4-fold speedup compared to the Mattor's solver. The code exhibits excellent weak scalability.

Keywords: Asymptotic parallel linear solver · Eighth-order compact difference scheme · Direct numerical simulation · Finite difference method · Vector system · SX-Aurora TSUBASA

1 Introduction

Turbulence is a core physical phenomenon in various natural phenomena and problems in science and technology, and it is important to understand and elucidate its universal nature. However, the Navier-Stokes equations, which are the governing equations for fluid flows, are known to be highly nonlinear and difficult to solve analytically. Therefore, it is effective to study turbulent flows by numerical simulations using supercomputers. Among numerical simulations of turbulent flows, a direct numerical simulation (DNS) is widely used for clarifying the universal properties of turbulence because it solves the governing equations directly without modeling and resolves eddy motions at the smallest scale[2].

Large-scale DNSs for incompressible turbulence using a Fourier spectral method with up to 12288^3 grid points have been performed to study the statistical properties in the inertial subrange of turbulence at the Taylor-microscale Reynolds number up to 2300 [3]. For compressible turbulence, on the other hand, such large-scale DNSs have not yet been performed, and the statistical properties in the inertial subrange of compressible turbulence at high Reynolds numbers have not yet been fully studied. Recently, to the best of our knowledge, the largest DNSs for compressible isothermal turbulence using finite difference methods up to 4096^3 grid points have been performed[8]. However, the Reynolds numbers achieved by these DNSs are not high enough to study inertial subrange properties. To obtain high-resolution results in DNSs for compressible turbulence using finite difference methods requires a large number of grid points and a large amount of computational resources[8]. Compact finite difference schemes are often used to discretize the convection terms in the Navier-Stokes equations. Though a high resolution solutions can be obtained compared to usual finite difference schemes, linear systems have to be solved. Therefore, it is necessary to develop a fast solver of the systems and a parallel DNS code using the solvers.

In parallel computations, a computation domain is divided into several subdomains, calculations of which are assigned to parallel tasks and executed. In this case, the number of parallel tasks and the method of dividing the computational domain, which can change the amount of computation and communication in each task, must be chosen appropriately to obtain results fast. In general, when a sufficiently large number of grid points is taken so as to calculate the flow fields precisely, a three-dimensional domain decomposition (cuboid decomposition) can increase the amount of parallelism compared to a two-dimensional domain decomposition (pencil decomposition), and large-scale computation with massively parallel supercomputers is expected.

In this paper, we propose an asymptotic parallel linear solver for the linear system of equations obtained when first-order derivatives in space is discretized by an eighth-order compact difference method. The solver is then used to develop a finite difference DNS code of three-dimensional homogeneous isotropic compressible isothermal turbulence in a box with a periodic boundary condition. The code is parallelized by the cuboid decomposition to increase the number of parallel tasks. The code is optimized and its performance is evaluated on the supercomputer SX-Aurora TSUBASA installed at Cyberscience Center, Tohoku University.

The remainder of this paper is organized as follows. Section 2 outlines a direct numerical simulation code for compressible turbulent flows under isothermal conditions. Section 3 describes parallel solutions of the linear systems of equations obtained by applying an eighth-order compact scheme to the first-order derivative calculations. Section 4 gives optimization and performance evaluation results of the code on SX-Aurora TSUBASA. Section 5 concludes the paper.

2 Direct Numerical Simulation Code

We consider the three-dimensional compressible turbulent flows with isothermal conditions in a cube with a side length of 2π subject to periodic boundary conditions that obey the following equations;

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{1}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u} + p\mathbf{I}) = \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F}, \tag{2}$$

$$p = \rho c^2, \tag{3}$$

where t is the time, ρ the density, $\boldsymbol{u}$ the velocity, p the pressure, $\boldsymbol{\tau}$ the viscous stress tensor, and c the speed of sound. $\boldsymbol{F}$ is the forcing terms to maintain the flows in statistically quasi-steady states. $\mathbf{I}$ is the identity matrix.

Here, Equations (1) and (2) are discretized at gird points that equally divide the computational region (the cube), those are $(x_i, y_j, z_k) = (i\varDelta, j\varDelta, k\varDelta)$ $(0 \le i, j, k \le N)$ and $\varDelta = 2\pi/N$, where N is the number of division in each Cartesian coordinate. Then, N^3 ordinary differential equations with respect to $\rho_{i,j,k}$ and $(\rho\boldsymbol{u})_{i,j,k}$ are obtained, where $\rho_{i,j,k} = \rho(x_i, y_j, z_k)$ and $(\rho\boldsymbol{u})_{i,j,k} = \rho(x_i, y_j, z_k)\boldsymbol{u}(x_i, y_j, z_k)$. In disretizing the equations in space, an eighth-order compact difference scheme (CD8)[5] for the convection terms and an eighth-order central finite difference scheme (FD8) for the viscous terms are used. This is because the amount of contribution from the viscous terms is small for high Reynolds number flows, and a higher precision scheme is not necessary for the viscous terms. Describing a physical value $f(x, y, z)$ and the first-order partial derivative in the x-direction $\frac{\partial f(x,y,z)}{\partial x}$ at the point (x_i, y_j, z_k) by $f_{i,j,k}$ and $f'_{i,j,k}$, respectively, the CD8 and FD8 schemes are written as follows,

