Parallel variable-density particle-laden turbulence simulation

By H. Pouransari, M. Mortazavi AND A. Mani

1. Motivation and objectives

Variable density flows are ubiquitous in a variety of natural and industrial systems. Multi-phase flows in natural and industrial processes, astrophysical flows, and flows involved in combustion processes are such examples. For an ideal gas subject to low-Mach approximation, variations in temperature can lead to a non-uniform density field. In this work, we consider radiatively heated particle-laden turbulent flows. We have developed a parallel C++/MPI-based simulation code for variable-density particle-laden turbulent flows†. The code is modular, abstracted, and can be easily extended or modified.

We were able to reproduce well-known phenomena such as particle preferential concentration (Eaton & Fessler 1994), preferential sweeping (Wang & Maxey 1993), turbulence modification due to momentum two-way coupling (Squires & Eaton 1990; Elghobashi & Truesdell 1993), as well as canonical problems such as the Taylor-Green vortices and the decay of homogeneous isotropic turbulence.

This code is also able to simulate variable-density flows. Therefore, we were able to study the effect of radiation on the system. We reproduced one of our previous results, using a different code assuming constant density and Boussinesq approximation, on radiation-induced turbulence (Zamansky *et al.* 2014). We also reproduced the results of the homogeneous buoyancy-driven turbulence (Batchelor *et al.* 1992), where the Boussinesq approximation is relaxed. Moreover, using the Lagrangian-Eulerian capabilities of the code, many new physics have been studied by different researchers. These include spectral analyses of the energy transfer between different phases (Pouransari *et al.* 2014; Bassenne *et al.* 2015), subgrid-scale modeling of particle-laden flows (Urzay *et al.* 2014; Park *et al.* 2015), settling of heated inertial particles in turbulent flows (Frankel *et al.* 2014), new forcing models for sustaining turbulence in particle-laden flows (Bassenne *et al.* 2015), and comparisons between Lagrangian and Eulerian methods for particle simulation subject to radiation (Vié *et al.* 2015).

This code is developed to simulate a particle-laden turbulent flow subject to radiation. Each of these three elements can be removed from the calculation. In the simplest case, the code can be used as a homogeneous isotropic turbulence (HIT) simulator. We implemented two different geometrical configurations as explained in Section 3.2.

Particles are simulated using a point-particle Lagrangian framework. Particles are two-way coupled with fluid momentum and energy equations using linear interpolation and projection. The fluid is represented through a uniform Eulerian staggered grid. Spatial discretization is second-order accurate, and time integration has a fourth-order accuracy. The fluid is assumed to be optically transparent. However, particles can be heated with an external radiation source. Hot particles can conductively heat their surrounding fluid. Therefore, the fluid density field is variable. This leads to a variable coefficient Poisson

† `https://hadip@bitbucket.org/hadip/soleilmpi`

equation for hydrodynamic pressure of the flow. We have developed a novel parallel linear solver for the variable density Poisson equation that arises in the calculation. Governing equations and numerical methods are explained in Sections 2 and 3, respectively.

2. Physical model

In this section we provide a set of equations for the background flow and particles that are solved in the code. The primary variables for the flow are density (ρ), momentum (ρu_{f_i}), and temperature (T_f), and those for particles are location (x_p), velocity (u_{p_i}), and temperature (T_p).

Mass and momentum conservation equations for gas are as follows

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_{f_j}) = 0, \tag{2.1}$$

$$\frac{\partial}{\partial t}(\rho u_{f_i}) + \frac{\partial}{\partial x_j}(\rho u_{f_i} u_{f_j}) = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_{f_i}}{\partial x_j} + \frac{\partial u_{f_j}}{\partial x_i} - \frac{2}{3}\frac{\partial u_{f_k}}{\partial x_k}\delta_{ij}\right) + A\rho u_{f_i} + g_i(\rho - \rho_0) + \mathcal{P}\left(\frac{u_{p_i} - \mathcal{I}(u_{f_i})}{\tau_p}\right). \tag{2.2}$$

The last term in Eq. (2.2) represents the momentum transfer from the particles to the fluid. The operator $\mathcal{P}$ projects data from particle location to the fluid grid(per unit volume). $\mathcal{I}$ represents interpolation from the Eulerian grid to a particle location. In the current version of the code, both $\mathcal{P}$ and $\mathcal{I}$ are implemented using a linear approximation. The term $A\rho u_{f_i}$ corresponds to the linear forcing (Rosales & Meneveau 2005). g_i in $g_i(\rho - \rho_0)$ is the gravitational acceleration in the i^{th} direction after subtracting the static pressure. Last three terms in Eq. (2.2) can be switched on/off for different calculations. τ_p is the particle inertial relaxation time.

