Computers & Fluids 76 (2013) 170–177

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

Highly parallel particle-laden flow solver for turbulence research

Peter J. Ireland, T. Vaithianathan [1], Parvez S. Sukheswalla, Baidurja Ray, Lance R. Collins *

Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA
International Collaboration for Turbulence Research

A R T I C L E I N F O

Article history:
Received 6 November 2012
Received in revised form 5 January 2013
Accepted 29 January 2013
Available online 16 February 2013

Keywords:
Isotropic turbulence
Homogeneous turbulent shear flow
Inertial particles
Direct numerical simulation

A B S T R A C T

In this paper, we present a Highly Parallel Particle-laden flow Solver for Turbulence Research (HiPPSTR). HiPPSTR is designed to perform three-dimensional direct numerical simulations of homogeneous turbulent flows using a pseudospectral algorithm with Lagrangian tracking of inertial point and/or fluid particles with one-way coupling on massively parallel architectures, and is the most general and efficient multiphase flow solver of its kind. We discuss the governing equations, parallelization strategies, time integration techniques, and interpolation methods used by HiPPSTR. By quantifying the errors in the numerical solution, we obtain optimal parameters for a given domain size and Reynolds number, and thereby achieve good parallel scaling on $\mathcal{O}(10^4)$ processors.

1. Introduction

Turbulent flows laden with inertial particles (that is, particles denser than the carrier fluid) are ubiquitous in both industry and the environment. Natural phenomena such as atmospheric cloud formation [1–3], plankton distributions in the sea [4], and planetesimal formation in the early universe [5,6] are all influenced by particle–turbulence interactions. Inertial particle dynamics also impact engineered systems such as spray combustors [7], aerosol drug delivery systems [8], and powder manufacturing [9,10], among many other systems [11]. Despite extensive research, however, open questions remain about the distribution of these particles in the flow, their settling speed due to gravity, and their collision rates. This is due in part to our incomplete understanding of the effect of the broad spectrum of flow scales that exists at intermediate or high Reynolds numbers.

Since inertial particle dynamics are strongly sensitive to the smallest scales in the flow [11], large-eddy simulation, with its associated small-scale modeling, has difficulties representing sub-filter particle dynamics accurately, including particle clustering that is driven by the Kolmogorov scales [12,13]. Consequently, our investigations rely on the direct numerical simulation (DNS) of the three-dimensional Navier–Stokes equations and the Maxey and Riley equation for particle motion [14]. DNS has proven to be an extremely effective tool for investigating inertial particle dynamics, albeit at modest values of the Reynolds number due to the heavy computational demands of resolving all relevant temporal and spatial scales.

To extend the range of Reynolds numbers that can be simulated, we have developed a more advanced DNS code: the Highly Parallel, Particle-laden flow Solver for Turbulence Research (HiPPSTR). HiPPSTR is capable of simulating inertial particle motion in homogeneous turbulent flows on thousands of processors using a pseudospectral algorithm. The main objective of this paper is to document the solution strategy and numerical algorithms involved in solving the relevant governing equations.

This paper is organized as follows. In Section 2, we show the equations governing the fluid and particle motion and the underlying assumptions of the flow solver. We discuss the time integration and interpolation techniques in Sections 3 and 4, respectively, complete with an error analysis for optimizing code performance. Section 5 then describes the parallelization strategy, and Section 6 provides conclusions.

2. Governing equations

2.1. Fluid phase

The underlying flow solver is based on the algorithm presented in Ref. [15] and summarized below. It is capable of simulating both homogeneous isotropic turbulence (HIT) and homogeneous turbulent shear flow (HTSF) with a pseudospectral algorithm, while avoiding the troublesome remeshing step of earlier algorithms [16]. The governing equations for the flow of an incompressible

* Corresponding author at: Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA. Tel.: +1 607 255 9679; fax: +1 607 255 9606.
E-mail address: lc246@cornell.edu (L.R. Collins).
[1] Present address: INVISTA S.a.r.l., PO Box 1003, Orange, TX 77631, USA.

0045-7930/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compfluid.2013.01.020

fluid are the continuity and Navier–Stokes equations, here presented in rotational form

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1}$$

$$\frac{\partial u_i}{\partial t} + \epsilon_{ijk}\omega_j u_k = -\frac{\partial(p/\rho + \frac{1}{2}u^2)}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_j} + f_i, \tag{2}$$

where u_i is the velocity vector (with magnitude u), ϵ_{ijk} is the alternating unit symbol, ω_i is the vorticity, p is the pressure, ρ is the density, ν is the kinematic viscosity, and f_i is a large-scale forcing function that is added to achieve stationary turbulence for HIT. Aside from the term $\frac{1}{2}u^2$, which ultimately will be subsumed into a modified pressure term—see Eq. (10) below, this form of the Navier–Stokes equation has only six nonlinear terms, as compared to nine in the standard form, which reduces the expense of computation and renders the solution method more stable.

2.1.1. Reynolds decomposition

We apply the standard Reynolds decomposition on the velocity, vorticity, and pressure, yielding

$$u_i = U_i + u'_i, \tag{3}$$

$$\omega_i = \Omega_i + \omega'_i, \tag{4}$$

$$p = P + p', \tag{5}$$

where capitalized variables denote mean quantities and primed variables denote fluctuating quantities.

Subtracting the ensemble average of Eqs. (1) and (2) from Eqs. (1) and (2), respectively, and specializing for the case of homogeneous turbulence (homogeneous turbulence implies all single-point statistics are independent of position, with the exception of the mean velocity U_i, which can vary linearly while still preserving homogeneity for all higher-order velocity statistics of the flow field) yields

$$\frac{\partial u'_i}{\partial x_i} = 0, \tag{6}$$

$$\frac{\partial u'_i}{\partial t} + \epsilon_{ijk}\left(\Omega_j u'_k + \omega'_j U_k + \omega'_j u'_k\right) = -\frac{\partial\left[p'/\rho + u'_j U_j + \frac{1}{2}u'^2\right]}{\partial x_i} + \nu\frac{\partial^2 u'_i}{\partial x_j \partial x_j} + f'_i. \tag{7}$$