$$\begin{aligned} &\alpha f'_{i-1,j,k} + f'_{i,j,k} + \alpha f'_{i+1,j,k} = \\ &\qquad a\frac{f_{i+1,j,k} - f_{i-1,j,k}}{2\varDelta} + b\frac{f_{i+2,j,k} - f_{i-2,j,k}}{4\varDelta} + c\frac{f_{i+3,j,k} - f_{i-3,j,k}}{6\varDelta}, \\ &\qquad\qquad \alpha = \frac{3}{8}, \quad a = \frac{25}{16}, \quad b = \frac{1}{5}, \text{ and } c = -\frac{1}{80}, \end{aligned} \tag{4}$$

and

$$f'_{i,j,k} = a\frac{f_{i+1,j,k} - f_{i-1,j,k}}{\Delta} + b\frac{f_{i+2,j,k} - f_{i-2,j,k}}{\Delta} + c\frac{f_{i+3,j,k} - f_{i-3,j,k}}{\Delta} + d\frac{f_{i+4,j,k} - f_{i-4,j,k}}{\Delta},$$
$$a = \frac{4}{5},\ b = -\frac{1}{5},\ c = \frac{4}{105},\ \text{and } d = -\frac{1}{280}. \quad (5)$$

The first-order partial derivative in each y- and z- direction is also written by the similar formula. To maintain a statistically quasi-steady state of turbulence, the same forcing terms as Petersen and Livescu [7] are incorporated. That is, the Fourier coefficients of the external force $\boldsymbol{F}$ is not zero only at low wavenumbers $|\boldsymbol{k}| = \sqrt{k_1^2 + k_2^2 + k_3^2} < 3$, where k_1, k_2, and k_3 are wavenumbers corresponding to each direction of the Cartesian coordinates. Since the number of non-zero Fourier coefficients is at most 100 out of N^3, not fast Fourier transforms but discrete Fourier transforms (DFT) are used to calculate them to decrease computational costs.

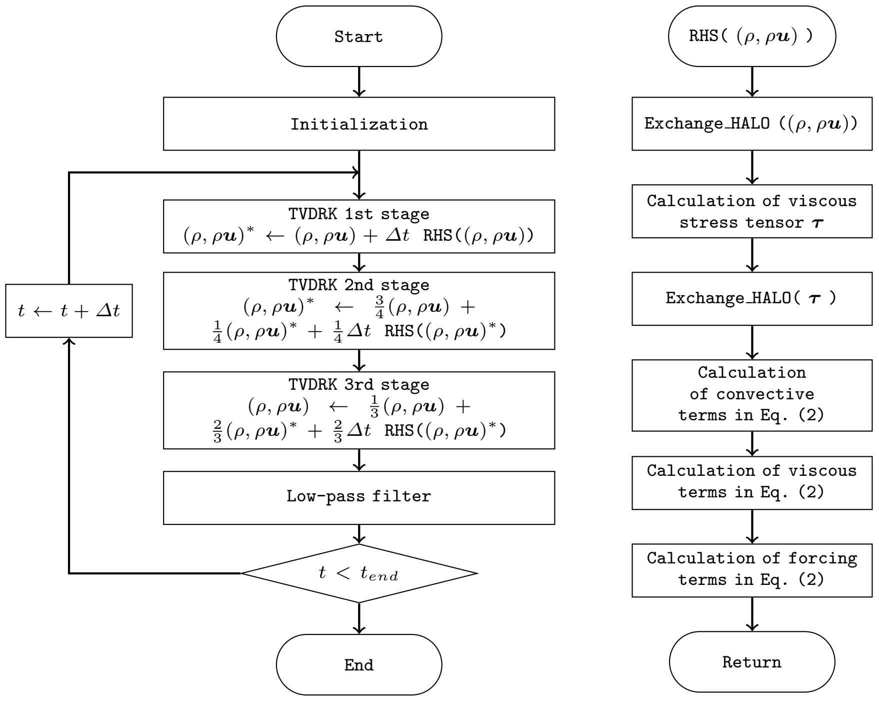

Start
Initialization
TVDRK 1st stage
$(\rho, \rho\boldsymbol{u})^* \leftarrow (\rho, \rho\boldsymbol{u}) + \Delta t$ RHS($(\rho, \rho\boldsymbol{u})$)
TVDRK 2nd stage
$(\rho, \rho\boldsymbol{u})^* \leftarrow \frac{3}{4}(\rho, \rho\boldsymbol{u}) + \frac{1}{4}(\rho, \rho\boldsymbol{u})^* + \frac{1}{4}\Delta t$ RHS($(\rho, \rho\boldsymbol{u})^*$)
TVDRK 3rd stage
$(\rho, \rho\boldsymbol{u}) \leftarrow \frac{1}{3}(\rho, \rho\boldsymbol{u}) + \frac{2}{3}(\rho, \rho\boldsymbol{u})^* + \frac{2}{3}\Delta t$ RHS($(\rho, \rho\boldsymbol{u})^*$)
Low-pass filter
$t < t_{end}$
End
$t \leftarrow t + \Delta t$
RHS($(\rho, \rho\boldsymbol{u})$)
Exchange_HALO ($(\rho, \rho\boldsymbol{u})$)
Calculation of viscous stress tensor $\boldsymbol{\tau}$
Exchange_HALO($\boldsymbol{\tau}$)
Calculation of convective terms in Eq. (2)
Calculation of viscous terms in Eq. (2)
Calculation of forcing terms in Eq. (2)
Return

Fig. 1: DNS Code Flow

The third-order total variation diminishing Runge-Kutta (TVDRK) scheme[1] with a constant time step is used for a temporal integration of the equations of

$\rho_{i,j,k}$ and $(\rho\boldsymbol{u})_{i,j,k}$ over time. Additionally, an eighth-order low-pass filter scheme is applied to remove unphysical numerical oscillations at high wavenumbers. The code flow is illustrated in Fig. 1.