Conservation of energy for the gas phase is described by the following equation

$$\frac{\partial}{\partial t}(\rho C_v T_f) + \frac{\partial}{\partial x_j}(\rho C_p T_f u_{f_j}) = k\frac{\partial^2 T_f}{\partial x_j \partial x_j} + \mathcal{P}\left(2\pi D_p k\left(T_p - \mathcal{I}(T_f)\right)\right). \tag{2.3}$$

The last term in Eq. (2.3) represents the heat transfer from particles to the fluid. D_p is the particle diameter, and k is the gas thermal conductivity coefficient.

The thermodynamics of the system is governed by the ideal gas equation of state $\rho R T_f = P_0$, where P_0 is the thermodynamic pressure. Spatial pressure variation in the equation of state is neglected. This is justified by the low-Mach assumption.

A point-particle Lagrangian framework is used for particles; therefore, particle location is governed by the following equation

$$\frac{dx_{p_i}}{dt} = u_{p_i}. \tag{2.4}$$

Spherical particles with a small particle Reynolds number based on the slip velocity in a dilute mixture are considered. Assuming a large density ratio $\rho_p/\rho_f \gg 1$ and particle diameter smaller than the Kolmogorov length scale, the Stokes drag is the most important force on the particles (Maxey & Riley 1983). Hence, the dynamics of the particles is governed by the following equation

$$\frac{du_{p_i}}{dt} = -\frac{u_{p_i} - \mathcal{I}(u_{f_i})}{\tau_p}. \tag{2.5}$$

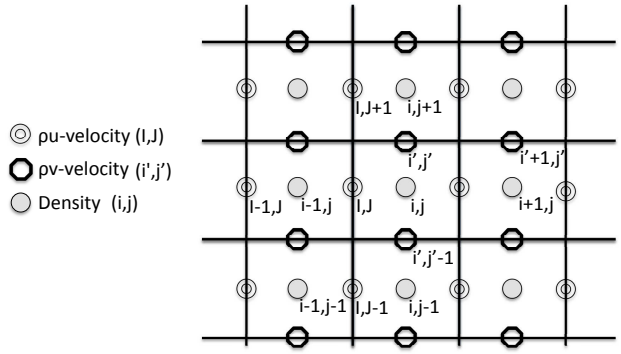

FIGURE 1. An illustration of a 2D uniform staggered grid, with momentum stored on cell faces, and scalar quantities at cell centers.

The following equation describes the conservation of energy for each particle;

$$\frac{d}{dt}(m_p C_{vp} T_p) = \frac{1}{4}\pi D_p^2 I \epsilon_p - 2\pi D_p k(T_p - \mathcal{I}(T_f)). \tag{2.6}$$

In the above equation, $m_p = \rho_p \pi D_p^3/6$ and C_{vp} are, respectively, particle mass and heat capacity. The first term in the right-hand side of Eq. (2.6) is the heat absorption by a particle with emissivity ϵ_p in an optically thin medium with uniform radiation intensity I. The second term represents heat transfer to the fluid.

3. Numerics

3.1. *Spatial discretization*

We have used a uniform three-dimensional staggered grid for the fluid in this code. A sketch of a two-dimensional uniform staggered grid is shown in Figure 1. Our discretization preserves the total mass, momentum, and kinetic energy. Therefore, for an inviscid flow, we enforce to conserve the total kinetic energy, as shown in Figure 2. The details of the energy conservative numerical method used in the code are explained in appendix B.

3.2. *Boundary and initial conditions*

This code supports two different configurations: 1) Triply periodic, which is a box with periodic boundary conditions (BC) in all directions. 2) Inflow-outflow, which is a duct with periodic BC in the $y-z$ plane, and convective BC in the x direction

$$\frac{\partial}{\partial t}(.) + U_x \frac{\partial}{\partial x}(.) = 0. \tag{3.1}$$

The inflow comes from an HIT simulation, sustained with linear forcing, which is running simultaneously with the duct simulation. At each time-step, the last slice of the HIT simulation is being copied to the first slice of the inflow-outflow simulation. The computational domain is shown in Figure 3.

The code accepts various initial conditions for particles and flow. In particular, we have provided a script to generate initial turbulence using a Passot-Pouquet spectrum (Passot & Pouquet 1987).

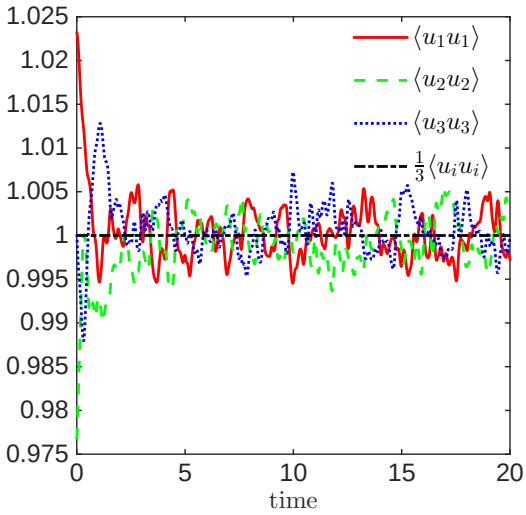

FIGURE 2. Conservation of the total kinetic energy for an inviscid calculation. Note that energy is being transferred from different components of the velocity, while the total kinetic energy is preserved.

