Seventh International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, Hawaii, July 9-13, 2012 ICCFD7-1505

Massively parallel simulations of inertial particles in high-Reynolds-number turbulence

Peter J. Ireland*, T. Vaithianathan* and Lance R. Collins*
Corresponding author: pji22@cornell.edu

* Cornell University, Ithaca, NY, USA.

Abstract: 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 with inertial particles 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. By quantifying the errors in the numerical solution, we obtain optimal parameters for a given domain size and flow Reynolds number. We conclude by discussing upcoming large-scale parallel runs at new national supercomputing centers and important implications for the multiphase flow community.

Keywords: Computational Fluid Dynamics, Direct Numerical Simulation, Inertial Particles, Isotropic Turbulence.

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, 2, 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] and aerosol drug delivery systems [8]. 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, due in part to the huge separation of scales at intermediate or high Reynolds numbers.

Since inertial particle dynamics are strongly sensitive to the smallest scales in the flow [9], large-eddy simulation, which its associated small-scale modeling, has difficulties representing sub-filter particle dynamics accurately, including particle clustering, which is driven by the Kolmogorov scales. Consequently, our investigations rely on the full, direct numerical simulation (DNS) of the three-dimensional Navier-Stokes equations and the Maxey & Riley equation [10]. 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 stringent computational demands of resolving all relevant temporal and spatial scales.

To enable higher resolution simulations at even higher Reynolds numbers, we have developed a HIghly Parallel, Particle-laden flow Solver for Turbulence Research (HIPPSTR). HIPPSTR is capable of simulating inertial particle motion in homogeneous turbulent flows on tens, or even hundreds of thousands of processors.

This paper is organized as follows. In §2, we show the equations governing fluid and particle motion and the underlying assumptions of the flow solver. We discuss the time integration and interpolation techniques in §3 and §4, respectively, complete with an error analysis for optimizing code performance. §5 then describes the parallelization strategy. We introduce upcoming simulations with HIPPSTR in §6, and conclude in §7 by discussing research opportunities enabled by this code and future investigations we are planning.

2 Governing equations

2.1 Fluid update

The underlying flow solver is based on the algorithm presented in Ref. [11] and summarized below. Since our present investigations are limited to the case of isotropic turbulence, we present the solution methodology for this case only.

The governing equations for the flow of an incompressible fluid are the continuity and Navier-Stokes equations, here presented in rotational form

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

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

where u_i is the velocity vector (with magnitude u), ϵ_{ijk} is the alternating unit symbol, ω_i is the vorticity vector, p is the pressure, ρ is the density, and ν is the kinematic viscosity. This form of Navier-Stokes equations has only six nonlinear terms, as compared to nine in the traditional Navier-Stokes equation, which reduces the expense of computation and renders the solution method more stable.

As a model for homogeneous isotropic turbulence, we consider fully periodic flows; that is, we assume that the velocity field and the pressure are periodic in all three spatial directions. This allows us to express a variable $\phi(\mathbf{x}, t)$ in terms of a Fourier series,

$$\phi(\mathbf{x}, t) = \frac{1}{N_1 N_2 N_3}\sum_{\mathbf{k}} \hat{\phi}(\mathbf{k}, t)\exp\left(I k_i x_i\right), \tag{3}$$

where $\mathbf{k}$ denotes the wavenumber vector, N_1, N_2, and N_3 the number of grid points in the three Cartesian directions, and I the imaginary unit. $\hat{\phi}(\mathbf{k}, t)$ is the Fourier transform of $\phi(\mathbf{x}, t)$, denoted as $\mathcal{F}\{\phi(\mathbf{x}, t)\}$. We determine $\hat{\phi}(\mathbf{k}, t)$ using the discrete forward Fourier transform

$$\hat{\phi}(\mathbf{k}, t) = \sum_{\mathbf{x}} \phi(\mathbf{x}, t)\exp\left(-I k_i x_i\right). \tag{4}$$

Applying this transform to (1) and (2), we obtain

$$k_i \hat{u}_i = 0, \tag{5}$$

$$\left[\frac{\partial}{\partial t} + \nu k^2\right]\hat{u}_i = \left(-\delta_{im} + \frac{k_i k_m}{k^2}\right)\epsilon_{mjn}\mathcal{F}\left\{\omega_j u_n\right\}, \tag{6}$$

where $k^2 = k_i k_i$. Note that we have used the incompressibility of the flow, as specified by (5), to project the pressure out of (6) (refer to Ref. [11] for a detailed derivation).

