Aggregation and Interpolation in CFD–DEM Based on Diffusion Equations

Rui Sun and Heng Xiao

November 2, 2017

1 Introduction and Motivation

1.1 Interpolations Needed in a CFD–DEM Solver

Here, aggregation means Lagrangian phase to Eulerian phase operations; Interpolation means Eulerian phase to Lagrangian phase operations. Sometimes they are also referred to as "forward" and "backward" interpolations.

1. Aggregation of solid volume fraction Eulerian field ε_s from particle data (position $\mathbf{x}_p$)
2. Aggregation of Eulerian solid-phase velocity $\mathbf{u}_s$ from particle data (position $\mathbf{x}_p$ and velocity $\mathbf{u}_p$)
3. Aggregation of fluid–particle interaction forces onto fluid cells (conservation is essential to ensure momentum balance of the fluid–particle system). See below.
4. Interpolation of Eulerian fluid-phase velocity field $\mathbf{u}_f$ at particle locations $\mathbf{u}_{f@p}$ (needed for calculating drag forces); Similar interpolation are needed for ε_s.

1.2 Desirable Properties of the Interpolation Procedure

1. (Essential) Free of boundary abnormally at both physical boundaries and processor boundaries. For example, for a case with uniformly distributed, the obtained solid volume fraction ε_s should be a uniform field.
2. (Essential) Conservation of mass and momentum, i.e.,

$$\sum_{i=1}^{N_c} \varepsilon_{s,i} V_{c,i} = \sum_{k=1}^{N_p} V_{p,k} \tag{1}$$

$$\sum_{i=1}^{N_c} \mathbf{F}_{c,i} = -\sum_{k=1}^{N_p} \mathbf{f}_{p,k} \tag{2}$$

where N_c is the number of cells in the OpenFOAM mesh; N_p is the number of particles in the system; $V_{c,i}$ is the volume of cell i; $\mathbf{F}_{c,i}$ is the force on cell i exerted by all particles (in this cell or adjacent cells, depending on the mapping scheme); $\mathbf{f}_{p,k}$ is the force exerted on particle k exerted by the fluid.

3. Convenience of implementation in a parallel code.
4. Convenience of implementation in an unstructured mesh as in OpenFOAM.

1.3 Existing Algorithms and Their Advantages and Drawbacks

Particle Centroid Method

- $+$ satisfies the conservation requirements with straightforward implementation;
- $+$ easy implementation on unstructured mesh;
- $+$ easy implementation on MPI parallel codes;
- $-$ large errors in obtained ε_s leading to errors in force calculations

Analytical Method

- $+$ better accuracy in obtained ε_s
- $-$ still would not work if particle diameter is larger than cell size.
- $-$ difficulty to implement in unstructured mesh.

Statistical Kernel Function (Xiao & Sun)

- $+$ Better accuracy than PCM, leading to smoother ε_s field
- $-$ Difficult and inefficient to implement when bandwidth is larger than 2 or 3 cell size.

Two-Grid Formulation (Deb & Tafti)

- $+$ better accuracy in obtained ε_s field
- $-$ Extension to unstructured grid not straightforward
- $-$ For unstructured grid, parallel computing pose challenge

2 Proposed Method for Aggregation and Interpolation

2.1 Equivalence Between Diffusion- and Kernel-Based Methods

Consider 1D diffusion equation in infinite domain:

$$\frac{\partial \varepsilon}{\partial t} = \frac{\partial^2 \varepsilon}{\partial x^2} \text{ on } -\infty < x < \infty, t > 0 \tag{3}$$

$$\varepsilon|_{t=0} = f(x) \tag{4}$$

where $f(x)$ is a square integrable function; x and t are normalized spatial and temporal coordinates, respectively. The solution to the equation above based on Green's function is:

$$\varepsilon(x,t) = \int_{-\infty}^{\infty} K(x-\xi,t) f(\xi) d\xi \tag{5}$$

$$\text{where } K(x,t) = \frac{1}{\sqrt{4\pi t}} \exp\left[-\frac{x^2}{4t}\right] \tag{6}$$

If the initial condition is a delta function centered at x_0, i.e., $f(\xi) = \delta(\xi - x_0)$, according to the sifting property of the delta function, the solution is then:

$$\varepsilon(x,t) = \int_{-\infty}^{\infty} K(x-\xi)\ \delta(\xi - x_0) d\xi \tag{7}$$

$$= K(x - x_0, t) \tag{8}$$

Rui: [Insert a 1D picture here for illustration]

Consider an initial condition with a superposition of multiple delta functions centered at x_i (locations of cell centers),

$$f(\xi) = \sum_{i=1}^{N_c} c_i\ \delta(\xi - x_i) \tag{9}$$

where c_i is the multiplier coefficient. Similarly, the solution is a linear combination of the kernel functions:

$$\varepsilon(x,t) = \sum_{i=1}^{N_c} c_i\ K(x - x_i, t) \tag{10}$$

We are interested in the solution at a fixed time τ, denoted as $\hat{\varepsilon}(x) \equiv \varepsilon(x,\tau)$. The solution is rewritten as:

$$\hat{\varepsilon}(x) = \sum_{i=1}^{N_c} c_i\ K(x - x_i) = \sum_{i=1}^{N_c} c_i\ \frac{1}{\sqrt{4\pi\tau}} \exp\left[-\frac{(x-x_i)^2}{4\tau}\right] \tag{11}$$

That is, the solution is a superposition of N_c kernel functions centered at cell centers x_i. This is equivalent to the kernel function smoothing. The bandwidth of the kernel function is determined by the time τ. One can interpret c_i as the total solid particle volume in cell i. Correspondingly, $\hat{\varepsilon}(x)$ is the solid volume fraction at any location x.

$$\hat{\varepsilon}(x) = \left.\frac{dV_s}{dV}\right|_x \tag{12}$$

where V_s and V indicate solid and total volume, respectively.

