An Adaptive Particle Tracking Algorithm for Lagrangian-Eulerian Simulations of Dispersed Multiphase Flows

Wenjun Ge* and Ramanan Sankaran†
Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831

The Lagrangian-Eulerian method is broadly used for simulations of dispersed multiphase flows by solving the continuous phase in the Eulerian framework while treating the dispersed phase as point particles in a Lagrangian framework. The accuracy of the Lagrangian-Eulerian method largely depends on the number of computational particles tracked for the Lagrangian phase. In this study, an adaptive Lagrangian particle tracking algorithm is proposed to balance the statistical error and the computational cost by dividing and merging the particles according to the local particle statistics in the computational domain. The computational particles are considered as weighted sampling points of the particle probability density functions which represents a statistical equivalence of the Lagrangian phase. The algorithm is implemented as a part of a C++ library for Lagrangian particles, Grit, with performance portability to multi-core or many-core CPUs and Graphic Processing Unit architectures. The accuracy of the proposed algorithm with two different schemes are studied through a test problem with analytical solutions for both the particle number density and momentum source term. The results are used to evaluate the proposed algorithm as well as to validate the parallel implementation.

Introduction

The modeling of dispersed multiphase flow [1] is of importance to several engineering applications, such as combustion [2], material processing [3]. Dispersed multiphase flows are flows in which the dispersed phases are not materially connected [1] and modeling the real physical process of dispersed multiphase flows is a tremendous task. For example, in many spray combustion processes, the fuel sprays are injected into a pressurized combustion chamber at a very high speed, breaking up, evaporating and mixing with the carrier phase, while the carrier phase is usually strongly turbulent and reacting, which is challenging to model by itself.

There are generally two modeling approaches for the numerical simulations of the dispersed multiphase flows, i.e., the Eulerian-Eulerian (EE) method [4] and the Lagrangian-Eulerian (LE) method [5]. The EE method treats both the continuous and dispersed phases in the Eulerian framework while the LE method solves the carrier phase in the Eulerian framework and the dispersed phases in the Lagrangian framework. Traditionally, the LE method is considered to be more flexible and can be applied to both dilute and dense flows. In comparison, the EE method is more effective only when the bulk properties of the dispersed phase are more statistically representative. However, the accuracy of the LE method largely depends on the number of computational particles tracked so that significant computational resources are required for realistic simulations with large numbers of computational particles. Furthermore, since the number density of the computational particles is not always uniform across the computational domain for realistic applications, cells enclosing large number of computational particles tend to introduce smaller errors while cells having less particles result in larger errors.

For the numerical solution of the Eulerian phase, the discretization errors due to a finite number of spatial and temporal points are usually determined by the mesh resolution and the numerical discretization scheme. One can either use a finer mesh/time step or employ high-order spatial/temporal discretization schemes to reduce the numerical

*Postdoctoral Research Associate and Member AIAA
†Computational Scientist and Senior Member AIAA

errors. Contrary to the popularity of the LE method, the statistical errors due to a finite number of particles for the Lagrangian phase in the LE method has rarely been studied until recently [6, 7]. Generally, the statistical error is inversely proportional to the square root of number of computational particles enclosed by a computational cell if there is no bias error. When refining the mesh for the Eulerian phase, one inevitably suffers from larger statistical errors for the Lagrangian particles due to less computational particles for each computational cell. For the opposite condition, although a coarser mesh will reduce the statistical errors, larger discretization error for the Eulerian phase and the interpolation errors due to the coarse mesh will still decrease the accuracy of the particle tracking.

Another difficulty is introduced by the fact that in modern large scale simulations, the mesh is often nonuniform or even dynamical changing. For example, the adoption of Adaptive Mesh Refinement (AMR) [8, 9] has been becoming more and more popular recently for practical simulations. The concept of AMR is to adapt the mesh dynamically wherever necessary such as in highly turbulent regions, flame fronts or shocks, etc., where localized large gradients are required to be numerically captured by a local mesh refinement. For many applications, these refinement regions for the Eulerian phase may or may not coincide with the positions of the Lagrangian particles. As a consequence, the traditional LE method results in much larger statistical errors at these refinement regions when the same number of computational particles enters a refinement mesh from a coarser mesh. Unlike the plenty algorithms for AMR, the algorithms for adaptive particle tracking are still at the trial and error level and need to be further investigated due to its stochastic nature.

One method to overcome this difficulty is to consider these Lagrangian particles as weighted sampling points of the particle probability density functions (PDF) [10, 11] which represents a statistical equivalence of the real physical particles. A comprehensive particle PDF formulation has been introduced by Subramaniam [10] and Pai and Subramanian [12]. The consistent relation between the physical system and a statistical expression through the particle PDF formulation is well established. However, the studies that discusses the algorithms that implements the particle PDF based formulation are quite limited. Garg et al. [13] proposed a numerically convergent algorithm by maintaining a near-uniform number density of the computational particles based on the particle PDF formulation. However, their particle number control strategy is too simple and might have problems extending to more general problems. Doisneau et al. [14] has developed a semi-Lagrangian transport scheme to eliminate the stochastic noises from the traditional Lagrangian method leading to a promising solution method for dense sprays.

The aim of this study is to develop and validate an adaptive Lagrangian particle tracking algorithm which divides and combines computational particles at targeted regions based on the statistics of the local computational particles. The idea is to extract the statistical information of the particles within a specified computational cell or domain and then redistribute more particles with smaller weights while conserving the original particle PDF in the same cell or domain. The most important part is how the particles are redistributed when divided. All variables of each particle need to be recalculated by specific schemes which are the key factors for conserving the particle PDF regarding redistributing the particles. Garg et al. [13] chooses to preserve a minimum statistical equivalence by spreading the locations of the newly generated particles uniformly within the computational cell and keeping the same velocity as the annihilated particle. In this study, we focus on two basic extraction and redistribution schemes for different variables. One approach is to presume a particle PDF with certain parameters extracted from the existing particle data. For example, if a Gaussian distribution is presumed for certain variable of the particles, the mean and standard deviation need to be extracted from the particle samples before being divided. Then the newly-generated particles can be redistributed according to the Gaussian distribution with the same mean and standard deviation. The other approach is to directly obtain a particle PDF from the particle data numerically, which is less practical in large simulations.

The proposed algorithm is implemented as part of a Lagrangian particle tracking library, Grit [15]. Grit is a C++ library for Lagrangian particle tracking for massively hierarchical parallel simulations with performance portability to diverse architectures such as multi-core or many-core CPUs and Graphic Processing Unit (GPU). The performance portability is realized through Kokkos [16] library which provides top-level abstractions for arrays, memory spaces, execution spaces and basic parallel kernels. All the computations in this study are performed on an Nvidia Pascal P100 GPU. This study focuses on the accuracy of the adaptive particle tracking algorithm, so the parallel performance will not be discussed in detail.

The adaptive particle tracking algorithm is tested through a two-phase flow problem proposed by Garg et al. [13] to analyze the resulting statistic errors, as well as to validate the parallel implementation. To quantify the statistical errors and the effectiveness of different particle PDF schemes of the adaptive particle tracking algorithm, the calculated particle density n and momentum source $\mathbf{S}_m$ are compared to the analytical solutions.