A uniform grid of dimensions $N \times N \times N$ discretizing the cube is partitioned into some sets of grid points using a standard cuboid decomposition and those are assigned to parallel tasks with layout $n_{px} \times n_{py} \times n_{pz}$. The number of grid points in a subregion assigned to a task is $(N/n_{px}) \times (N/n_{py}) \times (N/n_{pz})$. Halo regions are added around each subregion to perform the CD8 and FD8 calculations within a task. The number of grids of halo regions that are extended outward in the direction of each coordinate of the calculation region is 4 so that the computation of FD8 is possible. Physical values at the grid points in the halo regions between adjacent tasks are exchanged in the procedure `Exchange_HALO` in the code flow, and then the CD8 and FD8 values can be calculated using Eqs. (4) and (5) independent of other tasks. To calculate the viscous terms in Eq. (2), each element of the viscous stress tensor τ is first calculated using the FD8, and then the derivative of each element is computed by the FD8 again.

3 Calculation of the System Obtained by the Eighth-order Compact Scheme

3.1 Linear System Obtained by the Eighth-order Compact Scheme

We consider here a solution for $N \times N$ linear systems of equations to calculate the first partial derivatives in the x-direction written as a matrix-vector form of Eq. (4) below:

$$\begin{bmatrix} 1 & \alpha & & \cdots & \alpha \\ \alpha & 1 & \alpha & & \\ & \alpha & 1 & \alpha & \vdots \\ & & \ddots & \ddots & \ddots \\ \vdots & & \alpha & 1 & \alpha \\ \alpha & \cdots\cdots & & \alpha & 1 \end{bmatrix} \begin{bmatrix} f'_{1,j,k} \\ f'_{2,j,k} \\ \vdots \\ \vdots \\ f'_{N-1,j,k} \\ f'_{N,j,k} \end{bmatrix} = \begin{bmatrix} \varDelta_x f_{1,j,k} \\ \varDelta_x f_{2,j,k} \\ \vdots \\ \vdots \\ \varDelta_x f_{N-1,j,k} \\ \varDelta_x f_{N,j,k} \end{bmatrix} \qquad (1 \le j,k \le N), \tag{6}$$

where

$$\varDelta_x f_{i,j,k} = a\frac{f_{i+1,j,k} - f_{i-1,j,k}}{2\varDelta} + b\frac{f_{i+2,j,k} - f_{i-2,j,k}}{4\varDelta} + c\frac{f_{i+3,j,k} - f_{i-3,j,k}}{6\varDelta}.$$

The matrix is a slightly perturbed form of the tridiagonal matrix with α added to the upper-right and lower-left elements due to the periodic boundary conditions. Similar linear systems for each y- and z-direction are obtained.

3.2 Parallel Solution by the Mattor Method

To simplify the computational procedure, we denote the linear system as $A\boldsymbol{x} = \boldsymbol{b}$ as follows,

$$A = \begin{bmatrix} 1 & \alpha & & \cdots & & \alpha \\ \alpha & 1 & \alpha & & & \vdots \\ & \alpha & 1 & \alpha & & \vdots \\ & & \ddots & \ddots & \ddots & \\ \vdots & & & \alpha & 1 & \alpha \\ \alpha & \cdots\cdots & & & \alpha & 1 \end{bmatrix} \in R^{N\times N}, \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} \in R^N, \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{N-1} \\ b_N \end{bmatrix} \in R^N. \tag{7}$$

Consider the system $A\boldsymbol{x} = \boldsymbol{b}$ is solved by P parallel tasks, where $N = MP$ and M is the number of grid points to be solved in a task. The parallel solver of a tridiagonal matrix system proposed by Mattor et al. [6] is applied to the linear system (7). The matrix is divided into $P \times P$ blocks and the vectors are divided into P subvectors as

$$\begin{bmatrix} A_M & \alpha\boldsymbol{e}_M\boldsymbol{e}_1^T & & \cdots & & \alpha\boldsymbol{e}_1\boldsymbol{e}_M^T \\ \alpha\boldsymbol{e}_1\boldsymbol{e}_M^T & A_M & \alpha\boldsymbol{e}_M\boldsymbol{e}_1^T & & & \vdots \\ & \alpha\boldsymbol{e}_1\boldsymbol{e}_M^T & A_M & \alpha\boldsymbol{e}_M\boldsymbol{e}_1^T & & \vdots \\ & & \ddots & \ddots & \ddots & \\ \vdots & & & \alpha\boldsymbol{e}_1\boldsymbol{e}_M^T & A_M & \alpha\boldsymbol{e}_M\boldsymbol{e}_1^T \\ \alpha\boldsymbol{e}_M\boldsymbol{e}_1^T & \cdots & \cdots & & \alpha\boldsymbol{e}_1\boldsymbol{e}_M^T & A_M \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \vdots \\ \vdots \\ \boldsymbol{x}_{P-1} \\ \boldsymbol{x}_P \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \vdots \\ \vdots \\ \boldsymbol{b}_{P-1} \\ \boldsymbol{b}_P \end{bmatrix}, \tag{8}$$

where

$$A_M = \begin{bmatrix} 1 & \alpha & & & & \\ \alpha & 1 & \alpha & & \Large 0 & \\ & \alpha & 1 & \alpha & & \\ & & \ddots & \ddots & \ddots & \\ \Large 0 & & & \alpha & 1 & \alpha \\ & & & & \alpha & 1 \end{bmatrix} \in R^{M\times M}, \boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \in R^M, \boldsymbol{e}_M = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in R^M, \tag{9}$$

$$\boldsymbol{x}_p \text{ and } \boldsymbol{b}_p \in R^M \quad (p = 1, \cdots, P).$$