Exact evaluation of the non-linear convolution sum $\mathcal{F}\{\omega_j u_n\}$ with a spectral method is prohibitively expensive computationally. These sums are therefore evaluated using a pseudospectral algorithm [12], whereby the velocity and vorticity are transformed from Fourier space to physical space, their product is computed in physical space, and then the result is transformed back into Fourier space. This transformation between physical and Fourier space takes place using three-dimensional fast Fourier transforms (FFT), the parallel implementation of which are described in §5. The nonlinear products in the pseudospectral method introduce aliasing errors which we eliminate by means of both spherical truncation and phase-shifting (refer to Appendix A of Ref. [13]).

2.2 Low-wavenumber forcing

Without external forcing, the turbulent kinetic energy would dissipate with time. To achieve statistically stationary turbulence, we introduce two different forms of low-wavenumber forcing that mimic turbulence generation in more realistic flows with mean velocity gradients.

2.2.1 Deterministic forcing scheme

The deterministic forcing scheme [14] first computes the amount of turbulent kinetic energy dissipated in a given time step Δt, $\Delta E_{\text{tot}}(\Delta t)$. This energy is restored at the end of each time step by reintroducing it into the forcing range between $k_{\text{f,min}}$ and $k_{\text{f,max}}$

$$\hat{\mathbf{u}}(\mathbf{k}, t+\Delta t) = \hat{\mathbf{u}}(\mathbf{k}, t+\Delta t)\sqrt{1+\frac{\Delta E_{\text{tot}}(\Delta t)}{\displaystyle\int_{k_{\text{f,min}}}^{k_{\text{f,max}}} E(k, t+\Delta t)dk}}\,, \tag{7}$$

where $E(k, t+\Delta t)$ represents the turbulent kinetic energy in a wavenumber shell with magnitude k at time $t+\Delta t$. We typically choose $k_{\text{f,min}}$ and $k_{\text{f,max}}$ to be in the low-wavenumber region of the spectrum, so that our forcing scheme does not contaminate the small-scale statistics, essential for simulating sub-Kolmogorov particles.

2.2.2 Stochastic forcing scheme

The alternative forcing scheme is based on a stochastic forcing function [15]. In this scheme, stationarity is achieved by introducing an artificial forcing function, $f_i(\mathbf{k}, t)$, into the Navier-Stokes equation. This function is non-zero only over the forcing range between $k_{\text{f,min}}$ and $k_{\text{f,max}}$. $f_i(\mathbf{k}, t)$ evolves by a vector-valued complex Ornstein-Uhlenbeck process [15] as shown below

$$df_i(k,t) = -\frac{f_i(k,t)}{T_f}dt + \sqrt{\frac{2\sigma_f^2}{T_f}}dW_i(k,t), \quad \forall k \ni k_{\text{f,min}} < k \leq k_{\text{f,max}} \tag{8}$$

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

$$\langle dW_i dW_j^* \rangle = \Delta t \delta_{ij}. \tag{9}$$

While implementing on a computer, we set the increment $dW_i = (\alpha_i + I\beta_i)\sqrt{\Delta t}$, where α_i and β_j are two independent $\mathcal{N}(0,1)$ random numbers.

2.3 Particle update

The Maxey & Riley equation [10] 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 $Re_p \equiv \frac{||\mathbf{u}(\mathbf{X})-\mathbf{v}||d}{\nu} < 1$. Here, $\mathbf{u}(\mathbf{X})$ denotes the undisturbed fluid velocity $\mathbf{u}$ of a particle with center position $\mathbf{X}$, and $\mathbf{v}$ denotes the particle velocity. Under these assumptions, the Maxey & Riley equation simplifies to a system of ordinary differential equations for the position and velocity of a given particle

$$\frac{d\mathbf{X}}{dt} = \mathbf{v}, \tag{10}$$

$$\frac{d\mathbf{v}}{dt} = \frac{\mathbf{u}(\mathbf{X})-\mathbf{v}}{\tau_p} + \mathbf{g}, \tag{11}$$

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 (10) and (11) 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 §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 [16, 17], 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 [18].