Governing equations

The position equation and momentum equation for the dispersed phase in the Lagrangian coordinates are:

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \mathbf{u}_i \tag{1}$$

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = \frac{\mathbf{F}_i}{m_i}\,, \tag{2}$$

where $i = 1, 2, 3, \cdots, N(t)$ is the index of each physical particle and $N(t)$ is the total number of physical particles; $\mathbf{x}_i$ is the position of the ith particle with $\mathbf{u}_i$ being its velocity; m_i is the mass of the particle and is kept the same for all particles in this study; $\mathbf{F}_i$ represents the forces exerted on each particle and here only the drag force is considered:

$$\mathbf{F}_i = \frac{1}{2}\rho_f C_{D,i} A_i |\mathbf{U}_f - \mathbf{u}_i|(\mathbf{U}_f - \mathbf{u}_i)\,. \tag{3}$$

$\mathbf{U}_f$ is the velocity of the Eulerian phase at the particle location; A_i is the frontal area of the particle; $C_{D,i}$ is the drag coefficient according to the following empirical law:

$$C_{D,i} = \frac{24}{Re_i}\left(1 + \frac{1}{6}Re_i^{2/3}\right)\,, \tag{4}$$

where ρ_f is the density of the Eulerian phase at the particle location. The particle Reynolds number Re_i is defined as

$$Re_i = \frac{|\mathbf{U}_f - \mathbf{u}_i| d_i}{\nu_f} \tag{5}$$

with ν_f being the local kinematic viscosity of the Eulerian phase and d_i being the particle diameter. A particle momentum relaxation time is often used which reduces Eq. (2) to

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = \frac{(\mathbf{U}_f - \mathbf{u}_i)}{\tau_{p,i}}\,, \tag{6}$$

τ_p characterizes the relaxation time for a particle to respond to the change of flow field. Another parameter with physical significance is the Stokes number defined as $\mathrm{St} = \tau_p/\tau_f$, where τ_f is the a characteristic time scale of the Eulerian phase.

A momentum source term for the Eulerian momentum equation from the Lagrangian phase is collected from the particles by

$$\mathbf{S}_m = -\frac{1}{\Delta V}\sum_{i=1}^{N_l}\beta_i \mathbf{F}_i\,, \tag{7}$$

for two-way coupling calculations, where ΔV is the volume of the computational cell enclosing the nearby N_l particles and β_i is the deposition ratio from the ith particle to the grid node at $\mathbf{X}$.

In practice, computational particles are employed instead of real particles to save computational resources. Each computational particle represents a group of physical particles with the same properties. The ratio between real and computational particles is denoted as the weight w_i and the total number of particles are conserved if

$$\sum_{i=1}^{N_c} w_i = N\,, \tag{8}$$

where N_c is the total number of computational particles. Accordingly, Eq. (7) becomes

$$\mathbf{S}_m = -\frac{1}{\Delta V}\sum_{i=1}^{N_{c,l}} w_i \beta_i \mathbf{F}_i\,. \tag{9}$$

For the traditional LE method [5], the weight w_i is a constant. In the perspective of statistical formulations based on the particle PDF, the weight w_i can also be regarded as the sampling weight of the particle PDF. The position and velocity equations for computational particles with a changing weight are the same as Eq. (1) and (2) except now it is looping over N_c notional particles with changing weights.

The corresponding evolution equation for the particle PDF $f(\mathbf{x}, \mathbf{u}, t)$ is given by

$$\frac{\partial f}{\partial t} + \frac{\partial u_k f}{\partial x_k} + \frac{\partial}{\partial u_k}\left(\frac{F_k}{m} f\right) = 0\,, \tag{10}$$

where

$$f(\mathbf{x}, \mathbf{u}, t) = \sum_{i=1}^{N_c} w_i \delta(\mathbf{x} - \mathbf{x}_i)\delta(\mathbf{v} - \mathbf{v}_i)\,, \tag{11}$$

which integrates to the total number of particles N. The joint particle PDF described by Eq. (11) has a very large phase space for all possible values so that it is impractical to directly design an algorithm based on it. Instead, we focus on reconstructing the PDF of each variable for the group of particles within the computational cell/domain.

In the current study, an adaptive algorithm is proposed to evolve the number density of the sampling points n_c and the weight w_i, by dividing and combining the computational particles.

The procedure of the adaptive algorithm is described as the following:

(1) Designate the computational cells that need to divide or combine particles. The cells can be manually selected or depend on some parameters like grid resolution, turbulence length scale, sampling point density, etc;

(2) If the computational particles need to be divided, extract the statistical information from the existing computational particles. The extracted information can be the parameters for a presumed PDF or a directly collected PDF for different variables;

(3) Dividing each particle into $N_{dv,l}$ particles: Deactivate the particles that are to be divided and initialize new particles with a weight of $w_i/N_{dv,l}$ within each computational cell/domain according to the extracted particle PDF;

(4) Combining $N_{cb,l}$ particles: Deactivate $N_{cb,l}$ particles and apportion the weight and other conserved variables to the remaining particles $N_{rm,l}(= N_c - N_{cb,l})$ in the computational cell/domain;

(5) Check the consistency of weight if necessary;

(6) Evolve all the particles (Solve the ordinary differential equations).

The combining scheme is simply an even redistribution of weight and other conserved scalars. The weight and other conserved scalars are updated as:

$$w_i' = w_i + \frac{\sum_{j=1}^{N_{cb,l}} w_j}{N_{rm,l}} \tag{12a}$$

$$\phi_i' = \frac{1}{w_i'}\left(w_i\phi_i + \frac{\sum_{j=1}^{N_{cb,l}} w_j\phi_j}{N_{rm,l}}\right), \tag{12b}$$

where $i = 1, 2, 3, \cdots, N_{rm,l}$ is the index for the remaining computational particles within the domain/cell. The first moment of the particle PDF, or the mean, is conserved.

For the particle dividing algorithm, two basic particle PDF extraction and redistribution schemes will be described, i.e., the presumed PDF scheme and the directly collected PDF scheme. The Gaussian distribution is chosen for the presumed PDF scheme.

For the presumed Gaussian PDF scheme, the mean and the standard deviation for the variable of existing particles at the targeted computational cells/domains are calculated. The mean (or ensemble average) μ and the standard deviation σ for a cell or domain is calculated from the local computational particles as:

$$\mu = \langle\phi\rangle = \frac{\sum_{i=1}^{N_{c,l}} w_i\phi_i}{\sum_{i=1}^{N_{c,l}} w_i} \tag{13a}$$

$$\sigma = \sqrt{\frac{N_{c,l}}{(N_{c,l}-1)}\frac{\sum_{i=1}^{N_{c,l}} w_i(\phi_i - \langle\phi\rangle)^2}{\sum_{i=1}^{N_{c,l}} w_i}}\,. \tag{13b}$$

Let ψ be the sampling-space variable and assume the variable ϕ_i follows a Gaussian distribution:

$$f(\psi; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\psi-\mu)^2}{2\sigma^2}}\,. \tag{14}$$

Integrating the probability density function, we got the corresponding cumulative distribution function (CDF):

$$F(\psi) = \int_{-\infty}^{\psi} f(\phi; \mu, \sigma^2) d\phi = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{\psi - \mu}{\sqrt{2}\sigma}\right)\right], \tag{15}$$

where the error function is defined as

$$\mathrm{erf}(u) = \frac{2}{\sqrt{\pi}} \int_0^u e^{-t^2/2} dt \,. \tag{16}$$

The value of ϕ' for the newly generated particle can be found by

$$\phi' = \mu + \sqrt{2}\sigma \mathrm{erf}^{-1}(2R - 1), \tag{17}$$

where R is a random number uniformly distributed on [0.0, 1.0]. The presumed PDF scheme is more suitable for the cell-based division and for applications with large vorticities where the computational particles tend to get stranded by the eddies.