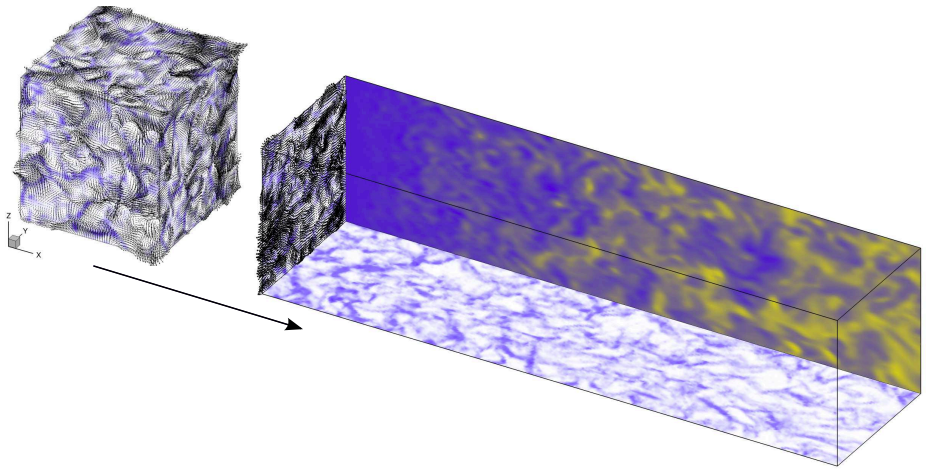

FIGURE 3. Inflow-outflow computational domain. Left: the triply periodic box used as a turbulence generator for the inflow-outflow domain (right). Particle concentration is shown on the sides of the box, and the bottom side of the inflow-outflow domain. Gas density contours are shown on the rear side of the inflow-outflow domain.

3.3. *Time advancement*

The time integration is performed using a fourth-order Runge Kutta (RK4) scheme for both particles and the flow. We first solve for momentum without considering the pressure gradient term. Using the divergence implied by the energy equation (note that the flow filed is not divergence free when radiation is on), we solve the variable coefficient Poisson equation and correct the fluid momentum field. The energy equation involves an energy transfer term between particles and fluid. Therefore, in order to accurately calculate the divergence condition for the fluid pressure, we need to have energy transfer from particles

at the next substep before solving the Poisson equation. In other words, at each step the right-hand side of the particle energy equation is calculated at the next substep. However, the right-hand sides of all other equations are evaluated at the current substep. Details of the numerical scheme are explained in appendix A.

3.4. *Poisson equation*

Solving for a variable density flow requires solving a variable coefficient Poisson equation for the hydrodynamic pressure as follows

$$\nabla . \left(\frac{1}{\rho} \nabla p \right) = f, \tag{3.2}$$

where f is determined using the energy equation. Consider $p = p_g + \delta p$, where p_g is our estimated value from the previous step. Then the above equation can be re-written as

$$\nabla^2 \delta p = \rho f - \rho \left(\nabla \frac{1}{\rho} \right) \cdot (\nabla p_g) - \nabla^2 p_g - \rho \left(\nabla \frac{1}{\rho} \right) \cdot (\nabla \delta p) \,. \tag{3.3}$$

All terms in the right-hand side of the above equation are known except for the last one. We ignore the last term and iterate to update $p_g \leftarrow p_g + \delta p$, until convergence. The convergence rate depends on the variation in the density field. Eq. (3.3) represents a constant coefficient Poisson equation (after ignoring the last term) that is solved using a parallel Fourier transform in periodic directions (Plimpton *et al.* 1997) and a parallel tri-diagonal solver in the non-periodic direction.

4. Software description

4.1. *Software Architecture*

In this section we briefly describe the significance and abilities of each class:

`params`: This class reads all input parameters from a file. After some initial calculations (before the simulation begins), all physical parameters can be accessed through this class.

`tensor0`: Stores a local 3D grid (i.e., for one process). Simple arithmetic, spatial discretization, interpolation, etc., are implemented in this class. Similarly, the class `tensor1` stores a local grid of 3D vectors. In other words, `tensor1` encompasses three instances of `tensor0`.

`gridsize`: stores and provides the logical information (e.g., size in each direction) for a local grid.

`proc`: has logical information (e.g., rank, neighbors, etc.) for each process.

`communicator`: takes care of updating halo cells for the grid by communicating between processors. All other parallel algorithms are implemented within this class.

`poisson`: solves the variable coefficient Poisson equation.