$\boldsymbol{x}^T$ stands for a transpose vector of the vector $\boldsymbol{x}$. The p-th block system is written as

$$\alpha\boldsymbol{e}_1\boldsymbol{e}_M^T\boldsymbol{x}_{p-1} + A_M\boldsymbol{x}_p + \alpha\boldsymbol{e}_M\boldsymbol{e}_1^T\boldsymbol{x}_{p+1} = \boldsymbol{b}_p. \tag{10}$$

Following the Mattor method, $\boldsymbol{x}_p$ is written by three vectors $\boldsymbol{x}_p^R$, $\boldsymbol{x}_p^U$ and $\boldsymbol{x}_p^L$, and two scalars s_p and t_p as

$$\boldsymbol{x}_p = \boldsymbol{x}_p^R - t_p\boldsymbol{x}_p^U - s_p\boldsymbol{x}_p^L \quad (p = 1, \cdots, P), \tag{11}$$

$$A_M\boldsymbol{x}_p^R = \boldsymbol{b}_p, \quad A_M\boldsymbol{x}_p^U = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \alpha \end{bmatrix}, \quad \text{and } A_M\boldsymbol{x}_p^L = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{12}$$

Note that $\boldsymbol{x}_p^U = F\boldsymbol{x}_p^L$ with a flip matrix F, i.e. $x_{p,j}^U = x_{p,M-j+1}^L (j = 1, \cdots, M)$, where $x_{p,j}$ stands for the j-th element of vector $\boldsymbol{x}_p$. By substituting $\boldsymbol{x}_p$ in Eq. (10) by Eq. (11) and considering Eq. (12), $2P$ equations below are obtained.

$$\begin{aligned} (x_{p-1,M}^R - t_{p-1}x_{p-1,M}^U - s_{p-1}x_{p-1,M}^L) - s_p = 0, \\ (x_{p+1,1}^R - t_{p+1}x_{p+1,1}^U - s_{p+1}x_{p+1,1}^L) - t_p = 0 \quad (p = 1, \cdots P), \end{aligned} \tag{13}$$

where $x_{0,M}^R = x_{P,M}^R$, $x_{P+1,M}^R = x_{1,M}^R$, and so forth in the first and the P-th block system. Since $x_{p,j}^U = x_{p,M-j+1}^L$, we obtain a following reduced system (14) whose matrix size is $2P \times 2P$, and solve this system to get the coefficients s_p and t_p $(p = 1, \cdots, P)$.

$$\begin{bmatrix} x_{1,M}^U & x_{1,1}^U & & & \cdots & & & 1 \\ x_{1,1}^U & x_{1,M}^U & 1 & & & & & \vdots \\ & 1 & x_{2,M}^U & x_{2,1}^U & & & & \\ & & x_{2,1}^U & x_{2,M}^U & & & & \\ & & & \ddots & \ddots & \ddots & & \\ \vdots & & & & \ddots & \ddots & \ddots & \\ & & & & & 1 & x_{P,M}^U & x_{P,1}^U \\ 1 & & \cdots & & & & x_{P,1}^U & x_{P,M}^U \end{bmatrix} \begin{bmatrix} s_1 \\ t_1 \\ s_2 \\ t_2 \\ \vdots \\ \vdots \\ s_P \\ t_P \end{bmatrix} = \begin{bmatrix} x_{1,1}^R \\ x_{1,M}^R \\ x_{2,1}^R \\ x_{2,M}^R \\ \vdots \\ \vdots \\ x_{P,1}^R \\ x_{P,M}^R \end{bmatrix}. \tag{14}$$

The final solution is computed by Eq. (11). Note that the reduced matrix is organized by the first and M-th elements of the vector $\boldsymbol{x}_p^U$ $(p = 1, \cdots, P)$ and the reduced matrix can be assembled once at the initialization phase of the code by solving the second linear system in Eq. (12). Hereafter, the procedure is referred to as a normal solver and is shown in **Algorithm 1**.

3.3 Asymptotic Property of the Reduced Matrix

Here, let us derive the asymptotic property of the system $A_M\boldsymbol{x}_p^U = \alpha\boldsymbol{e}_M^T$, *i.e.*