The other way is to directly numerically calculate the PDF of certain variable from the particle data first and then redistribute the particles according to the extracted PDF. Within a computational cell or domain, the PDF of certain variable ϕ_i bounded by ψ_{max} and ψ_{min} can be numerically extracted first. The minimum and the maximum for the sampling points are found by

$$\psi_{\max} = \max_{\phi \in \Omega}(\phi) \tag{18a}$$

$$\psi_{\min} = \min_{\phi \in \Omega}(\phi), \tag{18b}$$

where $\Omega = \{\phi_1, \phi_2, \cdots, \phi_{N_{c,l}}\}$ is the sampling space. After discretizing the range $[\psi_{min}, \psi_{max}]$ into a finite number (N_{int}) of intervals $\Delta\psi$, the PDF $f(\psi_j)$ is ready to be calculated numerically as

$$f(\psi_j) = \frac{\sum_{k=1}^{N_{c,j}} w_k \phi_k}{\sum_{i=1}^{N_{c,l}} w_i \phi_i}, \tag{19}$$

where the sum in the numerator is looping over the samples within the range of $[\psi_j, \psi_j + \Delta\psi]$. The CDF is obtained by numerically integrating the PDF as

$$F(\psi_j) = \sum_{j=1}^{N_{int}} f(\psi_j) \Delta\psi. \tag{20}$$

The values of $F(\psi_j)$ and ψ_j need to be stored for inverse calculation in the next step. When reconstructing the PDF, the new value ϕ' is calculated by

$$\phi' = F^{-1}(R). \tag{21}$$

The numerical inverse calculation of $F^{-1}(R)$ is by interpolation of the $F(\psi_j)$ and ψ_j table and R is a random number uniformly distributed on [0.0, 1.0]. The strength of the directly collected PDF scheme is in its flexibility. It can be employed either for a cell-based division or a larger-domain-based division. However, it can be seen that the directly collected PDF scheme is only accurate or efficient when the original PDF is smooth.

Implementation

The adaptive particle tracking algorithm described above is implemented in a performance portable particle tracking library, Grit. Grit is programmed with the top-level abstractions provided by Kokkos for multi-dimensional arrays, memory spaces, execution spaces and basic parallel kernels. All of the computational kernels are able to be accelerated with the on-node parallelism by GPUs or multi-core, many-core CPUs.

A two-level data structure is employed to store and operate on particles. Particles are grouped together in clouds, which are then managed as a dynamically linked list as shown in Fig. 1. The Cloud class is inherited from the list container in the C++ standard template library (STL). The addition and deletion of particles are accomplished at the cloud level while operations within each cloud are executed through parallel execution kernels on the devices. The list is periodically compacted by merging clouds with low occupancy to keep the data structures and storage requirement within bounds.

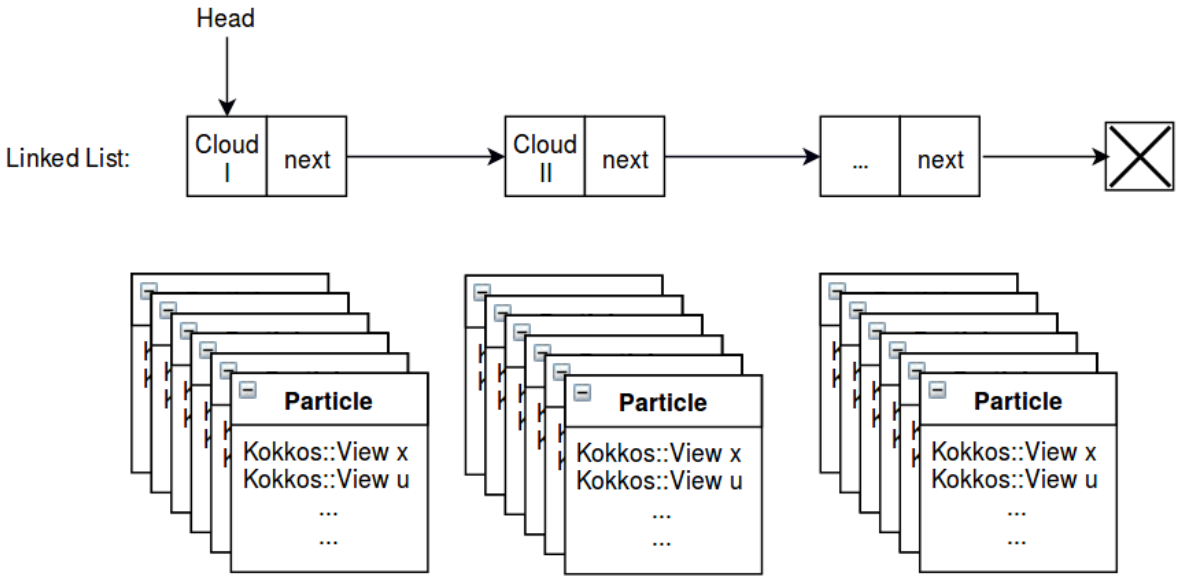

Fig. 1 Sketch of the two-level data structure of the particles in Grit.

For the current implementation, the size of the particle array in a cloud is statically allocated. A flag is used for each particle in the cloud to indicate whether the particle is active or inactive. For example, if $N_{c,l}$ active particles in a cloud are to be divided by a dividing ratio of $N_{dv,l}$. The $N_{c,l}$ active particles in the original cloud will first be deactivated. Then $N_{dv,l}$ new clouds will be inserted to the linked list. Each cloud will activate $N_{c,l}$ particles with the variables initialized and the weight being $1/N_{dv,l}$ of the original weight. At the end of the algorithm or a time step, the whole list will be compacted where the clouds with low occupancy will be merged together and the vacant cloud will be deleted.

The statistical calculations mentioned in this paper are programmed by parallel patterns. For example, counting particles with a certain flag can be done by the parallel reduce pattern, integrating a PDF can be efficiently accomplished by the parallel scan pattern, etc., which are provided by the Kokkos library. Yet, kernels for collecting values from multiple particles to a shared mesh node have to be based on the atomic operations because of potential race conditions.

Problem description: two-phase flow with steady-state gas velocities

In order to evaluate the accuracy of the adaptive particle tracking algorithm, a problem of two-phase dilute particle laden flow designed by Garg et al. [13] is studied. The Eulerian phase has a constant velocity field of

$$U_x(x, y) = U_0 \tag{22a}$$

$$U_y(x, y) = U_0 \left(1 - \frac{y}{L_y}\right), \tag{22b}$$

where $0 < x < L_x = L$ and $0 < y < L_y = L$. The particles enter the computational domain from the left boundary ($x = 0$) with a initial velocity of $(U_0, 0)$. Since the x–component of the gas phase velocity is the same as the initial velocity of the particles in x–direction, there is no relative motion between the particles and the gas phase in the x–direction. The two-phase flow is characterized by a Stokes number $St = \tau_f/\tau_p = 0.8$ with $\tau_f = L_x/U_0$ being the flow-through time. The particles with the same size and density are injected at $x = 0$ according to a specified volume fraction field:

$$\alpha(0, y) = \frac{\alpha_{\max} + \alpha_{\min}}{2} + \frac{\alpha_{\max} - \alpha_{\min}}{2} \sin(\frac{2\pi y}{L_y}), \tag{23}$$

with the $\alpha_{\max}$ and $\alpha_{\min}$ being 0.01 and 0.001, respectively. From the nonuniform distribution of the initial position of the computational particles, it is expected that there will be regions downstream that have less computational particles if

