# Derivation of fluid-particle flow equations in lammpsFoam

Rui Sun[1] and Heng Xiao*[1]

[1]Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, Virginia, USA

## Contents



## 1 Introduction

In CFD–DEM simulations, the fluid volume fraction has to be considered when solving the momentum equation. This will make the simulation of fluid motion different from solving the original Navier–Stokes Equation. Anderson and Jackson [**?**] derived the fluid phase momentum equations of particle-laden flow, and some people used their equation to perform numerical simulations and obtained very good results [**?**]. This document aims at showing how the CFD–DEM solver lammpsFoam works when solving the fluid phase equations.

*Email: hengxiao@vt.edu

# 2 Derivation of the equations

## 2.1 Fluid Phase Momentum Equations

The derivation of the fluid phase momentum equation can be seen in the study of Xiao et al. [?] and Kafui et al. [?]. Both simulations are based on the derivation of Anderson and Jackson [?]:

$$\frac{\partial(\varepsilon_f \rho_f u_i)}{\partial t} + \frac{\partial(\varepsilon_f \rho_f u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right) + \varepsilon_f \rho_f g_i - f_{i,fp} \tag{1}$$

the fluid-particle interaction force by volume $f_{i,fp}$ is:

$$f_{i,fp} = -\varepsilon_s \frac{\partial p}{\partial x_i} + \varepsilon_s \frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right) + n\varepsilon_f f_{i,drag}/V_c \tag{2}$$

Here, $n$ is the number of particles in the CFD cell, $V_c$ is the volume of CFD cell.

When the fluid-particle interaction force in Eq. (2) is plugged into Eq. (1), we have (mentioned in Xiao et al. [?] (2.8)):

$$\begin{aligned} &\frac{\partial(\varepsilon_f \rho_f u_i)}{\partial t} + \frac{\partial(\varepsilon_f \rho_f u_i u_j)}{\partial x_j} = \\ &\quad -\varepsilon_f \frac{\partial p}{\partial x_i} + \varepsilon_f \frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right) + \varepsilon_f \rho_f g_i - \varepsilon_f \sum_{i=1}^{c_n} f_{i,drag}/V_c \end{aligned} \tag{3}$$

In Kafui et al.'s study [?], the momentum equation is obtained differently:

$$\begin{aligned} &\frac{\partial(\varepsilon_f \rho_f u_i)}{\partial t} + \frac{\partial(\varepsilon_f \rho_f u_i u_j)}{\partial x_j} = \\ &\quad -\varepsilon_f \frac{1}{\rho} \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \varepsilon_f \frac{\partial u_i}{\partial x_j}\right) + \varepsilon_f \rho_f g_i - \varepsilon_f \sum_{i=1}^{c_n} f_{i,drag}/V_c \end{aligned} \tag{4}$$

Moreover, Kafui et al. combined the drag force with part of the pressure term in his paper:

$$\frac{\partial(\varepsilon_f \rho_f u_i)}{\partial t} + \frac{\partial(\varepsilon_f \rho_f u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \varepsilon_f \frac{\partial u_i}{\partial x_j}\right) + \varepsilon_f \rho_f g_i - f_{i,drag,kafui} \tag{5}$$

Since

$$f_{i,drag,Kafui} = -(1-\varepsilon_f)\frac{\partial p}{\partial x_i} + \varepsilon_f \sum_{i=1}^{c_n} f_{i,drag}/V_c, \tag{6}$$

Eq. (4) and Eq. (5) are theoretically equivalent equations in different forms. Here, the author of this handbook is still not sure about the derivation of Kafui et al when dealing with the stress tensor. However, the fluid momentum equation Kafui et al. used [?] is the same as the equations used to describe the motion of multipahse Eulerian flow [?].