`particle`: holds information of N_p particles. Also, all particle-related algorithms are implemented in this class.

`grid`: represents the full grid, including all scalar and vector quantities for the flow.

Time integration loops are implemented in `simulation.cpp`. Note that with RK4, every quantity Q has four instances: Q (value at time-step n), Q_{int} (value at substep k), Q_{new} (value at substep $k+1$), and Q_{np1} (value at time-step $n+1$).

4.2. *Software Functionalities*

Using different classes defined in the code, one can easily compute various calculus operations (differentiations, interpolations, etc.) in parallel, without working directly with any

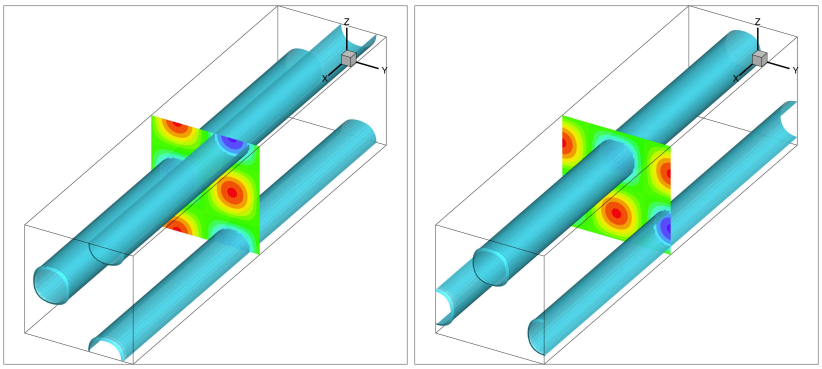

FIGURE 4. v contours and iso-surface (left); w contours and iso surface (right).

MPI-related command. Even though all parallel implementations are under the hood, they can be modified within the `communicator` class. The code is parallelized in three directions. The number of processes in each direction is determined in the `proc` class.

Flow and particle statistics can be computed at a desired frequency and stored as binary files (using parallel I/O). Numerical techniques are implemented completely modular; therefore, modifying the code to employ new numerics needs minimal effort (e.g., going from second-order to a fourth-order stencil).

After starting each simulation, a summary of information is stored in a file named `info.txt`. Simulation can be stopped, and all data be saved before the designated time by changing a flag in the file `touch.check`.

5. Verifications

5.1. *Decay of Taylor-Green vortices*

We have verified the carrier-flow implementation using the Taylor-Green vortices, an analytical solution to the Navier-Stokes equations. Having set the initial condition of the triply periodic box simulation as

$$\begin{aligned} u(x,y,z) &= U_x, \\ v(x,y,z) &= \sin(y)\cos(z), \\ w(x,y,z) &= -\cos(y)\sin(z), \end{aligned} \tag{5.1}$$

the velocity field will maintain its two-dimensional shape as shown in Figure 4.
We can also look at the decay rate of the Taylor-Green vortices in time. Starting with the initial velocity field given by the above equations, we expect an exponential decay rate for the turbulent kinetic energy $\sim e^{(2A-4\nu)t}$. Note that A is the linear forcing coefficient as in Eq. 2.2, which is only applied to the box simulation. A slice that enters the duct at time t_0 has turbulent kinetic energy (TKE) $\sim e^{(2A-4\nu)t_0}$. However, since there is no linear forcing in the duct, the decay rate of this slice scales like $e^{-4\nu t}$. In order to follow this decay we should consider the convective velocity in x direction, U_x (which is $U_x = 10$ in this case). It means that at time $t_0 + \Delta t$ the slice location is moved by $U_x \Delta t$. In other words, a slice located at position x has decayed from its enterance value for $\Delta t = x/U_x$ time units. Thus, by looking at one snapshot at an arbitrary time, t, we expect the following distribution for the TKE

$$\text{TKE} \sim e^{-4\nu\frac{x}{U_x}} \cdot e^{(2A-4\nu)t_0} = e^{-4\nu\frac{x}{U_x}} \cdot e^{(2A-4\nu)(t-\frac{x}{U_x})}.$$

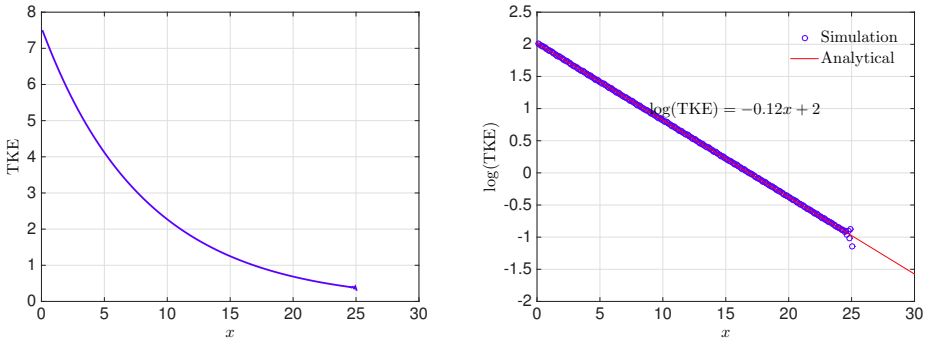

FIGURE 5. Decay of the TKE in x-direction (left). Linear fitting on the semi-log plane (right).