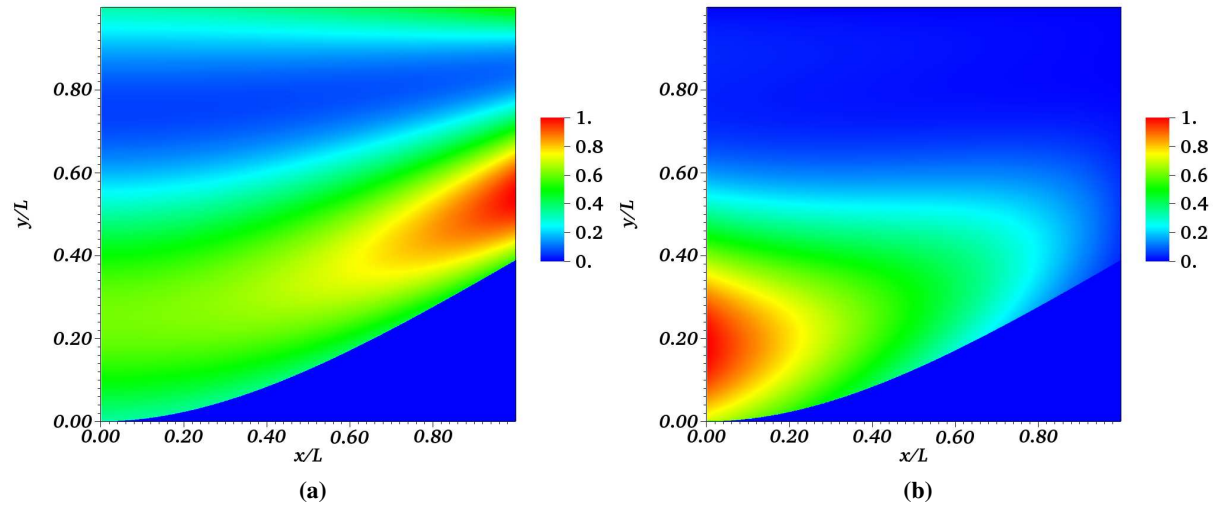

Fig. 2 Contour plots of the analytical solutions of (a) the normalized number density n^* and (b) the normalized y–component of the momentum source term $S^*_{m,y}$.

we choose to inject the particles with a same weight.

The analytical solutions in terms of normalized particle density $n^* = n/n_{\max}$ and the y–component of the normalized momentum source $S^*_{m,y} = S_{m,y}/S_{m,y,\max}$ are given by Garg et al. [13] and shown in Fig. 2. The upper region in Fig. 2(a) with smaller normalized particle density n^* is expected to have larger statistical errors for the traditional LE method since fewer particles are tracked locally.

The purpose of this test is to show the limitation of the traditional LE method and quantify the improvements from adaptive particle tracking algorithm of different schemes. A range of numerical simulations with different mesh sizes, particle numbers and different particle dividing schemes are studied to validate the algorithm and implementation discussed in this paper. All tests are conducted on a Nvidia Pascal 100 GPU. An explicit fourth-order six-stage Runge–Kutta method [17] is used for the time integration and a fourth-order Lagrange polynomial scheme is employed for the forward interpolation of velocity field from the Eulerian phase to the Lagrangian particles. The momentum deposition is by a Gaussian kernel, the deposition rate $\beta_i(\mathbf{X_j})$ in Eq. (7) and (9) from the particle i to the grid point $\mathbf{X_j}$ is calculated by

$$g_i(\mathbf{X}_j) = \exp\left(-\frac{|\mathbf{x}_i - \mathbf{X}_j|^2}{2h^2}\right) \tag{24a}$$

$$\beta_i(\mathbf{X}_j) = \frac{g_i(\mathbf{X}_j)}{\sum_{k=1}^{M_{g,l}} g_k}\,. \tag{24b}$$

For a 2-D 16-point stencil, each particle i located at $\mathbf{x_i}$ will deposit to a total number of $M_{g,l}$=16 grid nodes surrounding it with the area of $3\Delta x \times 3\Delta y$. The parameter h is a filter length scale which is set to 1 for this whole test. The normalization by Eq. (24b) is to conserve the momentum transfer, i.e., $\sum_{k=1}^{M_{g,l}} \beta_i(\mathbf{X}_k) = 1$. It is worth noting that this numerical deposition works as a filter so that unless the grid of a problem is infinitely fine and the number of particles tracked are infinitely large, the numerical solution of $S_{m,y}$ will not be as same as the analytical solution.

Results and Discussion

Numerical results from the traditional LE method in terms of relative errors from two cases with different mesh sizes and particle numbers are shown in Fig. 3. The relative errors of particle density and momentum source are calculated