Algorithm 1 (Normal Solver)

```
Initialization
1: Compute the Cholesky decomposition of the matrix $A_M$ in all tasks.
2: Solve the equation $A_M \boldsymbol{x}_p^U = [0, \cdots, 0, \alpha]^T$ and assemble the reduced matrix in all
   tasks concurrently.
In time advancement loop
1: Solve the equation $A_M \boldsymbol{x}_p^R = \boldsymbol{b}_p$ in each task after computing the vector $\boldsymbol{b}_p$.
2: Assemble the right hand side of the reduced system (14) by using MPI_Allgather
   function in all tasks.
3: Solve the system in each task in parallel to compute the coefficients $s_p$ and $t_p$.
4: Compute the final solution by Eq. (11) in each task.
```

$$\begin{bmatrix} 1 & \alpha & & & & \\ \alpha & 1 & \alpha & & \text{\Large 0} & \\ & \alpha & 1 & \alpha & & \\ & & \ddots & \ddots & \ddots & \\ & \text{\Large 0} & & \alpha & 1 & \alpha \\ & & & & \alpha & 1 \end{bmatrix} \begin{bmatrix} x_{p,1}^U \\ x_{p,2}^U \\ x_{p,3}^U \\ \vdots \\ \vdots \\ x_{p,M}^U \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \alpha \end{bmatrix}. \tag{15}$$

Applying the Gaussian elimination to the system, we can rewrite the system with a coefficient matrix having diagonal and super-diagonal elements, that is

$$\begin{bmatrix} \beta_1 & \alpha & & & & \\ & \beta_2 & \alpha & & \text{\Large 0} & \\ & & \beta_3 & \alpha & & \\ & & & \ddots & \ddots & \\ & \text{\Large 0} & & & \beta_{M-1} & \alpha \\ & & & & & \beta_M \end{bmatrix} \begin{bmatrix} x_{p,1}^U \\ x_{p,2}^U \\ x_{p,3}^U \\ \vdots \\ \vdots \\ x_{p,M}^U \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \alpha \end{bmatrix}, \tag{16}$$

where $\beta_1 = 1$, $\beta_j = 1 - \frac{\alpha^2}{\beta_{j-1}}$ $(j = 2, \cdots, M)$. The system can be easily solved by backward substitution and the solution is

$$\begin{aligned} x_{p,M}^U &= \frac{\alpha}{\beta_M}, \\ x_{p,j}^U &= -\frac{\alpha \cdot x_{p,j+1}^U}{\beta_j} = \frac{(-\alpha)^{M-j+1}}{\prod_{k=j}^{M} \beta_k} \quad (j = M-1, \cdots, 1). \end{aligned} \tag{17}$$

Since $\lim_{M\to\infty} \beta_M = \frac{1+\sqrt{1-4\alpha^2}}{2}$, $x_{p,M}^U = \frac{1-\sqrt{1-4\alpha^2}}{2\alpha}$ and $x_{p,1}^U = 0$ $(p = 1, \cdots, P)$ as M reaches infinity. Since $\alpha = 3/8$, $x_{p,M}^U$ and $x_{p,1}^U$ converge rapidly as shown in Table 1.

Table 1: Valuses of $x^U_{p,M}$ and $x^U_{p,1}$ when increasing M

M	$x^U_{p,M}$	$x^U_{p,1}$
8	0.451415160966647	$-1.372943122571951 \times 10^{-3}$
16	0.451416229641959	$-2.367389055633383 \times 10^{-6}$
32	0.451416229645136	$-7.038892637296893 \times 10^{-12}$
64	0.451416229645136	$-6.222626756534802 \times 10^{-23}$
128	0.451416229645136	$-4.863092990426517 \times 10^{-45}$

If the number of grid points M in the x-direction of the subregion is taken appropriately, at least $M \geq 64$, the reduced system (14) can be approximately expressed with the asymptotic values as follows;

$$\begin{bmatrix} d & 0 & & & & \cdots & & 1 \\ 0 & d & 1 & & & & & \\ & 1 & d & 0 & & & & \\ & & 0 & d & & & & \\ \vdots & & & & \ddots & \ddots & \ddots & \\ & & & & & 1 & d & 0 \\ 1 & \cdots & \cdots & & & & 0 & d \end{bmatrix} \begin{bmatrix} s_1 \\ t_1 \\ s_2 \\ t_2 \\ \vdots \\ s_P \\ t_P \end{bmatrix} = \begin{bmatrix} x^R_{1,1} \\ x^R_{1,M} \\ x^R_{2,1} \\ x^R_{2,M} \\ \vdots \\ x^R_{P,1} \\ x^R_{P,M} \end{bmatrix}, \tag{18}$$

where $d = x^U_{p,M} = \frac{1-\sqrt{1-4\alpha^2}}{2\alpha}$.

The coefficients s_p and t_p $(p = 1, \cdots, P)$ can be computed in each task independently of the others, using Eqs. (19) by transferring $x^R_{p+1,1}$ and $x^R_{p-1,M}$ from adjacent tasks. The final solution is computed by Eq. (11). Hereafter, the procedure is referred to as an asymptotic solver and is shown in **Algorithm 2**.

$$\begin{cases} t_p &= (d \cdot x^R_{p,M} - x^R_{p+1,1})/(d^2 - 1) \\ s_p &= (x^R_{p-1,M} - d \cdot x^R_{p,1})/(d^2 - 1) \end{cases} \tag{19}$$

Algorithm 2 (Asymptotic Solver)

```
Initialization
1: Compute the Cholesky decomposition of the matrix $A_M$ in all tasks.
2: Solve the equation $A_M \boldsymbol{x}^U_p = [0, \cdots, 0, \alpha]^T$ in all tasks.
In time advancement loop
1: Solve the equation $A_M \boldsymbol{x}^R_p = \boldsymbol{b}_p$ in each task after computing the vector $\boldsymbol{b}_p$.
2: Send the values $x^R_{p,1}$ and $x^R_{p,M}$ to the $(p-1)$ and $(p+1)$ task, respectively, by using
   MPI_Sendrecv function.
3: Compute the coefficients $t_p$ and $s_p$ by the equations (19).
4: Compute the final solution by Eq. (11) in each task.
```