We consider HTSF with a uniform mean shear rate $\mathcal{S}$, where coordinates x_1, x_2 and x_3 are the streamwise, spanwise and shear directions, respectively, without loss of generality. Under these conditions, the mean velocity and vorticity can be expressed as

$$\mathbf{U} = (\mathcal{S}x_3, 0, 0), \tag{8}$$

$$\mathbf{\Omega} = (0, \mathcal{S}, 0), \tag{9}$$

thus reducing Eq. (7) to

$$\frac{\partial u'_i}{\partial t} + u'_3\mathcal{S}\delta_{i1} + \mathcal{S}x_3\frac{\partial u'_i}{\partial x_1} + \epsilon_{ijk}\omega'_j u'_k = -\frac{\partial p^*}{\partial x_i} + \nu\frac{\partial^2 u'_i}{\partial x_j \partial x_j} + f'_i, \tag{10}$$

where the modified pressure $p^* \equiv p'/\rho + \frac{1}{2}u'^2$. The above equations are written to encompass both HIT and HTSF; for HIT $\mathcal{S} = 0$ and $f'_i \neq 0$, while for HTSF $\mathcal{S} \neq 0$ and $f'_i = 0$.

2.1.2. Spectral transforms

Pseudospectral algorithms take advantage of the fact that the fluid velocity and pressure satisfy periodic boundary conditions. For HIT the period is simply the box domain, and so for a box of length $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ in the x_1, x_2 and x_3 directions, respectively, the wavevector for the transforms takes the conventional form

$$\mathbf{k} \equiv \left(\frac{2\pi}{\mathcal{L}_1}n_1, \frac{2\pi}{\mathcal{L}_2}n_2, \frac{2\pi}{\mathcal{L}_3}n_3\right), \tag{11}$$

where the integers n_i vary between $-N_i/2 + 1$ and $N_i/2$ for N_i grid points in each direction [17]. Note that the code allows you to independently set the number of grid points N_i and the box length $\mathcal{L}_i$ in each direction.

For HTSF, the presence of the mean shear modifies this to a "shear-periodic boundary condition", as illustrated in Fig. 1. The classic Rogallo [16] algorithm eliminated this complication by solving the equations of motion in a frame of reference that deforms with the mean shear, enabling standard fast Fourier transforms (FFTs) to be applied; however, in order to relieve the distortion of the physical space grid with time, the standard Rogallo algorithm maps the variables on the positively deformed mesh onto a mesh that is deformed in the opposite direction, a step known as "remeshing". Remeshing leads to sudden changes in the turbulent kinetic energy and dissipation rate as a consequence of zeroing out certain wavenumbers. The algorithm of Brucker et al. [15] eliminates this remeshing step by directly applying the spectral transform in a fixed frame of reference while incorporating the phase shift that results from the mean shear analytically. We begin by defining a modified wavevector as

$$k'_i \equiv k_i - \mathcal{S}tk_1\delta_{i3} \tag{12}$$

and the Fourier decomposition of a variable $\phi(\mathbf{x}, t)$ as

$$\hat{\phi}(\mathbf{k}, t) \equiv \sum_{\mathbf{x}}\phi(\mathbf{x}, t)\exp\left(-Ik'_i x_i\right), \tag{13}$$

where $\hat{\phi}(\mathbf{k}, t)$ is the forward spectral transform of $\phi(\mathbf{x}, t)$, denoted as $\mathcal{F}\{\phi(\mathbf{x}, t)\}$, and I is the imaginary unit. The inclusion of the cross term $\mathcal{S}tk_1\delta_{i3}$ in the modified wavevector accounts for the shear-periodic boundary condition. We can similarly define a modified discrete backward spectral transform of $\hat{\phi}(\mathbf{k}, t)$ as

$$\phi(\mathbf{x}, t) = \frac{1}{N_1N_2N_3}\sum_{\mathbf{k}}\hat{\phi}(\mathbf{k}, t)\exp\left(Ik'_i x_i\right). \tag{14}$$

With these definitions, we can transform (6) and (10) to spectral space to obtain

$$k'_i\hat{u}'_i = 0, \tag{15}$$

$$\left[\frac{\partial}{\partial t} + \nu k'^2\right]\hat{u}'_i = \left(-\delta_{im} + \frac{k'_i k'_m}{k'^2}\right)\epsilon_{mjn}\mathcal{F}\left\{\omega'_j u'_n\right\} + 2\frac{k'_i}{k'^2}k'_1\mathcal{S}\hat{u}'_3 - \delta_{i1}\mathcal{S}\hat{u}'_3 + \hat{f}'_i, \tag{16}$$

where $k'^2 = k'_i k'_i$. Note that we have used the incompressibility of the flow, as specified by Eq. (15), to project the pressure out of Eq. (16) (refer to Ref. [15] for a detailed derivation).

The direct evaluation of the convolution $\mathcal{F}\left\{\omega'_j u'_n\right\}$ would be computationally prohibitively expensive. The "pseudospectral approximation" is obtained by first transforming the velocity and

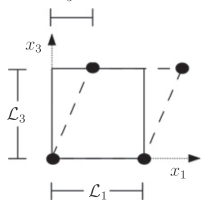

Fig. 1. Shear-periodic boundary conditions due to the mean shear applied in the x_3 direction. The orthogonal domain is shown with the solid lines, while the shear-periodic boundary points are indicated by the black dots.

vorticity into physical space, computing the product, then transforming the result back into spectral space [18]. The transformations between physical and spectral spaces are accomplished using modified fast Fourier transforms (FFTs) that are described in Section 5. The pseudospectral algorithm introduces aliasing errors that are eliminated by a combination of spherical truncation and phase-shifting (refer to Appendix A of Ref. [19] for details).

2.1.3. Low wavenumber forcing

To achieve stationary turbulence with HIT, we must introduce forcing to compensate for the energy that is dissipated [20]. The forcing is usually restricted to low wavenumbers (large scales) with the hope that the intermediate and high wavenumbers behave in a natural way that is not overly influenced by the forcing scheme. The two most widely used forcing schemes in the literature are “deterministic” and “random” forcing schemes. Both are described below.