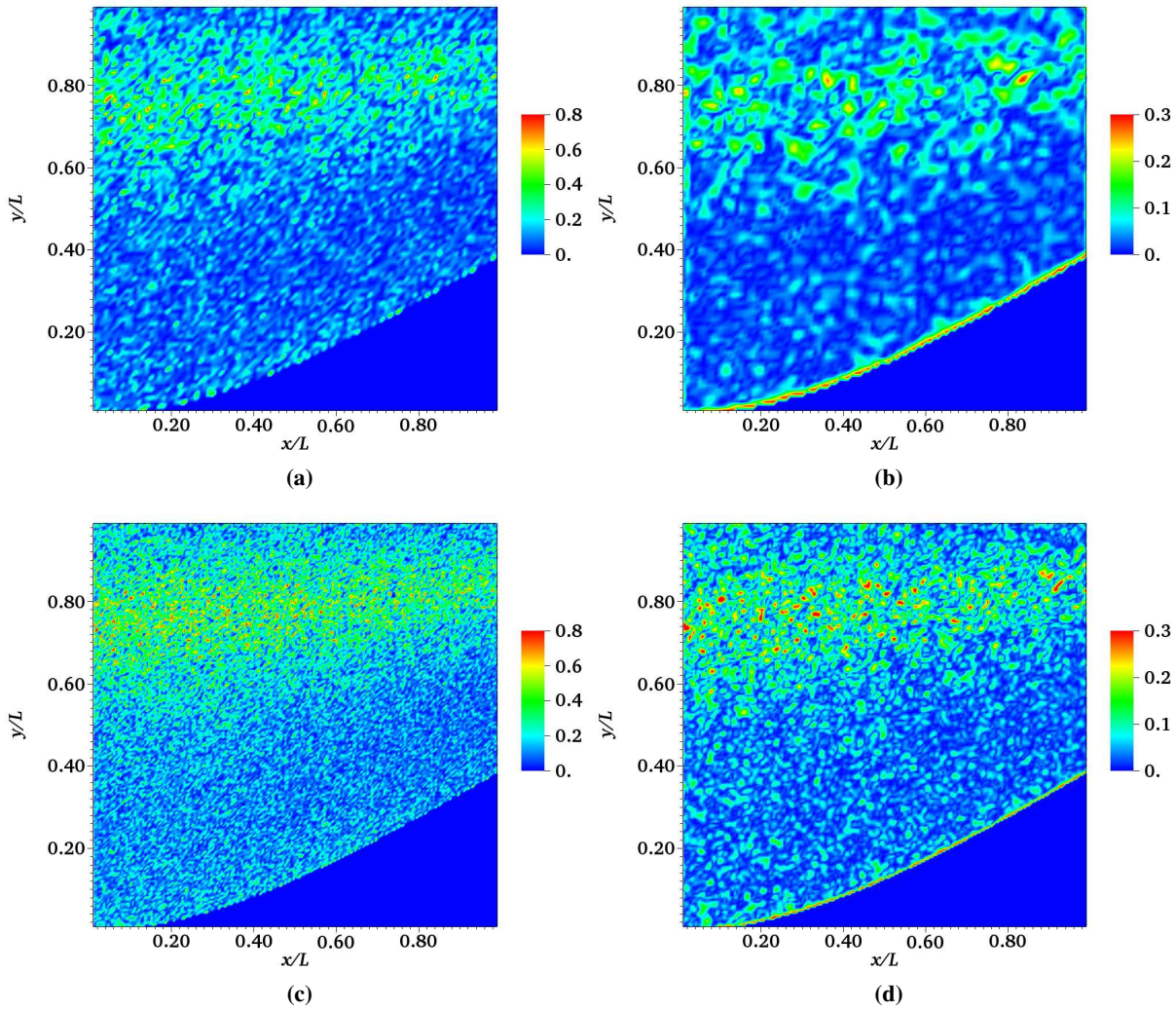

Fig. 3 Contour plots of relative error of number density ϵ_n (left) and the y–component of the momentum source term $\epsilon_{S_{m,y}}$ (right) for two cases: 100 × 100, 0.5 million particles (top); 200 × 200, 1 million particles (bottom).

from

$$\epsilon_n = \frac{n - n_{\text{analytical}}}{n_{\text{analytical}}} \tag{25a}$$

$$\epsilon_{S_{m,y}} = \frac{S_{m,y} - S_{m,y,\text{analytical}}}{S_{m,y,\text{analytical}}} \,. \tag{25b}$$

In the first case, Figs. 3(a) and 3(b), a total number of 0.5 million computational particles are tracked on a 100 × 100 uniform Cartesian mesh; In the second case, Figs. 3(c) and 3(d), a total number of 1.0 million computational particles are tracked on a 200 × 200 uniform Cartesian mesh. The time step $dt = 0.2\tau_f/N_x$ is scaled with the grid size N_x to ensure the consistency in the x–direction. At every time step, 1000 computational particles are injected according to the volume fraction, Eq. (23). The statistical error is mainly due to the uneven distribution of computational particles in the y–direction, and the number of sampling points on the y–direction are kept the same for both cases even though the total number of particles injected are different. The relative errors of particle density ϵ_n for these two cases are shown in Figs. 3(a) and 3(c) and the relative errors of the y–component of the momentum source $\epsilon_{S_{m,y}}$ are shown in Figs. 3(b) and 3(d). The scales of the relative errors (0 to 0.8 for ϵ_n and 0 to 0.3 for ϵ_{S_m}) are kept the same for the

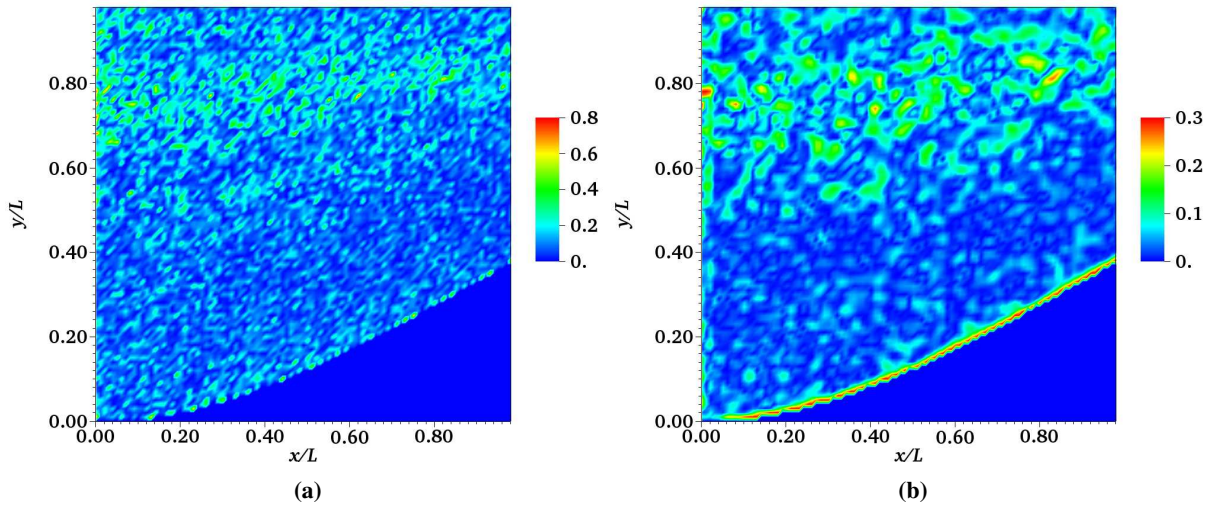

Fig. 4 Contour plots of (a) relative error of number density ϵ_n and (b) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 100 × 100, 0.5 million particles with Scheme I with each particle divided into 32 particles at ($x = 0.1, 0.5 < y < 1.0$).

ease of direct comparison. For both cases, the relative errors for both the particle density and momentum source are found to be nonuniform, with much larger relative errors at the upper region ($0.5L < y < L$). This is simply due to less computational particles being injected at the upper side of the inlet. Comparing the results between different meshes, the convergence problem of traditional LE method is visualized. The relative error for both the particle density and momentum source in the fine mesh (200×200) in the whole computational domain is significantly higher than that from the coarse mesh (100×100). In real applications, where the velocity of the Eulerian phase is numerical calculated, a convergence of numerical solutions for the Eulerian phase with respect to finer meshes is expected. However, the solution might not converge due to this large statistical errors of the Lagrangian phase for coupled cases. This is the fundamental motivation for the development of a better particle tracking algorithm.

To solve this problem, two different schemes for the adaptive particle tracking algorithm are to be tested. Both schemes are to divide the particles at the upper region ($0.5L < y < L$). For the first scheme (Scheme I), the positions of the newly generated particles are redistributed uniformly within the computational cell where the original particle resides. The y–component of the particle velocity u_y is calculated from the presumed Gaussian distribution as described in the section for governing equations.

The same simulation setup for the 100×100 mesh, 0.5 million particles case is employed except that the particles injected into the upper inlet are now divided at ($x = 0.1L, 0.5L < y < L$) into 32 particles with Scheme I. Surprisingly, the improvement is found to be very limited even with a large division ratio of 32 as shown in Fig. 4. For the relative error of the particle density n shown in Fig. 4(a), the Scheme I is found to be able to reduce the errors to a certain level. However, in terms of the relative error of the momentum source $S_{m,y}$ as shown in Fig. 4(b), there is almost no improvement. This indicates that the reconstructed velocity field from a presumed Gaussian distribution is not accurate. This can be seen from the velocity equation of the Eulerian phase. From Eq. (22b), it is clear that the velocity distribution of the particles should be monotonically decreasing with the increasing of y. So to presume a Gaussian distribution for u_y is not suitable for this example. Another reason for the ineffectiveness of Scheme I is because, the area of deposition kernel ($3\Delta x \times 3\Delta y$) is much larger than that of the computational cell used ($\Delta x \times \Delta y$).