4 Performance Evaluation of the Code on SX-Aurora TSUBASA

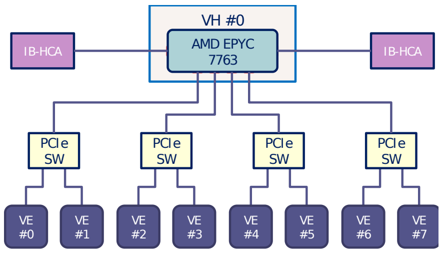
We have first built a parallel DNS code for compressible isothermal turbulence by the cuboid decomposition, as shown in Fig. 1, without considering optimization. Both the normal and asymptotic solvers are implemented in the code. When the first derivatives in a direction of the Cartesian coordinates are computed, the normal solver is used if the number of grid points in its direction is less than 64. Otherwise, the asymptotic solver is chosen. We confirmed that the computational results by the two solvers is the same as the results by the code used in [8].

In this section, an optimization of the first-created code on SX-Aurora TSUBASA is stated. Performance comparison of the two solvers and weak scaling characteristics of the code are also described.

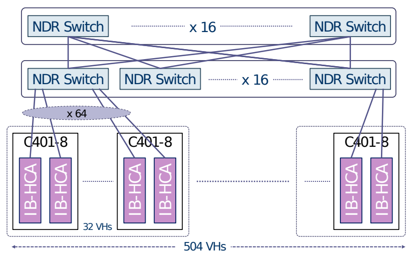
4.1 Measurement Environment

A vector-type supercomputer AOBA-S installed at Cyberscience Center, Tohoku University is used for optimization and performance evaluation of the code [4, 9].

AOBA-S consists of 504 compute nodes that are connected by a two-layer fat tree non-blocking network configured by the 32 Infiniband NDR switches. Each node, SX-Aurora TSUBASA C401-8, is configured by a vector host (VH) and eight vector engines (VEs) that are connected by four Peripheral Component Interconnect express (PCIe) Gen4 switches (Fig. 2a). The VH is a standard x86 server with an AMD EPCY 7736 processor running a standard Linux operating system, and has two InfiniBand NDR200 Host Channel Adapters connecting to the fat tree network (Fig. 2b). A VE is a vector accelerator implemented as a PCIe card, on which a vector engine Type 30A (VE30) and six extended high-bandwidth memory (HBM2E) modules, 96GB in total, are mounted. The VE30 processor integrates 16 vector cores, a 64MB shared last level cache (LLC), and a VE direct memory access (DMA) engine that are interconnected through a two-dimensional network (called Network on Chip) with a total bandwidth of



(a) C401-8 Configuration



(b) Interconnect Configuration

Fig. 2: AOBA-S Configuration

3.0 TB/s. Peak performance of the core is 307.2 Gflop/s in double precision floating-point operations, and therefore 4.92 Tflop/s in total for a VE30.

Operating system functions have been run on the VH to the greatest extent possible so that the VEs are able to effectively use their computational power by concentrating on program executions with vector instructions. The GNU C library (*glibc*) is ported to the VEs and applications can call *glibc* functions as normal I/O functions[10]. Vectorization should also be applied to achieve high computation performance.

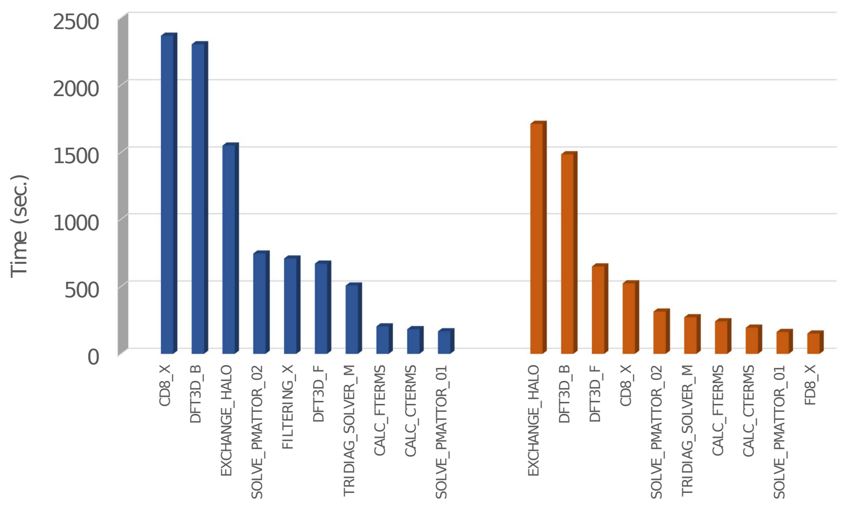


Fig. 3: Top 10 procedures of the code analyzed by Ftrace

4.2 Cost Distribution and Optimization

Computational cost distribution of the code is analyzed by using a performance analyzer **Ftrace** equipped on the SX-Aurora TSUBASA. The left bar graph in Fig. 3 shows the first 10 high-cost parts of the code, in the case that a DNS with the grid points $N^3 = 2048^3$ was performed by 512 MPI processes with an $8 \times 8 \times 8$ layout on 32 VEs (512 cores). Since the number of grid points in each direction assigned to a process is 256, the asymptotic parallel solver is used.