3 Time integration

3.1 Fluid update

To calculate the temporal evolution of the fluid, we introduce the integrating factor $e^{\nu k^2 t}$ to re-express (6) as

$$\frac{\partial}{\partial t}\left[\hat{u}_i e^{\nu k^2 t}\right] = e^{\nu k^2 t}\left(-\delta_{im} + \frac{k_i k_m}{k^2}\right)\epsilon_{mjn}\mathcal{F}\left\{\omega_j u_n\right\}. \tag{12}$$

Integrating (12) from time t_0 to time $t_0 + \Delta t$, we obtain

$$\hat{u}_i(t_0+\Delta t) = \hat{u}_i(t_0)e^{-\nu k^2 \Delta t} + e^{-(t_0+\Delta t)\nu k^2}\left(-\delta_{im} + \frac{k_i k_m}{k^2}\right)\epsilon_{mjn}\int_{t_0}^{t_0+\Delta t} e^{\nu k^2 t}\mathcal{F}\left\{\omega_j u_n\right\} dt. \tag{13}$$

A second-order Runge-Kutta method (RK2) is used to approximate the integral in (13), yielding

$$\int_{t_0}^{t_0+\Delta t} e^{\nu k^2 t}\mathcal{F}\left\{\omega_j u_n\right\} dt \approx \frac{\Delta t}{2}\left[e^{\nu k^2 t_0}\mathcal{F}\left\{\omega_j(t_0)u_n(t_0)\right\} + e^{\nu k^2 (t_0+\Delta t)}\mathcal{F}\left\{\omega_j(t_0+\Delta t)u_n(t_0+\Delta t)\right\}\right]. \tag{14}$$

Therefore, we simplify (13) and obtain

$$\begin{aligned}\hat{u}_i(t_0+\Delta t) &= \hat{u}_i(t_0)e^{-\nu k^2 \Delta t} + \frac{\Delta t}{2}\left(-\delta_{im} + \frac{k_i k_m}{k^2}\right)\epsilon_{mjn} \\ &\times \left[e^{-\nu k^2 \Delta t}\mathcal{F}\left\{\omega_j(t_0)u_n(t_0)\right\} + \mathcal{F}\left\{\omega_j(t_0+\Delta t)u_n(t_0+\Delta t)\right\}\right]. \end{aligned} \tag{15}$$

To avoid convective instabilities, the time step Δt is chosen to be sufficiently small to satisfy the Courant condition [19],

$$\frac{u_{\max}\Delta t\sqrt{3}}{\min(\Delta x_1, \Delta x_2, \Delta x_3)} \lesssim 0.5, \tag{16}$$

where $u_{\max}$ is the maximum velocity magnitude in the domain, and Δx_i is the grid spacing in the i^{th} coordinate direction.

3.2 Particle update

Inertial particles introduce another time scale into the simulation, namely the particle response time τ_p, which is independent of the fluid time step Δt. For the case where $\tau_p \ll \Delta t$, the system defined by (10) and (11) 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, since it is necessarily an iterative process involving multiple interpolations of the fluid velocity to the current particle location. The standard explicit Runge-Kutta method, under these circumstances, would require using a smaller time step for the particles than for the fluid, increasing the computational expense of the simulation. Additionally, inertial particle simulations may involve a spectrum of τ_p, which vary from processor to processor, causing the system 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 alternate second-order scheme that works well over the whole range of τ_p (particularly for $\tau_p \ll \Delta t$), while still using the fluid time step. The numerical scheme is based on 'exponential integrators' [20]. Exponential integrators are a broad class of methods that treat the linear term in (11) exactly and the inhomogeneous part using an exponential quadrature. Based on the results in Ref. [20], we pattern a second-order update of (11) as follows:

$$\mathbf{v}(t_0+\Delta t) = e^{-\Delta t/\tau_p}\mathbf{v}(t_0) + w_1\mathbf{u}[\mathbf{X}(t_0)] + w_2\mathbf{u}[\mathbf{X}(t_0)+\mathbf{v}(t_0)\Delta t] + (1-e^{-\Delta t/\tau_p})\tau_p\mathbf{g}, \tag{17}$$

where the weights w_1 and w_2 are given by

$$w_1 \equiv \left(\frac{\Delta t}{\tau_p}\right)\left[\varphi_1\left(-\frac{\Delta t}{\tau_p}\right) - \varphi_2\left(-\frac{\Delta t}{\tau_p}\right)\right], \ w_2 \equiv \left(\frac{\Delta t}{\tau_p}\right)\varphi_2\left(-\frac{\Delta t}{\tau_p}\right), \tag{18}$$

and

$$\varphi_1(z) \equiv \frac{e^z-1}{z}, \ \ \varphi_2(z) \equiv \frac{e^z-z-1}{z^2}. \tag{19}$$

It is important to note that as $\tau_p \to 0$, this explicit scheme recovers the correct limit for inertialess passive tracers. Over the range of τ_p and Δt generally considered in our simulations, we have found the second-order exponential integrator method to be more accurate than the standard RK2 and up to an order of magnitude faster. We discuss the errors incurred by the exponential integrator method below in §4.

4 Interpolation

As discussed in §2.3, the solution of (10) and (11) requires an interpolation of grid values of the fluid velocity to the particle centers. In principle, this interpolation could be done exactly (provided that the grid is sufficiently fine to capture all scales of motion) using a spectral interpolation [21]. 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 introduced several different interpolation methods into our code. They include linear, Lagrangian [22], Hermite [23], shape function [21], and B-spline interpolation [24]. The B-spline interpolation scheme, which is optimized for spectral simulations [24], provided the best trade-off between computational expense and accuracy.

The number of points used to construct the B-spline interpolant was varied between 4 and 10, and the RMS errors of the interpolated velocities were computed. The error analysis was performed on two representative grid sizes, 128^3 and 512^3. The small-scale resolution of the grids is specified by $k_{\max}\eta$ [25], where $k_{\max} = N\sqrt{2}/3$ is the maximum resolved wavenumber magnitude. For each grid size, we vary the small-scale resolution, from $k_{\max}\eta = 1$ (marginal small-scale resolution) to $k_{\max}\eta = 2$ (good small-scale resolution). In all cases, we determined the 'exact' values of the particle velocities from a spectral interpolation, and defined the velocity error as the RMS difference between the velocities obtained from spectral and B-spline interpolations at a given time.

To determine the optimal number of spline points, we compare the interpolation error to the local time-stepping error. Since we anticipate that the time-stepping error will be a function of the ratio $\Delta t/\tau_p$ (refer to §3.2), we choose two different values of τ_p for a fixed value of Δt: $\tau_p = 0.1\tau_\eta$ and $\tau_p = \tau_\eta$, where $\tau_\eta \equiv \sqrt{\nu/\epsilon}$ is the Kolmogorov time scale. (Note that the ratio τ_p/τ_η is usually called the Stokes number St of a particle.) To estimate the time-stepping error, we take the 'exact' result to be the particle velocities after ten velocity steps (using spectral interpolation) with a Courant number of 0.05. The 'approximate' result is the particle velocities after single time step (again using spectral interpolation) with a Courant number of 0.5.

The errors for each case are presented in figure 1, normalized by the RMS fluid velocity, u'. 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, thereby defining an optimal interpolation scheme for a given grid size and resolution.