Based on the analysis of the results of Scheme I, it is found that (1) the particles need to be redistributed in a larger domain other than within each cell; and (2) the velocity distribution for the newly-generated particles needs to be improved by using the information from the Eulerian phase. Thus in Scheme II, the following is employed: The PDF of y–location of the particles is extracted for a larger domain other than for each cell. Then the y–location of the newly-generated particles are initialized according to the stored PDF. The y–component of the particle velocity is presumed to be linear which is indicated by the distribution of U_y of Eulerian phase. The slope b and the intercept a are needed, which can be extracted from the particles through the weighted least squares method to minimize

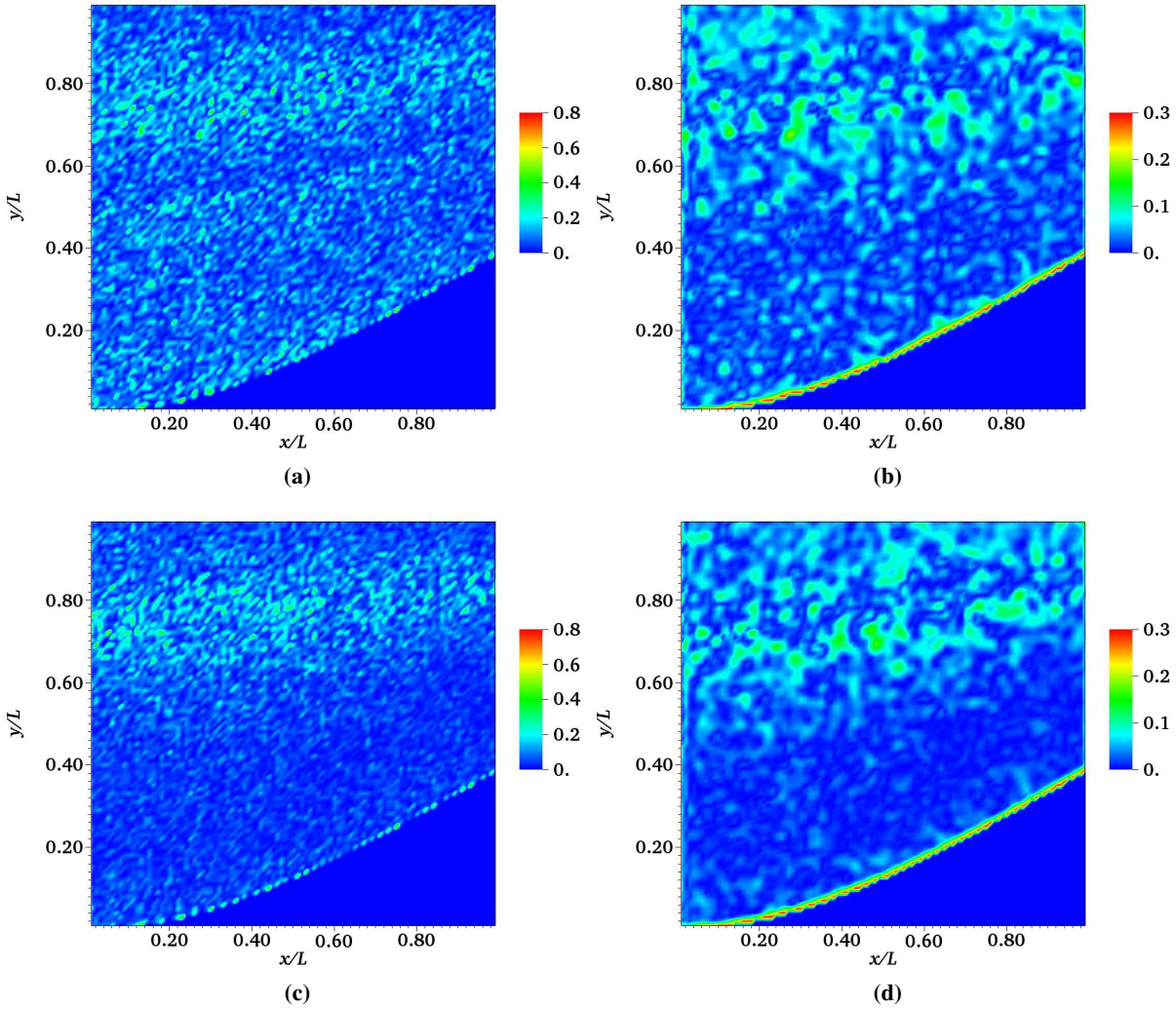

Fig. 5 Contour plots of (a) relative error of number density ϵ_n, (b) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 100×100, 0.5 million particles with Scheme II with each particle divided into 4 particles at $(x = 0.0, 0.5L < y < L)$, (c) relative error of number density ϵ_n, (d) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 100×100, 2.0 million particles with traditional LE method.

$\sum_1^{N_{c,l}} w_i[\phi_i - (a + bx_i)]^2$, i.e.,

$$b = \frac{\sum_{i=1}^{N_{c,l}} w_i x_i \phi_i - (\sum_{i=1}^{N_{c,l}} w_i x_i)(\sum_{i=1}^{N_{c,l}} w_i \phi_i)/\sum_{i=1}^{N_{c,l}} w_i}{\sum_{i=1}^{N_{c,l}} w_i x_i^2 - (\sum_{i=1}^{N_{c,l}} w_i x_i)^2/\sum_{i=1}^{N_{c,l}} w_i} \tag{26a}$$

$$a = \frac{\sum_{i=1}^{N_{c,l}} w_i \phi_i}{\sum_{i=1}^{N_{c,l}} w_i} - b\frac{\sum_{i=1}^{N_{c,l}} w_i x_i}{\sum_{i=1}^{N_{c,l}} w_i} . \tag{26b}$$

If the weight is a constant for all samples, Equation (26) reduces to the standard form [18]:

$$b = \frac{\sum_{i=1}^{N_{c,l}} x_i \phi_i - (\sum_{i=1}^{N_{c,l}} x_i)(\sum_{i=1}^{N_{c,l}} \phi_i)/N_{c,l}}{\sum_{i=1}^{N_{c,l}} x_i^2 - (\sum_{i=1}^{N_{c,l}} x_i)^2/N_{c,l}} \tag{27a}$$

$$a = \frac{\sum_{i=1}^{N_{c,l}} \phi_i}{N_{c,l}} - b\frac{\sum_{i=1}^{N_{c,l}} x_i}{N_{c,l}} . \tag{27b}$$