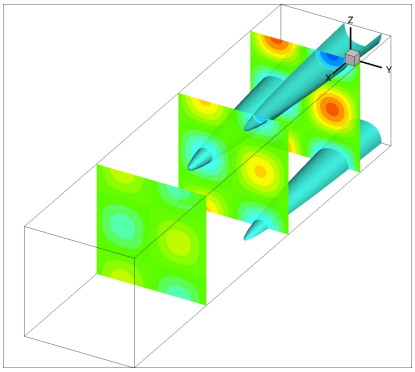

FIGURE 6. v contours and iso-surface

In the above expression, $e^{-4\nu\frac{x}{U_x}}$ is the decay that occurs in the duct, and $e^{(2A-4\nu)(t-\frac{x}{U_x})}$ is the TKE at the time at which this slice is copied from the box. Hence, TKE $\sim e^{\frac{2A}{U_x}x} = e^{\cdot 12x}$ (here, A=.6 is used). The decay of TKE is shown in Figure 5, and the v-component of the velocity iso-surface is illustrated in Figure 6. At the very end of the domain, due to a first-order accurate convective BC, there is a source of error, as is clear from Figure 5.

5.2. *Buoyancy driven turbulence*

Another sample problem, against which we tested the code is the homogeneous buoyancy-generated turbulence flow (Batchelor *et al.* 1992). In the original paper, the Boussinesq approximation is used, which is less accurate than a variable density calculation. The flow starts at rest, while the initial density field is non-homogeneous, and is drawn from an artificial spectrum. When gravity is present, the inhomogeneity of the density field results in the generation of turbulence. The turbulence decays eventually. These results are shown in Figure 7. The results are matched with those presented by (Batchelor *et al.* 1992) for smaller Reynolds numbers. For large Reynolds numbers (corresponding to larger variations in the gas density field) the results are slightly different. This can be explained by inaccuracy of the Boussinesq approximation when density variation is relatively large.

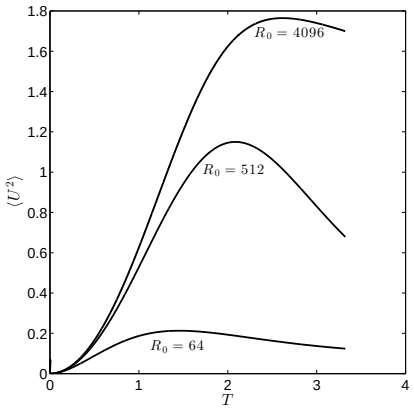

FIGURE 7. Generation of homogeneous buoyancy-driven turbulence. Non-dimensionalization is similar to that of the original paper by Batchelor *et al.* (1992).

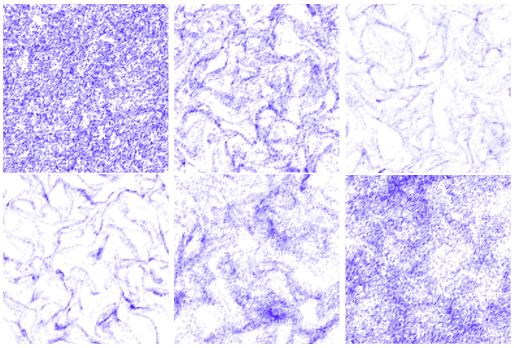

FIGURE 8. 2D slices of the three-dimensional particle-laden turbulence simulation for Kolmogorov-based Stokes number = 0.0625, 0.25, 0.5, 1, 4, and 8, respectively, from top left to bottom right. The slice thickness is 1/128 of the length of the box.

5.3. *Capturing particle preferential concentration*

In a particle-laden turbulence, the spatial distribution of the dispersed phase deviates from the homogeneous distribution (Eaton & Fessler 1994). This is due to the interaction of particles with turbulent eddies. Particle Stokes number (ratio of the particle relaxation time to the turbulent Kolmogorov timescale) is the relevant non-dimensional number that determines the inhomogeneity of particles. The highest preferential concentration is expected to appear for the Stokes number of order unity. In Figure 8, snapshots of particle distribution for different Stokes numbers are shown, confirming maximum preferential concentration at Stokes number 1. A quantitative analysis of particle preferential concentration is provided in (Pouransari & Mani 2015).