The deterministic forcing scheme [21] first computes the amount of turbulent kinetic energy dissipated in a given time step h, $\Delta E_{\mathrm{tot}}(h)$. This energy is restored at the end of each time step by augmenting the velocity over the forcing range between $k_{\mathrm{f,min}}$ and $k_{\mathrm{f,max}}$ so as to precisely compensate for the dissipated energy

$$\hat{\mathbf{u}}(\mathbf{k},t_0+h)=\hat{\mathbf{u}}(\mathbf{k},t_0+h)\sqrt{1+\frac{\Delta E_{\mathrm{tot}}(h)}{\int_{k_{\mathrm{f,min}}}^{k_{\mathrm{f,max}}}E(k,t_0+h)dk}}, \tag{17}$$

where $E(k, t_0 + h)$ represents the turbulent kinetic energy in a wavenumber shell with magnitude k at time $t_0 + h$.

The alternative forcing scheme is based on introducing a stochastic forcing function, $\hat{f}'_i(\mathbf{k},t)$ [20]. This function is non-zero only over the forcing range between $k_{\mathrm{f,min}}$ and $k_{\mathrm{f,max}}$ and evolves according to a vector-valued complex Ornstein–Uhlenbeck process [20], as shown below

$$d\hat{f}'_i(\mathbf{k},t)=-\frac{\hat{f}'_i(\mathbf{k},t)}{T_f}dt+\sqrt{\frac{2\sigma_f^2}{T_f}}dW_i(\mathbf{k},t),\quad \forall k(k_{\mathrm{f,min}},k_{\mathrm{f,max}}), \tag{18}$$

where T_f is the integral time-scale of the random forcing, σ_f^2 denotes its strength, and $W_i(\mathbf{k}, t)$ is the Wiener process whose increment dW_i is defined to be a joint-normal, with zero mean, and covariance given by

$$\langle dW_i dW_j^*\rangle = h\delta_{ij}. \tag{19}$$

While implementing on a computer, we set the increment $dW_i=(\alpha_i+I\beta_i)\sqrt{h/2}$, where α_i and β_i are two independent random numbers drawn from a standard normal distribution.

2.2. Particle phase

The Maxey and Riley equation [14] is used for simulating spherical, non-deforming particles in the flow. We take the particles to be small (i.e., $d/\eta \ll 1$, where d is the particle diameter, $\eta \equiv \nu^{3/4}/\epsilon^{1/4}$ is the Kolmogorov length scale, and ϵ is the average turbulent kinetic energy dissipation rate) and heavy (i.e., $\rho_p/\rho \gg 1$, where ρ_p and ρ are the densities of the particles and fluid, respectively). We also assume that the particles are subjected to only linear drag forces, which is valid when the particle Reynolds number $Re_p \equiv \|\mathbf{u}(\mathbf{x})-\mathbf{v}\|d/\nu < 1$. Here, $\mathbf{u}(\mathbf{x})$ denotes the undisturbed fluid velocity at the particle position, and $\mathbf{v}$ denotes the particle velocity. For HTSF, we express $\mathbf{u}(\mathbf{x})$ as

$$u_i(\mathbf{x}) = u'_i(\mathbf{x}) + \mathcal{S}x_3\delta_{i1} \tag{20}$$

and for HIT, $u_i(\mathbf{x}) = u'_i(\mathbf{x})$.

Under these assumptions, the Maxey and Riley equation simplifies to a system of ordinary differential equations for the position and velocity of a given particle

$$\frac{d\mathbf{x}}{dt}=\mathbf{v}, \tag{21}$$

$$\frac{d\mathbf{v}}{dt}=\frac{\mathbf{u}(\mathbf{x})-\mathbf{v}}{\tau_p}+\mathbf{g}, \tag{22}$$

where $\tau_p \equiv \frac{\rho_p}{\rho}\frac{d^2}{18\nu}$ is the particle response time and $\mathbf{g}$ is the gravitational acceleration vector. Note that the numerical solution of (21) and (22) requires an interpolation of grid values of the fluid velocity to the location at the center of each particle. The interpolation methods used are discussed in Section 4.

The influence of particles on the continuity and momentum equations is neglected due to the low volume ($\mathcal{O}(10^{-6})$) and mass ($\mathcal{O}(10^{-3})$) loadings [22,23], and therefore we consider only one-way coupling between the flow field and the particles. All particles are represented as point-particles, and collisions are neglected [24,25].

3. Time integration

3.1. Fluid update

To calculate the temporal evolution of the fluid, we introduce the integrating factor

$$A_{ij}(t') \equiv \exp\left[\int_0^{t'}\left(\nu k'^2\delta_{ij}-2\delta_{i3}\delta_{j3}\mathcal{S}\frac{k'_1k'_3}{k'^2}\right)dt\right] \tag{23}$$

into (16), yielding

$$\frac{\partial}{\partial t}\left[A_{ij}(t)\hat{u}'_j\right]=A_{ij}(t)RHS_j(t), \tag{24}$$

where for ease of notation, we have defined

$$RHS_j \equiv \left(-\delta_{jm}+\frac{k'_jk'_m}{k'^2}\right)\epsilon_{m\ell n}\mathcal{F}\{\omega'_\ell u'_n\}+2\frac{k'_j}{k'^2}k'_1\mathcal{S}\hat{u}'_3-\delta_{j1}\mathcal{S}\hat{u}'_3. \tag{25}$$

Integrating (24) from time t_0 to time $t_0 + h$, we obtain

$$A_{ij}(t_0+h)\hat{u}'_j(t_0+h)=\hat{u}'_j(t_0)A_{ij}(t_0)+\int_{t_0}^{t_0+h}RHS_j(t)A_{ij}(t)dt. \tag{26}$$

The trapezoid rule is used to approximate the integral on the RHS of (26), yielding

$$\int_{t_0}^{t_0+h}RHS_j(t)A_{ij}(t)dt \approx \frac{h}{2}RHS_j(t_0)A_{ij}(t_0)+\frac{h}{2}RHS_j(t_0+h)A_{ij}(t_0+h). \tag{27}$$