5 Parallelization

5.1 Domain decomposition

The solution domain is decomposed using two-dimensional ('pencil') domain decomposition. Fast Fourier transforms (FFTs), required to advance the fluid velocity, are performed using the P3DFFT library [26]. P3DFFT uses standard FFT libraries (such as FFTW [27] and ESSL [28]) to compute the three-dimensional

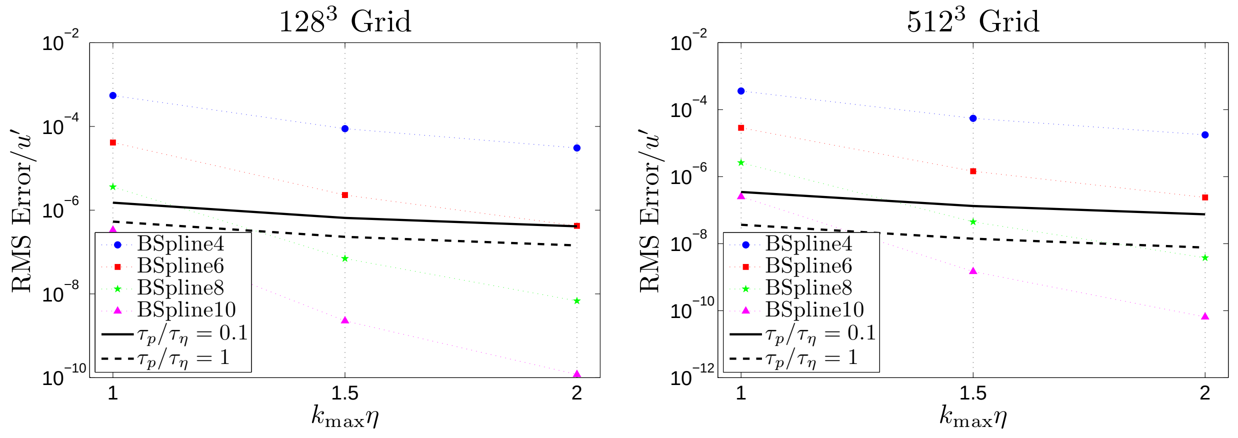

Figure 1: Interpolation error and time-stepping error for different numbers of spline points (4 to 10), particle response times ($\tau_p/\tau_\eta = 0.1, 1$), small-scale resolution ($k_{\max}\eta = 1, 1.5, 2$), and grid points ($128^3, 512^3$). All errors are normalized by the RMS fluid velocity, u'.

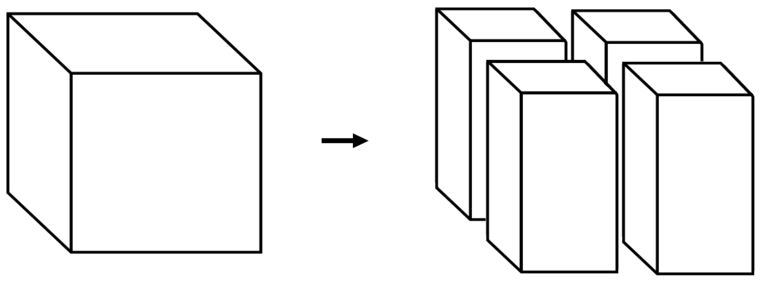

Figure 2: A representative two-dimensional decomposition. The domain (shown on the left) is decomposed into a series of 'pencils' (shown on the right) for parallelization over four processors.

FFT of the velocity or pressure distributed over the two-dimensional array of processors. A representative two-dimensional decomposition on four processors is shown in figure 2. HIPPSTR uses a distributed memory setup, and all necessary data are exchanged between processors using the Message Passing Interface (MPI).

A previous version of our 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. For large values of N, memory restrictions limited the grid sizes (and hence the Reynolds numbers) we were able to simulate. With one-dimensional domain decomposition, the largest simulation we could achieve was 1024^3 grid points, yielding only modest Reynolds numbers $R_\lambda \lesssim 400$, where $R_\lambda \equiv 2E_{\text{tot}}\sqrt{5/(3\nu\epsilon)}$ and E_{tot} is the turbulent kinetic energy.

Two-dimensional domain decomposition allows up to N^2 processors for a simulation with N^3 grid points. This new parallelization strategy essentially eliminates the memory limitations and increases the range of Reynolds numbers we can achieve. We discuss upcoming high-Reynolds-number simulations using HIPPSTR in §6.1.