5.4. *Effect of domain partitioning*

We used various domain decompositions (for parallelization), and with the same initial condition, measured the difference between solution (momentum, density, particle location/velocity, etc.) after 10 time-steps. We found that solutions are equal up to machine precision. Some of the domain decompositions that we used are shown in Figure 9.

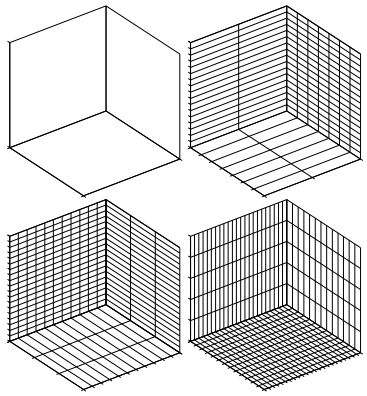

FIGURE 9. Different domain decompositions for parallelization.

6. Conclusions

In this paper, we introduced a variable-density particle-laden turbulent flow simulation code. The code is fourth-order accurate in time, and second-order accurate in space. It is fully parallel using MPI. Particles are modeled as Lagrangian points, while fluid is represented using a uniform staggered Eulerian grid. This code has demonstrated the expected results for various canonical problems, and has been used to discover physics in a variety of new fields involving particles, flow, and radiation.

Acknowledgments

This investigation was funded by the Advanced Simulation and Computing (ASC) program of the US Department of Energy's National Nuclear Security Administration via the PSAAP-II Center at Stanford.

Appendix A

In this section, we explain the details of the numerics used to solve for a variable-density flow with low-Mach number assumption.

Consider $C_{RK4}^{post} = [\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}]$ and $C_{RK4}^{pre} = [\frac{1}{2}, \frac{1}{2}, 1]$ as constant coefficients used to implement the RK4 time integration. Also, assume $k = 0, 1, 2, 3$ corresponds to the RK4 four substeps. Assume Q is some quantity of interest (e.g., density). At each RK4 substep Q_{new} is obtained from $Q^{(n)}$ and Q_{int}. $Q^{(n)}$ is the value of Q at time-step n, Q_{int} is the value at the previous RK4 substep, and Q_{new} is the new value of Q after the RK4 substep. For the first RK4 substep we initiate $Q_{int} = Q^{(n)}$. Note that we are solving for conservative fields ρ and ρu_i (here, we dropped the fluid subscript for simplicity). In the following, the time integration steps are explained.

We start by solving for the flow density

$$\rho_{new} = \rho^{(n)} - C_{RK4}^{pre}[k] \cdot \Delta t \cdot \frac{\partial}{\partial x_j}(\rho u_j)_{int}. \tag{A 1}$$

Next, solve for momentum without pressure

$$(\rho u_i)_{new} = \widetilde{(\rho u_i)}_{new} - C_{RK4}^{pre}[k] \cdot \Delta t \cdot \left(\frac{\partial p}{\partial x_i}\right)_{int}, \tag{A 2}$$

$$\widetilde{(\rho u_i)}_{new} = (\rho u_i)^{(n)} + C^{pre}_{RK4}[k] \cdot \Delta t \cdot \widetilde{\mathrm{RHS}}^{mom}_{int}, \tag{A 3}$$

where

$$\widetilde{\mathrm{RHS}}^{mom} = \frac{\partial}{\partial x_j}\left(\mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij})\right) + A\rho u_i + (\rho - \rho_0)g_i - \frac{\partial}{\partial x_j}(\rho u_i u_j) + \mathcal{P}\left(\frac{u_{p_i} - \mathcal{I}(u_{f_i})}{\tau_p}\right). \tag{A 4}$$

Here, $\widetilde{(.)}$ refers to the value of some quantity without considering the pressure gradient.

We then compute divergence of the velocity at the next substep

$$\nabla.\vec{u}_{new} = \frac{k}{C_p}\nabla^2(\frac{1}{\rho_{new}}) + \frac{\alpha R}{C_p}\frac{1}{P_{0new}} - \frac{C_v}{C_p}\frac{1}{P_{0new}}\left(\frac{dP_0}{dt}\right)_{new}, \tag{A 5}$$

where $\alpha = \mathcal{P}\left(2\pi D_p k(T_p - \mathcal{I}(T_f))\right)$ is the energy transferred from the particles to the gas per unit time per unit volume. Note that in the above equation P_{0new} and $\left(\frac{dP_0}{dt}\right)_{new}$ are not known. The last term, $\frac{C_v}{C_p}\frac{1}{P_{0new}}\left(\frac{dP_0}{dt}\right)_{new} = \chi$, is a scalar (only function of time) and will be determined later to satisfy the consistency of the Poisson equation. In order to compute P_{0new}, we can obtain the time-evolution equation for P_0, which is assumed to be only a function of space with a low-Mach number assumption. To do this, one can integrate the energy equation in space, after which all divergence terms vanish due to periodicity. Hence,

$$\frac{dP_0}{dt} = \frac{\alpha R \bar{c}}{C_v}. \tag{A 6}$$

Note that for the inflow-outflow case P_0 is constant in time and space. The right-hand side of the above equation is a constant number (in time and space). The time evolution-equation for P_0 using RK4 is

$$P_{0new} = {P_0}^n + C^{pre}_{RK4}[k] \cdot \Delta t \cdot \frac{\langle\alpha\rangle R}{C_v}. \tag{A 7}$$