As the integrand in Eq. (23) is a rational function of time, we can evaluate $A_{ij}(t')$ analytically, substitute the result into Eq. (26), and apply the approximation in Eq. (27) to formulate a second-order Runge–Kutta method (RK2) for Eq. (26) as follows:

$$\begin{aligned}\hat{u}'_i(t_0+h)=&\ \hat{u}'_j(t_0)\exp\left[-\int_{t_0}^{t_0+h}\left(\nu k'^2\delta_{ij}-2\delta_{i3}\delta_{j3}\mathcal{S}\frac{k'_1k'_3}{k'^2}\right)dt\right]\\&+\frac{h}{2}RHS_j(t_0)\\&\times\exp\left[-\int_{t_0}^{t_0+h}\left(\nu k'^2\delta_{ij}-2\delta_{i3}\delta_{j3}\mathcal{S}\frac{k'_1k'_3}{k'^2}\right)dt\right]\\&+\frac{h}{2}RHS_i(t_0+h).\end{aligned} \tag{28}$$

To avoid convective instabilities, the time step h is chosen to satisfy the Courant condition [26]

$$\frac{u'_{\max}h\sqrt{3}}{\min(\Delta x_1,\Delta x_2,\Delta x_3)} \lesssim 0.5, \tag{29}$$

where $u'_{\max}$ is the maximum velocity fluctuation in the domain, and Δx_i is the grid spacing in the ith coordinate direction.

3.2. Particle update

Inertial particles introduce another time scale into the simulation, namely the particle response time τ_p, which is a parameter that is independent of the fluid time step h. For the case where $\tau_p \ll h$, the system defined by (21) and (22) is stiff, and traditional explicit Runge–Kutta schemes require an extremely small time step for numerical accuracy and stability. Note that updating the particle equations implicitly would be prohibitively expensive, as it necessarily would involve multiple interpolations of the fluid velocity to the particle location. An alternative scheme we have used in the past is known as "subcycling", whereby the time step used to update the particle field is chosen so as to ensure $h/\tau_p \leqslant 0.1$. The fluid velocity field is linearly interpolated between the values at consecutive fluid time steps. While this method improves the accuracy of the RK2 method, it is computationally expensive, particularly for particles with low values of τ_p. Additionally, subcycling can degrade the parallel scaling of the code for simulations with a distribution of values of τ_p. The distribution of τ_p may vary from processor to processor, causing the entire calculation to run at the speed of the slowest processor (i.e., the processor with the highest load).

We have overcome these limitations by formulating an alternative second-order scheme that is uniformly effective over the entire range of τ_p, including the challenging case of $\tau_p \ll h$. The numerical scheme is based on "exponential integrators" [27]. Exponential integrators are a broad class of methods that treat the linear term in (22) exactly and the inhomogeneous part using an exponential quadrature.

The starting point for all of the updates is as follows:

$$\mathbf{v}(t_0 + h) = e^{-h/\tau_p}\mathbf{v}(t_0) + w_1\mathbf{u}[\mathbf{x}(t_0)] + w_2\mathbf{u}[\mathbf{x}(t_0) + \mathbf{v}(t_0)h] + (1 - e^{-h/\tau_p})\tau_p\mathbf{g}, \tag{30}$$

where the weights w_1 and w_2 define the method. In our standard RK2 implementation, the weights are defined by

$$w_1^{RK} = 0.5(h/\tau_p)e^{-h/\tau_p}, \quad w_2^{RK} = 0.5(h/\tau_p), \tag{31}$$

where the superscript "RK" indicates the standard RK method. In the new formulation based on the exponential quadrature, we redefine the weights as follows

$$w_1 \equiv \left(\frac{h}{\tau_p}\right)\left[\varphi_1\left(-\frac{h}{\tau_p}\right) - \varphi_2\left(-\frac{h}{\tau_p}\right)\right], \quad w_2 \equiv \left(\frac{h}{\tau_p}\right)\varphi_2\left(-\frac{h}{\tau_p}\right) \tag{32}$$

and

$$\varphi_1(z) \equiv \frac{e^z - 1}{z}, \quad \varphi_2(z) \equiv \frac{e^z - z - 1}{z^2}. \tag{33}$$

Fig. 2 compares the behavior of the RK weights with the newly defined exponential weights as a function of h/τ_p for particles with Stokes numbers $St \equiv \tau_p/\tau_\eta \leqslant 0.1$, where $\tau_\eta \equiv (\nu/\epsilon)^{1/2}$ is the Kolmogorov time scale. In the limit $h/\tau_p \to 0$, we see all the weights converge, implying consistency between the RK and exponential-quadrature methods in the limit of a highly resolved simulation, and confirming the new method's second-order accuracy. In the other limit, that is $St \to 0$ (or equivalently $h/\tau_p \to \infty$), we see the standard RK weights diverge, which clearly will lead to large numerical errors. In contrast, the exponential-quadrature weights have limits $w_1 \to 0$ and $w_2 \to 1$, implying from Eq. (30) that the particle velocity smoothly approaches $\mathbf{u}[\mathbf{x}(t_0) + \mathbf{v}(t_0)h] + \tau_p\mathbf{g}$ in this limit, which is the fluid velocity (corrected for gravitational settling) at the anticipated particle position at time $t_0 + h$ based on an Euler time step, i.e., the correct limit. Furthermore, the truncation error per step (as $\tau_p \to 0$) for the RK2 scheme is $\mathcal{O}\left(h^3\tau_p^{-3}\mathbf{u}\right)$

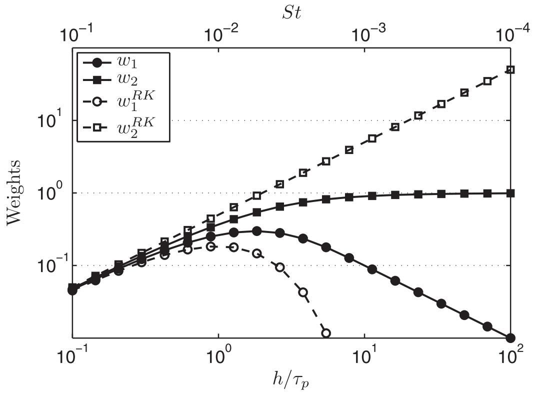

Fig. 2. Functional behavior of weights for the exponential integrator (denoted as solid lines with closed symbols) and the Runge–Kutta scheme (denoted as dashed lines with open symbols) in a Stokes number range of interest.

while that of the new scheme is $\mathcal{O}\left(h^3\tau_p^{-1}\frac{d^2\mathbf{u}}{dt^2}\right)$, indicating that the error constant for the RK2 is $\mathcal{O}\left(\tau_p^{-2}\right)$ greater than that of the exponential integrator.