5.2 Parallel scaling

In figure 3, 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 processors M. 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,

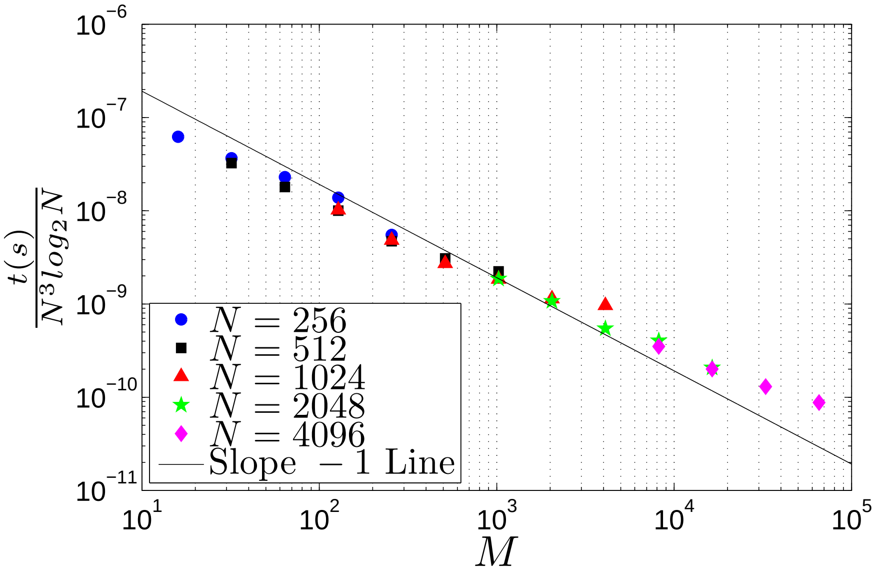

Figure 3: Parallel scaling on ORNL Jaguar for grids of size N^3 with $(N/4)^3$ particles on M processors. The wall-clock time per step t is normalized by $N^3 log_2 N$, the expected scaling for a three-dimensional FFT.

Table 1: Computing requirements for an upcoming simulation on NCAR Yellowstone using HIPPSTR.

Grid Points	2048^3
Particles	2.5 billion
R_λ	650
Cores	16,384
Core-hours	19 million
Duration	49 days

and nearly 60% strong scaling on up to $16,384$ processors.

6 Future work

Our primary goals for HIPPSTR are twofold: to perform massively parallel simulations at national supercomputing centers, and to make these simulation results publicly available.

6.1 Upcoming massively parallel simulations

As early users on the supercomputer 'Yellowstone' at the National Center for Atmospheric Research (NCAR), we will perform a 2048^3 simulation with over two billion inertial particles on $16,384$ processors. This will be the largest such simulation ever performed in the United States and will provide unprecedented information about particle-turbulence interactions in atmospheric clouds. From the simulation results, we hope to better understand droplet dynamics in high-Reynolds-number turbulence and thereby answer open questions regarding cloud evolution and precipitation [1, 2, 3]. The computing requirements for this simulation are summarized in table 1.

6.2 Dissemination of simulation results

Since the simulation described in §6.1 is unfeasible for most researchers, we will make the results of our study publicly accessible via an internet database. Two possible options are to make the data available for direct download (analagous to the iCFDdatabase [29]), or for remote analysis through user-created scripts (analagous to the JHU turbulence database cluster [30, 31]).

7 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 $\mathcal{O}(10^5)$ cores, making it ideal for high-Reynolds-number simulations of particle-laden turbulence.

References

[1] G. Falkovich, A. Fouxon, and M. G. Stepanov. Acceleration of rain initiation by cloud turbulence. *Nature*, 419:151–154, 2002.

[2] R. A. Shaw. Particle-turbulence interactions in atmospheric clouds. *Annu. Rev. Fluid Mech.*, 35:183–227, 2003.

[3] Z. Warhaft. Laboratory studies of droplets in turbulence: towards understanding the formation of clouds. *Fluid Dyn. Res.*, 41:011201, 2009.