As explained earlier, we keep χ as an unknown to be determined later in order to make the Poisson equation well-posed. † Rewrite Eq. (A 5) as below:

$$\nabla.\vec{u}_{new} = \mathrm{RHS}^{div}_{new} - \chi, \tag{A 8}$$

where in the above equation:

$\mathrm{RHS}^{div}_{new} = \frac{k}{C_p}\nabla^2(\frac{1}{\rho_{new}}) + \frac{\alpha R}{C_p}\frac{1}{P_{0new}}$, and χ is a number need to be determined.

Divide Eq. (A 2) by ρ_{new} and take divergence

$$\nabla.\vec{u}_{new} = \nabla.\left(\frac{\widetilde{(\rho u_i)}_{new}}{\rho_{new}}\right) - C^{pre}_{RK4}[k] \cdot \Delta t \cdot \nabla.(\frac{1}{\rho_{new}}\nabla p). \tag{A 9}$$

† Theoretically, the values resulting from Eq. (A 6) should be consistent with the Poisson equation. However, we make this adjustment in order to overcome the ill-posedness resulting from the numerical errors.

Now, we can substitute $\nabla.\vec{u}_{new}$ from Eq. (A 9) in Eq. (A 8)

$$\mathrm{RHS}^{div}_{new} - \chi = \nabla.\left(\frac{\widetilde{(\rho u_i)}_{new}}{\rho_{new}}\right) - C^{pre}_{RK4}[k]\cdot\Delta t\cdot\nabla.(\frac{1}{\rho_{new}}\nabla p) \tag{A 10}$$

$$\Rightarrow \nabla.(\frac{1}{\rho_{new}}\nabla p) = \frac{1}{C^{pre}_{RK4}[k]\cdot\Delta t}\left\{\nabla.\left(\frac{\widetilde{(\rho u_i)}_{new}}{\rho_{new}}\right) - \mathrm{RHS}^{div}_{new} + \chi\right\} = \mathrm{RHS}^{Pois}_{new} + \chi'. \tag{A 11}$$

This is a variable coefficient Poisson equation. When we take spatial integral of the above equation, for a fully periodic domain, the left hand side term vanishes. Therefore,

$$\chi' = \frac{\chi}{C^{pre}_{RK4}[k]\cdot\Delta t} = \frac{-\int \mathrm{RHS}^{Pois}_{new}\,dv}{\int dv} \rightarrow \chi = -C^{pre}_{RK4}[k]\cdot\Delta t\cdot\langle\mathrm{RHS}^{Pois}_{new}\rangle. \tag{A 12}$$

After solving the variable coefficient Poisson equation, we update the momentum based on Eq. (A 2). Note that during the RK4 loop, for a quantity Q with the time evolution equation $\frac{\partial Q}{\partial t} = \mathrm{RHS}_Q$, we construct $Q^{(n+1)}$ as

$$Q^{(n+1)} = Q^{(n+1)} + C^{post}_{RK4}[k]\cdot\Delta t\cdot\mathrm{RHS}_Q[k], \tag{A 13}$$

where $Q^{(n+1)}$ is set to be $Q^{(n)}$ before entering the RK4 loop.

Appendix B

Using a uniform staggered grid, our discretization is kinetic energy conservative. We are solving for momentum components, namely g_x, g_y, and g_z (e.g., $g_x = \rho u$). Here, we briefly explain the second-order energy conservative discretization. We use the notation used in Figure 2 (without losing generality of the 3D case). For example, consider the following terms for the $x-$momentum equation $\frac{\partial}{\partial x}(\frac{g_x g_x}{\rho}) + \frac{\partial}{\partial y}(\frac{g_x g_y}{\rho}) + \frac{\partial}{\partial z}(\frac{g_x g_z}{\rho})$, where g_x, g_y, and g_z are momentum in the x, y, and z directions, respectively. Note that we want all three terms to be computed on the faces (where $x-$momentum variables are stored).