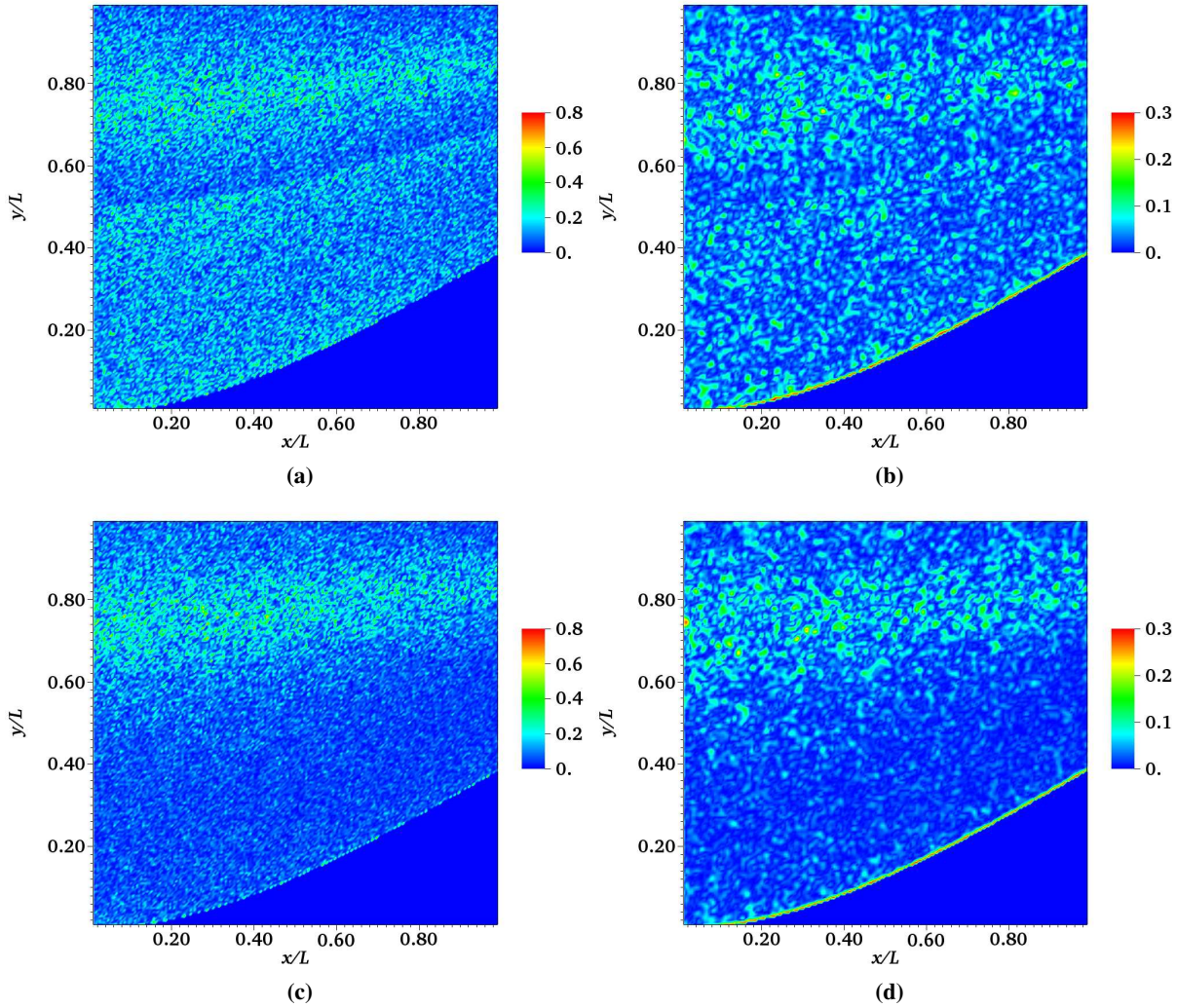

Fig. 6 Contour plots of (a) relative error of number density ϵ_n, (b) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 200×200, 1.0 million particles with Scheme II with each particle divided into 4 particles at $(x = 0.0, 0.5 < y < 1.0)$, (c) relative error of number density ϵ_n, (d) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 200×200, 4.0 million particles with traditional LE method.

The value of ϕ' for the newly generated particle can be found by

$$\phi' = bx + a\,, \tag{28}$$

x is the y-coordinates of the particle for this case.

The results with Scheme II for a domain of $(x = 0.0,\ 0.5L < y < 1.0L)$ with the division ratio of 4 for the 100×100 mesh are shown in Figs. 5(a) and 5(b). The computational particles are divided and redistributed in the whole region of $(x = 0.0, 0.5L < y < L)$ instead of at the cell basis. Only one PDF of y for this region is extracted and reconstructed at each time step. Compared with the results of the LE method without division shown in Figs. 4(a) and 4(b), the relative errors of particle density and momentum source are significantly reduced in the domain of $(0.5L < y < L)$ where the particles are divided after injection. The results for 100×100 mesh with 2 million particles are shown in Figs. 5(c) and 5(d). The relative errors in the upper region are reduced to the same level as that with four times more particles for the traditional LE method.

Similar comparison is made for the 200×200 mesh as shown in Fig. 6. The results with Scheme II for a division domain of $(x = 0.0, 0.5L < y < L)$ with the division ratio of 4 are shown in Fig. 6(a) and 6(b). The results for 200×200

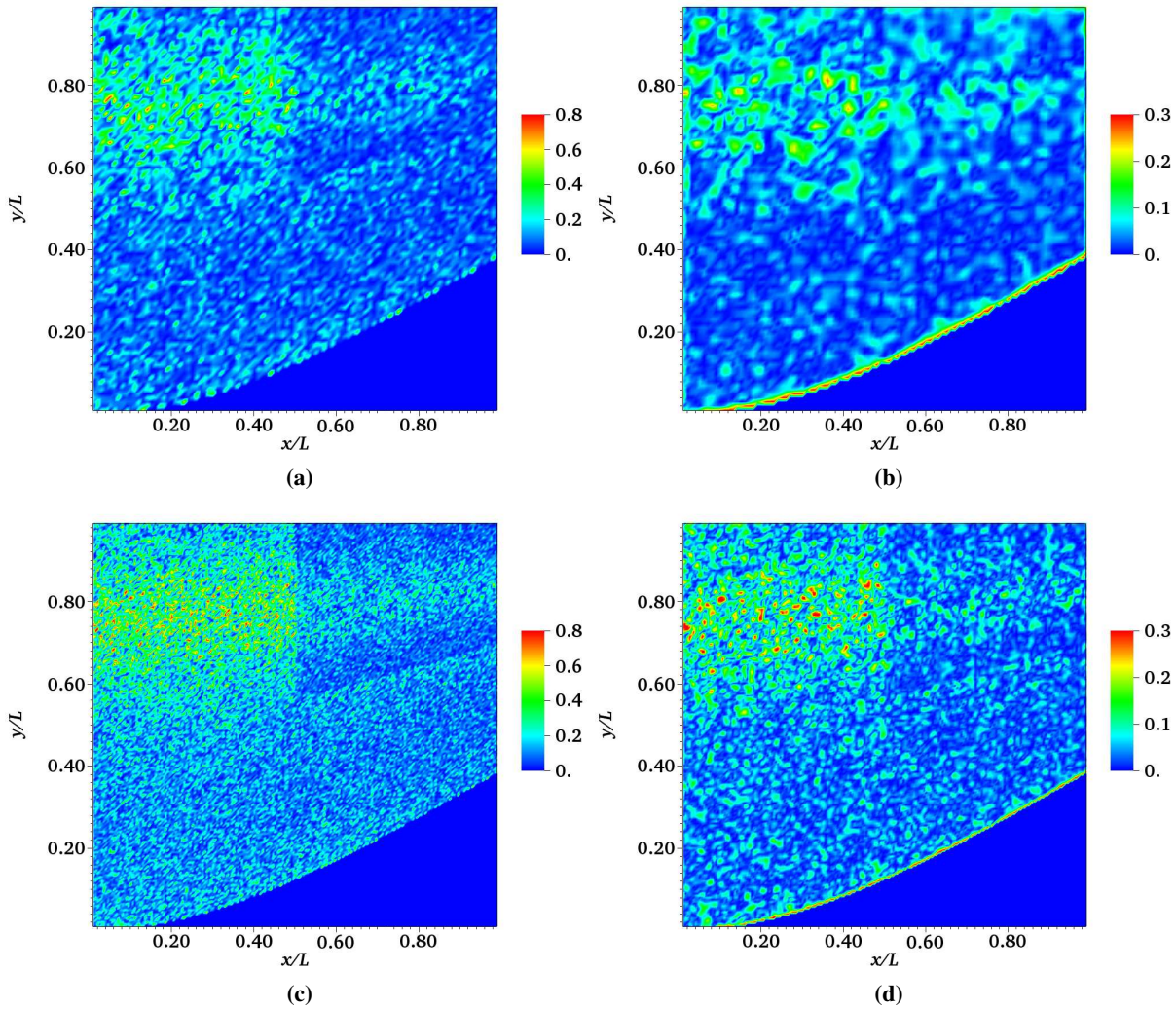

Fig. 7 Contour plots of (a) relative error of number density ϵ_n, (b) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 100 × 100, 0.5 million particles, (c) relative error of number density ϵ_n, (d) the y–component of the momentum source term $\epsilon_{S_{m,y}}$ for the case of 200 × 200, 1.0 million particles. Both are run with Scheme II with each particle divided into 4 particles at $(x = 0.5L, 0.56L < y < L)$**.**

mesh with 4 million particles are shown in Figs. 6(c) and 6(d). Comparing the results from the traditional LE method, Figs. 3(c) and 3(d) for the 200 × 200 mesh and the results with the adaptive particle tracking algorithm with Scheme II for the same mesh, Figs. 6(a) and 6(b), the larger statistical errors in the upper region of the computational domain due to a smaller sampling density n_c are significantly reduced. The results of the Lagrangian phase are converging with the refinement of the mesh when applying the adaptive particle tracking algorithm with Scheme II.