[4] E. Malkiel, J. N. Abras, E. A. Widder, and J. Katz. On the spatial distribution and nearest neighbor distance between particles in the water column determined from in situ holographic measurements. *Journal of Plankton Research*, 28(2):149–170, 2006.

[5] J. N. Cuzzi and R. C. Hogan. Blowing in the wind I. Velocities of chondrule-sized particles in a turbulent protoplanetary nebula. *Icarus*, 164:127–138, 2003.

[6] J. N. Cuzzi, S. S. Davis, and A. R. Dobrovolskis. Blowing in the wind II. Creation and redistribution of refractory inclusions in a turbulent protoplanetary nebula. *Icarus*, 166:385–402, 2003.

[7] G. M. Faeth. Spray combustion phenomena. *Int. Combust. Symp.*, 26(1):1593–1612, 1996.

[8] W. I. Li, M. Perzl, J. Heyder, R. Langer, J. D. Brain, K. H. Englemeier, R. W. Niven, and D. A. Edwards. Aerodynamics and aerosol particle deaggregation phenomena in model oral-pharyngeal cavities. *Journal of Aerosol Science*, 27(8):1269–1286, 1996.

[9] S. Balachandar and J. K. Eaton. Turbulent dispersed multiphase flow. *Annu. Rev. Fluid Mech.*, 42:111–133, 2010.

[10] M. R. Maxey and J. J. Riley. Equation of motion for a small rigid sphere in a nonuniform flow. *Phys. Fluids*, 26:883–889, 1983.

[11] K. A. Brucker, J. C. Isaza, T. Vaithianathan, and L. R. Collins. Efficient algorithm for simulating homogeneous turbulent shear flow without remeshing. *J. Comp. Phys.*, 225:20–32, 2007.

[12] S. A. Orszag and G. S. Patterson. *Numerical simulation of turbulence.* Springer-Verlag, New York, 1972.

[13] R. W. Johnson. *The Handbook of Fluid Dynamics.* CRC Press, 1998.

[14] A. Witkowska, J. G. Brasseur, and D. Juvé. Numerical study of noise from isotropic turbulence. *J. Comput. Acoust.*, 5:317–336, 1997.

[15] V. Eswaran and S. B. Pope. An examination of forcing in direct numerical simulations of turbulence. *Comput. Fluids*, 16:257–278, 1988.

[16] S. E. Elghobashi and G. C. Truesdell. On the two-way interaction between homogeneous turbulence and dispersed particles. i: Turbulence modification. *Phys. Fluids A*, 5:1790–1801, 1993.

[17] S. Sundaram and L. R. Collins. A numerical study of the modulation of isotropic turbulence by suspended particles. *J. Fluid Mech.*, 379:105–143, 1999.

[18] W. C. Reade and L. R. Collins. Effect of preferential concentration on turbulent collision rates. *Phys. Fluids*, 12:2530–2540, 2000.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in Fortran.* Cambridge University Press, Cambridge, 1999.

[20] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[21] S. Balachandar and M. R. Maxey. Methods for evaluating fluid velocities in spectral simulations of turbulence. *J. Comput. Phys.*, 83:96, 1989.

[22] J. P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *Siam Rev.*, 46:501–517, 2004.

[23] F. Lekien and J. Marsden. Tricubic interpolation in three dimensions. *International Journal for Numerical Methods in Engineering*, 63(3):455–471, 2005.

[24] M. A. T. van Hinsberg, J. H. M. ten Thije Boonkkamp, F. Toschi, and H. J. H. Clercx. On the efficiency and accuracy of interpolation methods for spectral codes. *SIAM J. Sci. Comput.*, 2012. in press.

[25] S. B. Pope. *Turbulent Flows.* Cambridge University Press, New York, 2000.

[26] http://code.google.com/p/p3dfft/.

[27] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, February 2005.

[28] http://www 03.ibm.com/systems/software/essl/.

[29] http://mp0806.cineca.it/icfd.php.

[30] E. Perlman, R. Burns, Y. Li, and C. Meneveau. Data exploration of turbulence simulations using a database cluster. *Supercomputing SC07, ACM, IEEE*, 2007.

[31] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink. A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9:31, 2008.