The first part `CD8_X` is the first-order derivative calculation in the x-direction. In the right-hand-side calculation of Eq. (6), since the coefficient matrix is the same for all gird points `(j,k)` of the y-z cross-section of an assigned subregion to an MPI process, all right-hand-side vectors corresponding to the grid points `(j,k)` are computed as follows:

```
for all grid points (j,k) of the y-z cross-section
  for i = is, ie
    b(jk,i) = a1*( f(i+1,j,k) - f(i-1,j,k) )
               + b1*( f(i+2,j,k) - f(i-2,j,k) )
               + c1*( f(i+3,j,k) - f(i-3,j,k) )
```

where `a1`, `b1`, and `c1` correspond to the coefficients in Eq. (4) and `is` and `ie` are the start and end grid numbers of the x-direction in each subregion. It is clear that the right-hand-side array "`f`" accesses memory addresses at equal intervals.

The second part `DFT3D_B` is the backward 3D-DFT used to calculate the Fourier coefficients of the low wavenumbers side applying external force. The third part `EXCHANGE_HALO` is a halo region exchange procedure by the `MPI_Sendrecv` function. The fourth part `SOLVE_PMATTOR_02` is the asymptotic solver. The fifth part `FILTERING_X` is the low-pass filter procedure and has almost the similar memory access pattern to the `CD8_X`.

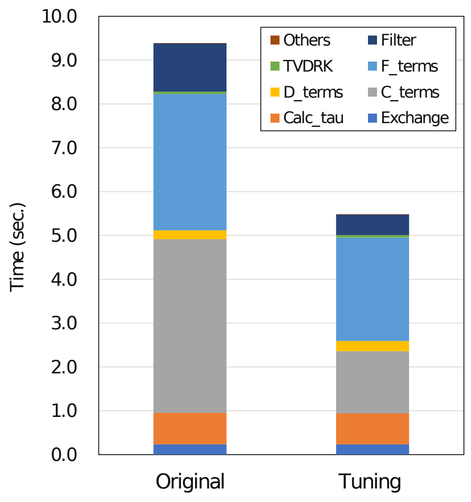


Fig. 4: Comparison of Calculation time between original and tuning codes

An loop order of indexes is changed to obtain the highest performance on the SX-Aurora TSUBASA, and it is found that the order of `k, i`, and `j` from the outer loop is the best. The right bar chart in Fig. 3 is the top 10 cost-consuming parts after the loop order exchanges, in that the `CD8_X` is in the fourth and 4.5 times faster than the original.

Figure 4 shows computational time for both the original and tuning codes. We confirmed that the results of the two DNS codes are identical. The times of the parts `C_terms` and `Filter` of the tuning code, in which the `CD8_X` or similar is used, are found to be shorten compared to the original one, resulting in 1.7 times faster.

Table 2: Computation time for a sigle solver call (in seconds)

Solvers	Calculation	Communication	Synchronization
Normal solver	0.540	0.419	0.041
Asymptotic solver	0.181	0.040	0.031

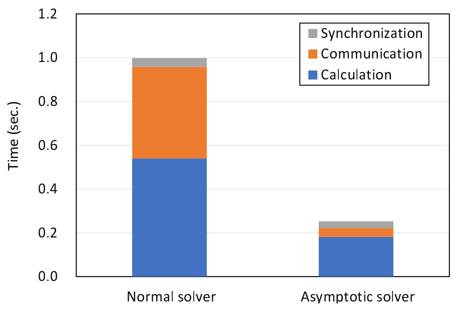


Fig. 5: Comparison of parallel solvers

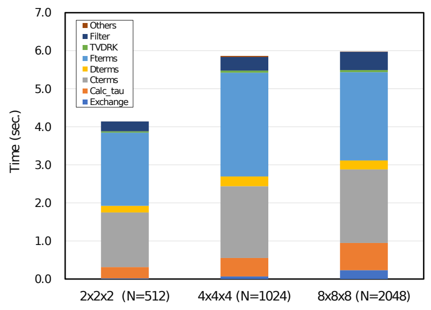


Fig. 6: Weak scaling characteristics of the code

4.3 Comparison of the Two Parallel Solvers

Here, the two parallel solvers in Algorithms 1 and 2 are compared for the DNS with the grid points $N^3 = 2048^3$. The calculation with the optimized DNS code is performed on 512 MPI processes with an $8 \times 8 \times 8$ layout on 32 VEs (512 cores). Since the number of grid points in any direction in a process is 256, the asymptotic solver is usually selected for the calculation of the first-order derivatives. The normal solver, however, is used for comparison.

Table 2 and Figure 5 show the computation times of both solvers for a single solver call. These solvers are called multiple times in one time step. Calculation and communication times are greatly decreased in the asymptotic solver, and approximately the 4-fold performance of the asymptotic solver is achieved. The normal solver uses the MPI function `MPI_Allgather` to construct the right-hand-side in the system (14) in order to calculate the coefficients s_p and t_p, and has to solve the same linear system constructed in duplicate in all MPI processes simultaneously. On the other hand, a function `MPI_Sendrecv` is sufficient to get the values $x^R_{p+1,1}$ and $x^R_{p-1,M}$ from the adjacent MPI processes and the values are calculate by Eq. (19) in all processes concurrently in Algorithm 2.