We quantify the integration errors in the two schemes by performing a numerical experiment on a domain with 512^3 grid points, 2,097,152 particles, and small scale resolution $k_{\max}\eta = 2$ (where $k_{\max} = N\sqrt{2}/3$ is the maximum resolved wavenumber magnitude). We first use a Courant number of 0.5 to advance the particle field for one time step, with both the RK2 scheme and the exponential integrator scheme. To reduce time-stepping errors and attain a baseline case for quantifying these errors, we perform a set of low-Courant-number simulations, starting from the same initial condition as the simulations with a Courant number of 0.5. We advance the particle field for 1000 time steps using the exponential integrator scheme with the Courant number of 0.0005 (i.e., a time step equal to 1/1000th that of the other simulations) and $h/\tau_p \leqslant 0.1$. The estimated root-mean-square (RMS) velocity errors incurred during numerical integration for the exponential integrator and RK2 schemes are plotted in Fig. 3a. The time to update the positions and velocities of the particles for each of the two schemes is plotted in Fig. 3b, normalized by the time to complete one step of the Navier–Stokes solver, which remains constant for all the cases. From Fig. 3a and b, it is evident that the exponential integrator outperforms the RK2 scheme, both in terms of numerical accuracy and computational expense, particularly for high values of h/τ_p (i.e., low values of St).

4. Interpolation

4.1. Interpolation methods

As discussed in Section 2.2, the solution of (21) and (22) requires an interpolation of grid values of the fluid velocity to the particle centers. In principle, this interpolation could be performed exactly (provided that the grid is sufficiently fine to capture all scales of motion) using a spectral interpolation [28]. For a typical simulation involving at least a million particles, however, such an interpolation technique is prohibitively expensive.

To compute the fluid velocities at the particle centers, we have embedded several different interpolation methods into our code. They include linear, Lagrangian [29], Hermite [30], shape function method (SFM) [28], and B-spline interpolation [31]. For the Lagrangian and B-spline interpolations, we use 4, 6, 8, and 10 interpolation points (denoted as Lag$P$ and BSpline$P$, where P is the number of interpolation points). These interpolation methods are

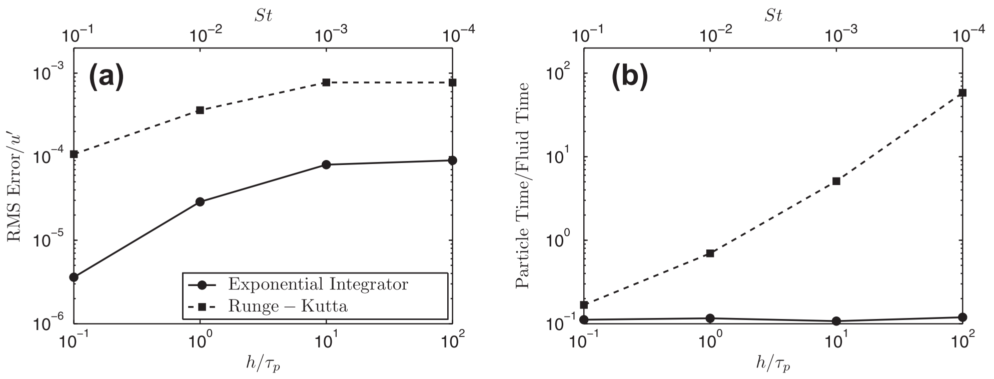

Fig. 3. (a) RMS velocity integration errors in the particle update (normalized by the RMS fluid velocity) as a function of h/τ_p and St for both the exponential integrator and RK2 schemes. (b) Time to update the positions and velocities of 2,097,152 particles for both schemes, normalized by the time to complete one step of the Navier–Stokes solver.

compared in Fig. 4, both in terms of accuracy and computational expense. All data in Fig. 4 is from a simulation with 512^3 grid points, 2,097,152 particles, and $k_{\max}\eta = 2$. In all cases, we determined the "exact" values of the particle velocities from a spectral interpolation, and defined the velocity error of a particular scheme as the RMS difference between the velocities obtained from that scheme and the spectral interpolation at a given time.

Based on Fig. 4, the B-spline interpolation scheme, which is optimized for spectral simulations [31], provides the best trade-off between computational expense and accuracy. To determine the optimal number of spline points, we compare the interpolation error to the local time-stepping error, as given in Fig. 3a. From these results, for a given run, we choose the number of spline points so that the interpolation and time-stepping errors are of the same order of magnitude.

4.2. Interpolation in shear flow

All our interpolation methods are designed for velocities which are stored on an orthogonal grid in physical space. As discussed in Section 2.1.2, the Brucker et al. [15] algorithm for the fluid velocity stores the physical-space velocity on an orthogonal mesh that is shear periodic. Consequently, the interpolation methods must be adapted to accommodate particles that require data from shear-periodic grid points. As an example, consider linear interpolation for the particle shown in Fig. 5. The schematic is shown in two dimensions on a 4 × 4 grid for clarity.

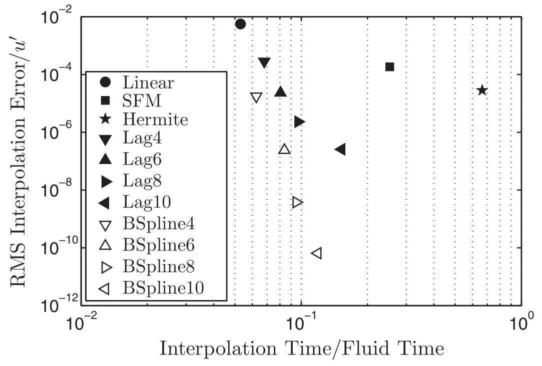

Fig. 4. Interpolation error for different methods as a function of computation time, which is normalized by the time for one step of the Navier–Stokes solver. All errors are normalized by the RMS fluid velocity, u'.