For the previous two cases, the particles are divided right after the injection with Scheme II. Since the particles are initialized with $u_y = 0$, the velocity of the redistributed particles are kept zero. The improvement is solely because of the redistribution of the particles in the y–direction according to the extracted PDF for y–location. For the next two cases, the division domain is then changed to $(x = 0.5L, 0.56L < y < L)$ so that the u_y needs to be redistributed with the presumed linear distribution of Scheme II. The results with Scheme II for a division domain of $(x = 0.5L, 0.56L < y < L)$ with the division ratio of 4 are shown in Figs. 7(a) and 7(b) for the 100 × 100 mesh and Figs. 7(c) and 7(d) for the 200 × 200 mesh. Both the ϵ_n and $\epsilon_{S_{m,y}}$ are much smaller in the domain of $(0.5L < x < L, 0.56L < y < L)$ for both meshes compared with that from the traditional LE method shown in Fig. 3. This proves that the local statistical error can be controlled if more computational particles can be reconstructed from the particle PDF. This adaptiveness of the

proposed algorithm is crucial for the potential applications to problems with AMR at the mesh refinement regions.

Conclusions

An adaptive particle tracking algorithm for reducing the statistical error of the LE method was developed and validated. The algorithm was based on locally reconstructing the particle PDF of the Lagrangian particles. Two schemes were employed for an example of a two-phase particle-laden flow. The effectiveness of these two schemes was quantified by directly comparing to the analytical solution. It was found that the choices of schemes for different variables are critical for the effectiveness of the algorithm. Although the theory of presenting the physical system with particle PDF formulation is well established, there are still many issues at the algorithm level. It is found that to repopulate particles at the cell level may not be always good if the source of the statistical error is rooted in conserving the particle PDF for a large sampling space covering a larger domain. Another observation is that the information of the Eulerian phase can be very helpful in determining the specific schemes or parameters that are to be used for the adaptive particle tracking algorithm. Future studies will be focused on further developing and applying the adaptive particle tracking algorithm to multiphase flows with AMR.

Acknowledgments

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

References

[1] Crowe, C. T., Schwarzkopf, J. D., Sommerfeld, M., and Tsuji, Y., *Multiphase Flows with Droplets and Particles*, 2nd ed., CRC Press, 2011.

[2] Jenny, P., Roekaerts, D., and Beishuizen, N., "Modeling of turbulent dilute spray combustion," *Progress in Energy and Combustion Science*, Vol. 38, No. 6, 2012, pp. 846–887.

[3] Chattopadhyay, K., "Multiphase Flows in Materials Processing: The Road Map for Modeling, Experimentation, Validation, and Optimization," *The Journal of The Minerals, Metals & Materials Society (TMS)*, Vol. 70, No. 10, 2018, pp. 2048–2050.

[4] Chapman, S., Cowling, T., and Burnett, D., *The Mathematical Theory of Non-Uniform Gases: an Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion and Diffusion in Gases*, Cambridge University Press, 1990.

[5] Sundaram, S., and Collins, L. R., "Numerical Considerations in Simulating a Turbulent Suspension of Finite-Volume Particles," *Journal of Computational Physics*, Vol. 124, No. 2, 1996, pp. 337–350.

[6] Are, S., Hou, S., and Schmidt, D. P., "Second-Order Spatial Accuracy in Lagrangian–Eulerian Spray Calculations," *Numerical Heat Transfer, Part B*, Vol. 48, No. 1, 2005, pp. 25–44.

[7] Garg, R., Narayanan, C., Lakehal, D., and Subramaniam, S., "Accurate Numerical Estimation of Interphase Momentum Transfer in Lagrangian–Eulerian Simulations of dispersed two-phase flows," *International Journal of Multiphase Flow*, Vol. 33, No. 12, 2007, pp. 1337–1364.

[8] Bangerth, W., Burstedde, C., Heister, T., and Kronbichler, M., "Algorithms and Data Structures for Massively Parallel Generic Adaptive Finite Element Codes," *ACM Transactions on Mathematical Software*, Vol. 38, No. 2, 2011, p. 14.

[9] Day, M., Tachibana, S., Bell, J., Lijewski, M., Beckner, V., and Cheng, R. K., "A Combined Computational and Experimental Characterization of Lean Premixed Turbulent Low Swirl Laboratory Flames: I. Methane Flames," *Combustion and Flame*, Vol. 159, No. 1, 2012, pp. 275–290.

[10] Subramaniam, S., "Statistical Modeling of Sprays Using the Droplet Distribution Function," *Physics of Fluids*, Vol. 13, No. 3, 2001, pp. 624–642.

[11] Fox, R. O., "Large-Eddy-Simulation Tools for Multiphase Flows," *Annual Review of Fluid Mechanics*, Vol. 44, 2012, pp. 47–76.

[12] Pai, M. G., and Subramaniam, S., "A Comprehensive Probability Density Function Formalism for Multiphase Flows," *Journal of Fluid Mechanics*, Vol. 628, 2009, pp. 181–228.

[13] Garg, R., Narayanan, C., and Subramaniam, S., "A Numerically Convergent Lagrangian-Eulerian Simulation Method for Dispersed Two-Phase Flows," *International Journal of Multiphase Flow*, Vol. 35, No. 4, 2009, pp. 376–388.

[14] Doisneau, F., Arienti, M., and Oefelein, J. C., "A Semi-Lagrangian Transport Method for Kinetic Problems with Application to Dense-to-Dilute Polydisperse Reacting Spray Flows," *Journal of Computational Physics*, Vol. 329, 2017, pp. 48–72.

[15] Ge, W., and Sankaran, R., "Grit: A Performance-Portable Particle Library for Lagrangian-Eulerian Simulation of Particle-Laden Turbulent Flow," Poster at 37th International Symposium on Combustion, Dublin, Ireland, 2018.

[16] Edwards, H. C., Sunderland, D., Porter, V., Amsler, C., and Mish, S., "Manycore performance-portability: Kokkos multidimensional array library," *Scientific Programming*, Vol. 20, No. 2, 2012, pp. 89–114. doi:10.3233/SPR-2012-0343.

[17] Kennedy, C. A., Carpenter, M. H., and Lewis, R. M., "Low-storage, Explicit Runge–Kutta Schemes for the Compressible Navier–Stokes Equations," *Applied Numerical Mathematics*, Vol. 35, No. 3, 2000, pp. 177–219.

[18] Kenny, J. F., and Keeping, E. S., "Linear Regression and Correlation," *Mathematics of Statistics*, Vol. 1, 1962, pp. 252–285.