In addition, the number of compute nodes used increases to solve problems with a large number of grid points. The communication paths of the `MPI_Allgather` function among all compute nodes are more complicated than the `MPI_Sendrecv` neighbor communications, and a larger communication time is required. Therefore, the asymptotic solver can significantly reduce computation time not only for large time-steps simulations, but also for simulations with a larger number of grid points.

4.4 Weak Scaling Characteristics

Here, the same number of grid points is assigned to all MPI processes to check for weak scaling characteristics. Three cases of DNSs with

(1) the number of grid points 512^3 partitioning into 8 MPI processes with a $2 \times 2 \times 2$ layout on a VE,
(2) the number of gird point 1024^3 partitioning into 64 MPI processes with a $4 \times 4 \times 4$ layout on 4 VEs, and
(3) the number of gird point 2048^3 partitioning into 512 MPI processes with a $8 \times 8 \times 8$ layout on 32 VEs

are run on SX-Aurora TSUBASA. The number of grid points computed in one MPI process is 256^3 in all cases. Computation time for the three cases are shown in Fig. 6.

Computation times for Cases 2 and 3 are longer than that for Case 1. This is because Case 1 can be carried out within a VE and therefore MPI communications in the x direction is processed within the VE via interconnection between cores, whereas data by MPI communications for Cases 2 and 3 are transferred via PCIe switches in the same node and among four nodes, respectively. The communication time is slightly increased for Cases 2 and 3. It is found that good weak scalability of the code is achieved with several VEs.

5 Conclusions

Turbulent flows play an important role in many flow-related phenomena that occur in various fields of science and technology. It is important to understand and elucidate their universal nature. DNS of turbulent flows in a widely used method for clarifying the properties of turbulence because it solves the governing equations directly without modeling and resolves eddy motion on the smallest scales.

In this study, at first, the asymptotic parallel solver for the reduced linear system obtained from the parallelization of the linear system for the compact finite difference scheme is proposed. A parallel DNS code incorporating the solver as well the Mattor's solver has been developed for isothermal compressible turbulent flows in a cube using a finite difference method. The convective terms of the governing equations are calculated using an eighth-order compact difference scheme, while the viscous term is calculated using an eighth-order central difference scheme.

The computational cost of the asymptotic solver is lower than that of the normal solver in the condition that the number of grid points along a Cartesian coordinate in a task is greater than or equal to 64. In addition, the code was been optimized on the supercomputer SX-Aurora TSUBASA. As a result, a 1.7-fold speedup and excellent weak scalability were performed. A further code tuning for scalar processors like the supercomputer Fugaku is underway. Larger DNSs with over 8192^3 grid points are possible to compute.

Acknowledgements. This work was partially conducted and funded at Joint-Research Division of High-Performance Computing (NEC) of Cyberscience Center at Tohoku University. This study was partially supported by JSPS KAKENHI Grant number JP23K11124 and the HPCI System Research project (Project ID: hp230143).

References

1. Gottlieb, S., Shu, C.W.: Total variation diminishing Runge-Kutta schemes. Math. Comput. **67**, 73–85 (1998). https://doi.org/10.1090/S0025-5718-98-00913-2
2. Ishihara, T., Gotoh, T., Kaneda, Y.: Study of high–reynolds number isotropic turbulence by direct numerical simulation. Annu. Rev. of Fluid Mech. **41**(1), 165–180 (2009). https://doi.org/10.1146/annurev.fluid.010908.165203
3. Ishihara, T., Morishita, K., Yokokawa, M., Uno, A., Kaneda, Y.: Energy spectrum in high-resolution direct numerical simulations of turbulence. Phys. Rev. Fluids **1**, 082403 (Dec 2016). https://doi.org/10.1103/PhysRevFluids.1.082403
4. Komatsu, K., et al.: Performance Evaluation of a Vector Supercomputer SX-Aurora TSUBASA. In: SC'18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 685–696. Dallas, TX, U.S.A. (2018). https://doi.org/10.1109/SC.2018.00057
5. Lele, S.K.: Compact finite difference schemes with spectral-like resolution. J. Comput. Phys. **103**(1), 16–42 (1992). https://doi.org/10.1016/0021-9991(92)90324-R
6. Mattor, N., Williams, T.J., Hewett, D.W.: Algorithm for solving tridiagonal matrix problems in parallel. Parallel Comput. **21**(11), 1769–1782 (1995). https://doi.org/10.1016/0167-8191(95)00033-0
7. Petersen, M.R., Livescu, D.: Forcing for statistically stationary compressible isotropic turbulence. Phys. Fluids **22**(11), 116101 (2010). https://doi.org/10.1063/1.3488793
8. Sakurai, Y., Ishihara, T.: Direct numerical simulations of compressible isothermal turbulence in a periodic box: Reynolds number and resolution-level dependence. Phys. Rev. Fluids **8**, 084606 (2023). https://doi.org/10.1103/PhysRevFluids.8.084606
9. Takahashi, K., et al.: Performance Evaluation of a Next-Generation SX-Aurora TSUBASA Vector Supercomputer. In: Bhatele, A., Hammond, J., Baboulin, M., Kruse, C. (eds.) High Performance Computing - Proc. 38th Int. Conf. High Performance 2023. pp. 359–378. Springer Science and Business Media Deutschland GmbH, Germany (2023). https://doi.org/10.1007/978-3-031-32041-5_19
10. Yokokawa, M., et al.: I/O performance of the SX-Aurora TSUBASA. In: 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 27–35 (2020). https://doi.org/10.1109/IPDPSW50202.2020.00014