Rui: [Insert a 1D picture here for illustration]

2.2 Algorithm for Solid Volume Calculation

1. Sum up particle volume in each cell, and obtain solid volume fraction $\tilde{\varepsilon}$ by dividing corresponding cell volumes.
2. Determine τ based on desired kernel bandwidth b, which is in turn specified based on particle diameters, $b = nd_p$.

   To be consistent with the definition of bandwidth specified in (Xiao and Sun 2010), the diffusion time should be chosen as
   $$\sqrt{4\tau} = b, \tag{13}$$
   or equivalently
   $$\tau = b^2/4. \tag{14}$$
   If we chose the bandwidth to be $6d_p$, then $\tau = 9d_p^2$.

   Alternatively, one can fix $\tau = 1$, and specify diffusion coefficient D. The benefit of the latter is being able to specify a homogeneous field based on the mean (or max) particle diameter d_p in each cell. In this case, we have
   $$D = 1/(9d_p^2) \tag{15}$$
3. Solve diffusion equation to τ, the obtained solid volume fraction is ε.

Other Lagrangian-to-Eulerian aggregations, i.e., computing particle forces on the fluid according to particle forces, and obtaining solid phase velocity field $\mathbf{u}_s$ according to individual particle velocities can be done in a similar manner.

2.3 Eulerian-to-Lagrangian Interpolation

Another operation needed for the calculation of particle drag forces is interpolation of Eulerian field quantities such as volume fraction ε and fluid phase velocity $\mathbf{u}_f$ to particle locations. The Eulerican quantities may be based on fine meshes comparable to or smaller than particle diameters. Directly taking values of the host cell or averaging from the neighbors of the host cell introduces large representation errors. One needs to smooth out the small-scale features.

1. Integrate the diffusion equation with the Eulerian field to be interpolated as initial condition. The obtained field will be used for interpolation in next step only, and will be discarded afterward.
2. Use the“direct-mapping” or “neighbor-weight” algorithm in OpenFOAM to interpolate cell values to particle locations.

2.4 Conservation Characteristics

The diffusion equations are solved based on conservative finite volume method. Local and global conservation of the diffused quantity (e.g., volume fraction) is guaranteed.

2.5 Effects Near Boundaries

The analysis above assumed infinite domain, and would be a good estimation in the interior of the computational domain. Within a few bandwidth of physical boundaries, further analysis are needed. Note that, however, processor boundaries are taken care of when solving the diffusion equations, and thus do not pose major difficulty.

Rui: test this algorithm with a uniform particle distribution.

2.6 Computational Cost

Rui: Estimate how many time steps are needed to solve the diffusion cost if explicit time stepping is used, assuming $\Delta x = d_p$. Note the time step restrictions for diffusion equation.

2.7 Summary of Advantages and Disadvantages of Aggregation Methods

*Rui: Fix the format of this table (Table **??**).*

Table 1: Summary of Advantages and Disadvantages of Aggregation Methods

methods	unstructured mesh	implementation in parallel code	smooth field on small cells
PCM	easy	easy	no
two-grid formulation	difficult	difficult	yes
analytical method	difficult	difficult	yes (if $\Delta x >\sim 2d_p$)
statistical kernel function	moderate	moderate	yes
diffusion-based method	easy	easy	yes

3 Numerical Results

The following case are tested:

Case 1: Comparison of kernel function and diffusion equation based aggregation with 3 to 5 particles on 1D and 2D domains. Demonstrate the equivalence between the two methods. [Use Matlab]

Case 2: On a 2D Cartesian grid, perform volume fraction aggregation using diffusion, and compare with analytical and two-grid formulations. [Use Matlab]

Case 3: Aggregation of volume fraction on a stretched grid, and on a unstructured grid. [Use OpenFOAM]

Case 4: Test fluidization case with PCM and diffusion-based aggregation, and compare pressure signal. (Ref to AIChe paper).

The purpose of each test case is as follows

Case 1: Demonstrate equivalence of between kernel function based aggregation and diffusion-based aggregation.

Case 2: Examine the accuracy of diffusion-based aggregation method.

Case 3: Evaluate the performance of diffusion-based aggregation on unstructured meshes.

Case 4: Evaluate the performance

Rui: Put case details in this file, including figures.

Table 2: Computational setup of test cases.

	Case 1	Case 2	Case 3	Case 4
Cell size Δx				
Dimension	1D			
particle distribution pattern		random/lattice	random	
number of particles				
Mesh	structured	structured	unstructured	?