$$\begin{aligned}\Delta x\cdot\frac{\partial}{\partial x}\left(\frac{g_x g_x}{\rho}\right)_{(I,J)} = &\frac{\left(g_{x(I+1,J)} + g_{x(I,J)}\right)\left(g_{x(I+1,J)} + g_{x(I,J)}\right)}{4\rho_{(i,j)}} - \\ &\frac{\left(g_{x(I,J)} + g_{x(I-1,J)}\right)\left(g_{x(I,J)} + g_{x(I-1,J)}\right)}{4\rho_{(i-1,j)}},\end{aligned} \tag{B 1}$$

$$\begin{aligned}\Delta y\cdot\frac{\partial}{\partial y}\left(\frac{g_x g_y}{\rho}\right)_{(I,J)} = &\frac{(g_{x(I,J+1)} + g_{x(I,J)})(g_{y(i',j'+1)} + g_{y(i'-1,j'+1)})}{\rho_{(i,j)} + \rho_{(i-1,j)} + \rho_{(i-1,j+1)} + \rho_{(i,j+1)}} - \\ &\frac{(g_{x(I,J-1)} + g_{x(I,J)})(g_{y(i',j')} + g_{y(i'-1,j')})}{\rho_{(i,j)} + \rho_{(i-1,j)} + \rho_{(i-1,j-1)} + \rho_{(i,j-1)}}.\end{aligned} \tag{B 2}$$

In 3D, all other terms are computed similarly: first, they are interpolated, and then finite difference is applied.

REFERENCES

Bassenne, M., Urzay, J. & Moin, P. 2015 Spatially-localized wavelet-based spectral analysis of preferential concentration in particle-laden turbulence. *Annual Research Briefs*, Center for Turbulence Research, Stanford University, pp. 3–16.

Bassenne, M., Urzay, J., Park, G. I. & Moin, P. 2015 Constant-energetics physical-space forcing methods for improved convergence to homogeneous-isotropic turbulence with application to particle-laden flows. Submitted to *Phys. Fluids*.

Batchelor, G., Canuto, V. & Chasnov, J. R. 1992 Homogeneous buoyancy-generated turbulence. *J. Fluid Mech.* **235**, 349–378.

Eaton, J. K. & Fessler, J. 1994 Preferential concentration of particles by turbulence. *Int. J. Multiphase Flow* **20**, 169–209.

Elghobashi, S. & Truesdell, G. 1993 On the two-way interaction between homogeneous turbulence and dispersed solid particles. i: Turbulence modification. *Phys. Fluids* **5** (7), 1790–1801.

Frankel, A., Pouransari, H., Coletti, F. & Mani, A. 2014 Settling of heated inertial particles through homogeneous turbulence. *Proceedings of the Summer Program*, Center for Turbulence Research, Stanford University, pp. 15–25.

Maxey, M. R. & Riley, J. J. 1983 Equation of motion for a small rigid sphere in a nonuniform flow. *Phys. Fluids* **26** (4), 883–889.

Passot, T. & Pouquet, A. 1987 Numerical simulation of compressible homogeneous flows in the turbulent regime. *J. Fluid Mech.* **181**, 441–466.

Park, G. I., Urzay, J., Bassenne, M. & Moin, P. 2015 A dynamic subgrid-scale model based on differential filters for LES of particle-laden turbulent flows. *Annual Research Briefs*, Center for Turbulence Research, Stanford University, pp. 17-26.

Plimpton, S., Pollock, R. & Stevens, M. 1997 Particle-mesh Ewald and rRESPA for parallel molecular dynamics simulations. In *PPSC*. Citeseer.

Pouransari, H., Kolla, H., Chen, J. & Mani, A. 2014 Spectral analysis of energy transfer in variable density, radiatively heated particle-laden flows. *Proceedings of the Summer Program*, Center for Turbulence Research, Stanford University, pp. 27–36.

Rosales, C. & Meneveau, C. 2005 Linear forcing in numerical simulations of isotropic turbulence: Physical space implementations and convergence properties. *Phys. Fluids* **17** (9), 095106.

Squires, K. D. & Eaton, J. K. 1990 Particle response and turbulence modification in isotropic turbulence. *Phys. Fluids* **2** (7), 1191–1203.

Urzay, J., Bassenne, M., Park, G. I. & Moin, P. 2015 Characteristic regimes of subgrid-scale coupling in LES of particle-laden turbulent flows. *Annual Research Briefs*, Center for Turbulence Research, Stanford University, pp. 3–13.

Vié, A., Pouransari, H., Zamansky, R. & Mani, A. 2015 Particle-laden flows forced by the disperse phase: comparison between Lagrangian and eulerian simulations. *Int. J. Multiphase Flow*.

Wang, L.-P. & Maxey, M. R. 1993 Settling velocity and concentration distribution of heavy particles in homogeneous isotropic turbulence. *J. Fluid Mech.* **256**, 27–68.

Zamansky, R., Coletti, F., Massot, M. & Mani, A. 2014 Radiation induces turbulence in particle-laden fluids. *Phys. Fluids* **26** (7), 071701.

Pouransari, H. & Mani, A. 2015 On the effects of preferential concentration on heat transfer in particle-based solar receivers. Submitted to *J. Fluid Mech.*