Grid values of velocity are stored at the filled circles, and periodic points are shown as squares: standard periodic points are filled squares and shear-periodic points are open squares. The dotted lines in the figure illustrate the phase shift that occurs between points A, B, C and D on the bottom surface and the shear-periodic points on the top surface. To complete the interpolation, the data on the top surface must be phase shifted back to the orthogonal mesh points A′, B′, C′ and D′. This is accomplished by applying a one-dimensional spectral transform along the x_1-direction

$$\check{\mathbf{u}}'_{\text{boundary}}(k_1,x_2,x_3) \equiv \sum_{x_1} \mathbf{u}'_{\text{boundary}}(x_1,x_2,x_3)\exp(-Ik_1x_1), \tag{34}$$

multiplying by the following factor to shift the points in the x_1-direction

$$\check{\mathbf{u}}''_{\text{boundary}}(k_1,x_2,x_3) \equiv \check{\mathbf{u}}'_{\text{boundary}}(k_1,x_2,x_3)\exp(-Ik_1\mathcal{S}t\mathcal{L}_3) \tag{35}$$

and converting back to physical space to recover the values at the open squares A′, B′, C′ and D′ (denoted $\mathbf{u}''_{boundary}$ below)

$$\mathbf{u}''_{\text{boundary}}(x_1,x_2,x_3) \equiv \frac{1}{N_1}\sum_{k_1} \check{\mathbf{u}}''_{\text{boundary}}(k_1,x_2,x_3)\exp(Ik_1x_1). \tag{36}$$

Given these velocity values, we can complete the interpolation using the standard approaches for orthogonal grids.

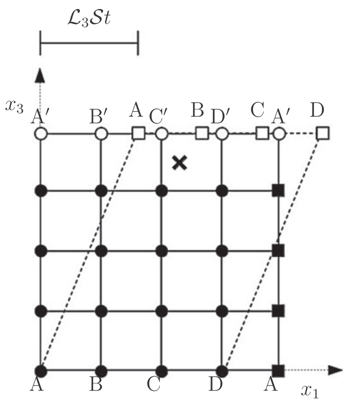

Fig. 5. Example to demonstrate the interpolation approach for shear-periodic boundary conditions on a 4 × 4 grid. The particle is indicated by the symbol ×, the stored grid values by filled circles, the standard periodic boundary points by filled squares, and the shear-periodic boundary points by open squares. Corresponding boundary points are labeled A, B, C, D to illustrate the shear-periodic boundary condition. We determine the velocity values at the open circles (A′, B′, C′, D′) to complete the interpolation.

5. Parallelization

5.1. Fluid update

As the Reynolds number increases, the range of spatial and temporal scales also increases, resulting in the approximate scaling $N_T \sim R_\lambda^{9/2}$, where N_T is the total number of grid points, $R_\lambda \equiv 2K\sqrt{5/(3\nu\epsilon)}$ is the Reynolds number based on the Taylor microscale, and K is the average turbulent kinetic energy. The rapid increase in the computational load with Reynolds number has limited DNS to relatively modest values of this parameter. Furthermore, with the slowing of processor speeds from Moore's Law over the past decade, the strategy for advancing supercomputer speeds has shifted toward increasing the number of accessible processors on the system. Exploiting these massively parallel architectures requires a new class of algorithms.

The previous version of the code utilized one-dimensional ("plane") parallel domain decomposition. Using this parallelization strategy, a domain with N^3 grid points can be parallelized on at most N processors, which realistically can accommodate a maximum simulation size of 1024^3 grid points or Reynolds numbers in the range $R_\lambda \lesssim 400$.

To increase the granularity of the parallelization, we have modified the code to allow two-dimensional ("pencil") domain decomposition. With pencil decomposition, we are able to parallelize a domain with N^3 grid points on up to N^2 processors. Fig. 6 illustrates the difference between the plane and pencil domain decompositions.

While the Message Passing Interface (MPI) is used as the communication environment, the detailed bookkeeping for the FFTs with pencil decomposition is performed using the P3DFFT library [32]. P3DFFT uses standard FFT libraries (such as FFTW [33] or ESSL [34]) to compute the three-dimensional FFT of the velocity or pressure distributed over the two-dimensional array of processors. The FFTs are necessary for computing the nonlinear convolution sums described in Section 2.1.1 and constitute the bulk of the computational expense of the overall fluid solver. They are performed as a series of one-dimensional FFTs in each of the three coordinate dimensions. Starting from a spectral-space variable $\hat{\phi}(k_1,k_2,k_3)$, where the entire k_3-dimension is stored on each pencil, we first perform $N_1 \times N_2$ one-dimensional complex-to-complex pencil transforms in the x_3 direction to obtain

$$\check{\phi}(k_1,k_2,x_3) \equiv \frac{1}{N_3}\sum_{k_3}\hat{\phi}(k_1,k_2,k_3)\exp(Ik_3x_3). \tag{37}$$

For HTSF only, we phase shift $\check{\phi}(k_1,k_2,x_3)$ to account for the shear-periodic boundary conditions, as discussed in Section 2.1.1

$$\tilde{\phi}(k_1,k_2,x_3) \equiv \check{\phi}(k_1,k_2,x_3)\exp(-Ik_1\mathcal{S}tx_3). \tag{38}$$

The data in the second and third dimensions is then transposed using the P3DFFT column communicator, so that all the data from the second dimension is now contiguous. Next, $N_1 \times N_3$ one-dimensional complex-to-complex FFTs are performed on the second dimension

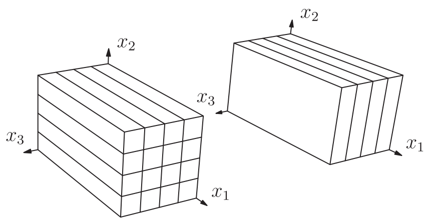

Fig. 6. A representative two-dimensional ('pencil') decomposition (left) and one-dimensional ('plane') decomposition (right).

$$\breve{\phi}(k_1,x_2,x_3) \equiv \frac{1}{N_2}\sum_{k_2}\tilde{\phi}(k_1,k_2,x_3)\exp(Ik_2x_2). \tag{39}$$

The data in the first and second dimensions is then transposed using the P3DFFT row communicator, after which $N_2 \times N_3$ complex-to-real one-dimensional transforms are performed on the first dimension, to obtain

$$\phi(x_1,x_2,x_3) = \frac{1}{N_1}\sum_{k_1}\breve{\phi}(k_1,x_2,x_3)\exp(Ik_1x_1). \tag{40}$$

The forward transform follows by analogy. Starting from a real-space variable $\phi(x_1, x_2, x_3)$, we first perform $N_2 \times N_3$ real-to-complex one-dimensional transforms on the first dimension

$$\breve{\phi}(k_1,x_2,x_3) = \sum_{x_1}\phi(x_1,x_2,x_3)\exp(-Ik_1x_1). \tag{41}$$

The data in the first and second dimensions is transposed using the P3DFFT row communicator, and $N_1 \times N_3$ complex-to-complex one-dimensional transforms are performed on the second dimension

$$\tilde{\phi}(k_1,k_2,x_3) = \sum_{x_2}\breve{\phi}(k_1,x_2,x_3)\exp(-Ik_2x_2). \tag{42}$$

We then perform a transpose of the second and third dimensions using the P3DFFT column communicator, and phase shift to account for the shear-periodic boundary conditions

$$\check{\phi}(k_1,k_2,x_3) = \tilde{\phi}(k_1,k_2,x_3)\exp(-Ik_1\mathcal{S}tx_3). \tag{43}$$

We complete the transformation by calling $N_1 \times N_2$ complex-to-complex one-dimensional transforms

$$\hat{\phi}(k_1,k_2,k_3) = \sum_{x_3}\check{\phi}(k_1,k_2,x_3)\exp(-Ik_3x_3). \tag{44}$$

5.2. Particle update

The scale separation between the largest and smallest turbulent flow features increases with the flow Reynolds number. Therefore, an increase in Reynolds number requires an increase in the number of Lagrangian particles, to ensure that the particles sample the flow with sufficient resolution. The flow Reynolds number we achieve is determined by both the number of grid points and the small-scale resolution $k_{\max}\,\eta$. From experience, for $k_{\max}\eta \approx 2$, we have found that a population of $(N/4)^3$ particles (where N is the number of grid points in each direction) for a given St provides a good balance between statistical convergence and computational expense.

At the start of a simulation, particles are placed randomly throughout the solution domain with a uniform distribution. These random initial positions are generated efficiently in parallel using version 2.0 of the SPRNG library [35]. Each particle is assigned to a processor based on its physical location in the flow. Some of the grid values of the fluid velocity needed for the interpolation discussed in Section 4 may reside outside of that processor's memory, however. We therefore pad the processor's velocity field with ghost cells, as illustrated in Fig. 7. The unshaded grid cells represent fluid velocities that are local to a given processor, and the shaded grid cells represent ghost cell values from adjacent processors. The cells are numbered to indicate the correspondence between standard grid cells and ghost cells. The ghost cell exchanges are performed using two-sided, nonblocking MPI. As the simulation progresses, particles travel throughout the simulation domain under the influence of the turbulent flow. Particles which cross processor boundaries are exchanged at the end of every time step, also using two-sided, nonblocking MPI.

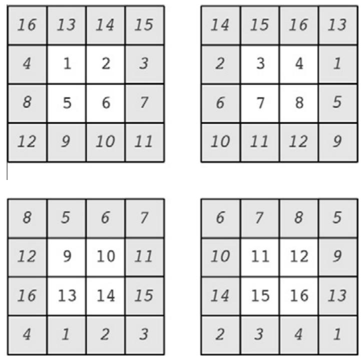

Fig. 7. Ghost cell communication for interpolation for a representative two-dimensional parallel domain decomposition over four processors. The unshaded grid cells are fluid velocities which are local to each processor, and the shaded grid cells are ghost cells from adjacent processors. The cells are numbered to indicate the correspondence between standard grid cells and ghost cells.

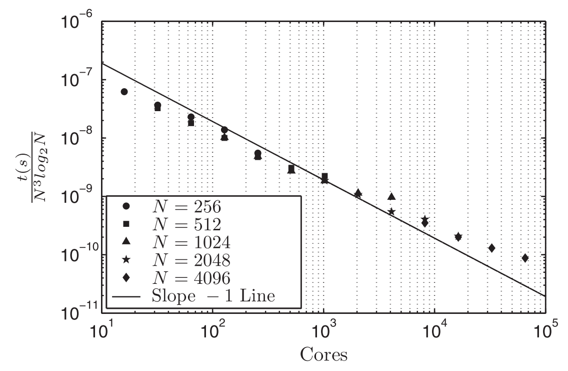

Fig. 8. Parallel scaling on ORNL Jaguar for grids of size N^3 with $(N/4)^3$ particles on different numbers of cores. The wall-clock time per step t is normalized by $N^3\log_2 N$, the expected scaling for a three-dimensional FFT.

5.3. Parallel scaling

In Fig. 8, we show timing data for simulations with a total of N^3 grid points and $(N/4)^3$ particles as a function of the number of cores. The wall-clock time per step t is normalized by $N^3\log_2 N$, the expected scaling for a three-dimensional FFT. The ideal scaling case (slope -1 line) is shown for comparison. All timings were performed on the computing cluster "Jaguar" at Oak Ridge National Laboratory (ORNL).

We achieve good scaling on ORNL Jaguar for the largest problem sizes. For a domain with 2048^3 grid points, for example, we observe 85% strong scaling when moving from 1024 processors to 4096 processors, and nearly 60% strong scaling on up to 16,384 processors.

6. Conclusion

We have presented a highly parallel, pseudospectral code ideally suited for the direct numerical simulation of particle-laden turbulence. HiPPSTR, the most efficient and general multiphase flow code of its kind, utilizes two-dimensional parallel domain decomposition, Runge–Kutta and exponential–integral time-stepping, and accurate and efficient B-spline velocity interpolation methods. All of these methods are selected and tuned for optimal performance on massively parallel architectures. HiPPSTR thus achieves good parallel scaling on $\mathcal{O}(10^4)$ cores, making it ideal for high-Reynolds-number simulations of particle-laden turbulence.

Acknowledgments

This article is a revised and expanded version of a conference proceeding for the Seventh International Conference on Computational Fluid Dynamics (ICCFD7) [36]. We are grateful to M. van Hinsberg and Professor P.-K. Yeung for helpful discussions during the code development. The authors acknowledge financial support from the National Science Foundation through Grant CBET-0967349 and through the Graduate Research Fellowship awarded to PJI. Additionally, the authors are indebted to several organizations for computational time used in developing and optimizing HiPPSTR. We would like to acknowledge high-performance computing support provided by the Extreme Science and Engineering Discovery Environment (supported by National Science Foundation Grant No. OCI-1053575), NCAR's Computational and Information Systems Laboratory (sponsored by the National Science Foundation), and the resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory (supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725).

References

[1] Falkovich G, Fouxon A, Stepanov MG. Acceleration of rain initiation by cloud turbulence. Nature 2002;419:151–4.
[2] Shaw RA. Particle–turbulence interactions in atmospheric clouds. Annu Rev Fluid Mech 2003;35:183–227.
[3] Warhaft Z. Laboratory studies of droplets in turbulence: towards understanding the formation of clouds. Fluid Dyn Res 2009;41. 011201–+.
[4] Malkiel E, Abras JN, Widder EA, Katz J. On the spatial distribution and nearest neighbor distance between particles in the water column determined from in situ holographic measurements. J Plankton Res 2006;28(2):149–70.
[5] Cuzzi JN, Hogan RC. Blowing in the wind I. Velocities of chondrule-sized particles in a turbulent protoplanetary nebula. Icarus 2003;164:127–38.
[6] Cuzzi JN, Davis SS, Dobrovolskis AR. Blowing in the wind II. Creation and redistribution of refractory inclusions in a turbulent protoplanetary nebula. Icarus 2003;166:385–402.
[7] Faeth GM. Spray combustion phenomena. Int Combust Symp 1996; 26(1):1593–612.
[8] Li WI, Perzl M, Heyder J, Langer R, Brain JD, Englemeier KH, et al. Aerodynamics and aerosol particle deaggregation phenomena in model oral-pharyngeal cavities. J Aerosol Sci 1996;27(8):1269–86.
[9] Pratsinis SE, Zhu W, Vemury S. The role of gas mixing in flame synthesis of titania powders. Powder Tech 1996;86:87–93.
[10] Moody EG, Collins LR. Effect of mixing on nucleation and growth of titania particles. Aerosol Sci Tech 2003;37:403–24.
[11] Balachandar S, Eaton JK. Turbulent dispersed multiphase flow. Annu Rev Fluid Mech 2010;42:111–33.
[12] Fede P, Simonin O. Numerical study of the subgrid fluid turbulence effects on the statistics of heavy colliding particles. Phys Fluids 2006;18. 045103–+.
[13] Ray B, Collins LR. Preferential concentration and relative velocity statistics of inertial particles in Navier–Stokes turbulence with and without filtering. J Fluid Mech 2011;680:488–510.
[14] Maxey MR, Riley JJ. Equation of motion for a small rigid sphere in a nonuniform flow. Phys Fluids 1983;26:883–9.
[15] Brucker KA, Isaza JC, Vaithianathan T, Collins LR. Efficient algorithm for simulating homogeneous turbulent shear flow without remeshing. J Comput Phys 2007;225:20–32.
[16] Rogallo RS. Numerical experiments in homogeneous turbulence. Tech. Rep. 81315; NASA; 1981.
[17] Pope SB. Turbulent flows. New York: Cambridge University Press; 2000.
[18] Orszag SA, Patterson GS. Numerical simulation of turbulence. New York: Springer-Verlag; 1972.
[19] Johnson RW. The handbook of fluid dynamics. CRC Press; 1998.
[20] Eswaran V, Pope SB. An examination of forcing in direct numerical simulations of turbulence. Comput Fluids 1988;16:257–78.
[21] Witkowska A, Brasseur JG, Juvé D. Numerical study of noise from isotropic turbulence. J Comput Acoust 1997;5:317–36.
[22] Elghobashi SE, Truesdell GC. On the two-way interaction between homogeneous turbulence and dispersed particles. i: Turbulence modification. Phys Fluids A 1993;5:1790–801.
[23] Sundaram S, Collins LR. A numerical study of the modulation of isotropic turbulence by suspended particles. J Fluid Mech 1999;379:105–43.

[24] Sundaram S, Collins LR. Collision statistics in an isotropic, particle-laden turbulent suspension I. Direct numerical simulations. J Fluid Mech 1997;335:75–109.
[25] Reade WC, Collins LR. Effect of preferential concentration on turbulent collision rates. Phys Fluids 2000;12:2530–40.
[26] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipies in Fortran. Cambridge: Cambridge University Press; 1999.
[27] Hochbruck M, Ostermann A. Exponential integrators. Acta Numer 2010;19:209–86.
[28] Balachandar S, Maxey MR. Methods for evaluating fluid velocities in spectral simulations of turbulence. J Comput Phys 1989;83:96–125.
[29] Berrut JP, Trefethen LN. Barycentric Lagrange interpolation. Siam Rev 2004;46:501–17.
[30] Lekien F, Marsden J. Tricubic interpolation in three dimensions. Int J Numer Methods Eng 2005;63(3):455–71.
[31] van Hinsberg MAT, Thije Boonkkamp JHM, Toschi F, Clercx HJH. On the efficiency and accuracy of interpolation methods for spectral codes. SIAM J Sci Comput 2012;34(4):B479–98.
[32] Pekurovsky D. P3DFFT: a framework for parallel computations of Fourier transforms in three dimensions. SIAM J Sci Comput 2012;34(4):C192–209.
[33] Frigo M, Johnson SG. The design and implementation of FFTW3. Proc IEEE 2005;93(2):216–31. http://dx.doi.org/10.1109/JPROC.2004.840301.
[34] Engineering and scientific subroutine library (ESSL) and parallel ESSL; 2012. <http://www-03.ibm.com/systems/software/essl/>.
[35] Mascagni M, Srinivasan A. Algorithm 806: SPRNG: a scalable library for pseudorandom number generation. ACM Trans Math Softw 2000;26(3):436–61.
[36] Ireland PJ, Vaithianathan T, Collins LR. Massively parallel simulation of inertial particles in high-Reynolds-number turbulence. In: Seventh international conference on computational fluid dynamics; 2012. p. 1–9.