## 2.2 Multiphase Flow in OpenFOAM

lammpsFoam is based on the Ph.D. thesis of H. Rusche, following the equation for fluid phase velocity proposed by Ishii [?] and Kafui et al. [?]. The L.H.S. of the equation, using to describe

the fluid motion, is:

$$\frac{\partial(\varepsilon_f\rho_f u_i)}{\partial t}+\frac{\partial(\varepsilon_f\rho_f u_i u_j)}{\partial x_j}- \left(u_i\frac{\partial(\varepsilon_f\rho_f)}{\partial t}+u_i\frac{\partial(\varepsilon_f\rho_f u_j)}{\partial x_j}+\varepsilon_f\rho_f\frac{\partial u_i}{\partial t}+\varepsilon_f\rho_f u_j\frac{\partial u_i}{\partial x_j}\right) \tag{7}$$

From the mass conservative equation:

$$\frac{\partial(\varepsilon_f\rho_f)}{\partial t}+\frac{\partial(\varepsilon_f\rho_f u_i)}{\partial x_i}=0 \tag{8}$$

We have:

$$\frac{\partial(\varepsilon_f\rho_f u_i)}{\partial t}+\frac{\partial(\varepsilon_f u_i u_j)}{\partial x_j}=\varepsilon_f\rho_f\frac{\partial u_i}{\partial t}+\varepsilon_f\rho_f u_j\frac{\partial u_i}{\partial x_j} \tag{9}$$

Then,

$$\varepsilon_f\rho_f\frac{\partial u_i}{\partial t}+\varepsilon_f\rho_f u_j\frac{\partial u_i}{\partial x_j}- \left(-\varepsilon_f\frac{\partial p}{\partial x_i}+\varepsilon_f\frac{\partial}{\partial x_j}\left(\mu\frac{\partial u_i}{\partial x_j}\right)+\frac{\partial u_i}{\partial x_j}\frac{\partial}{\partial x_j}(\mu\varepsilon_f)+\varepsilon_f\rho_f g_i-\varepsilon_f\sum_{i=1}^{c_n}f_{i,drag}/V_c\right). \tag{10}$$

If we divide both sides of Eq. (10), by $\varepsilon_f\rho_f$, we have:

$$\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}- \left(-\frac{1}{\rho_f}\frac{\partial p}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\nu\frac{\partial u_i}{\partial x_j}\right)+\frac{1}{\varepsilon_f}\frac{\partial u_i}{\partial x_j}\frac{\partial}{\partial x_j}(\nu\varepsilon_f)+g_i-\frac{1}{\rho_f}\sum_{i=1}^{c_n}f_{i,drag}/V_c\right). \tag{11}$$

It is equivalent with the equation (3.26) proposed by Rusche.

The code of lammpsFoam (if the turbulence is added) is in a consistent form with Eq. (11):

```
UbEqn =
(
    (scalar(1) + Cvm*rhob*alpha/rhob)*
    (
        fvm::ddt(Ub)
      + fvm::div(phib, Ub, "div(phib,Ub)")
      - fvm::Sp(fvc::div(phib), Ub)
    )

  - fvm::laplacian(nuEffb, Ub)
  + fvc::div(Rcb)

  + fvm::div(phiRb, Ub, "div(phib,Ub)")
  - fvm::Sp(fvc::div(phiRb), Ub)

  + (fvc::grad(beta)/(fvc::average(beta) + scalar(0.001)) & Rcb)
 ==
//  g  //  Buoyancy term transfered to p-equation
  - fvm::Sp(dragCoef/rhob, Ub)
//  Explicit drag transfered to p-equation
//+ alpha/rhob*dragCoef*Ua
  + alpha/rhob*(liftCoeff + Cvm*rhob*DDtUa)
);
```

The second half of UEqn.H is the expression of various forces of in the simulations. The top half of UEqn.H can be interpreted as:

$$\begin{aligned}\frac{\partial u_i}{\partial t}+\frac{\partial(u_i u_j)}{\partial x_j}-u_i\frac{\partial u_j}{\partial x_j}-\frac{\partial}{\partial x_j}\left(\nu_{eff}\frac{\partial u_i}{\partial x_j}\right)+\frac{\partial R^c_{ij}}{\partial x_j}\\ +\frac{\partial}{\partial x_j}\left(u_i\frac{-\nu_{eff}}{\varepsilon_f}\frac{\partial \varepsilon_f}{\partial x_j}\right)-u_i\frac{\partial}{\partial x_j}\left(\frac{-\nu_{eff}}{\varepsilon_f}\frac{\partial \varepsilon_f}{\partial x_j}\right)+\frac{1}{\varepsilon_f}\frac{\partial \varepsilon_f}{\partial x_j}R^c_{ij}\end{aligned} \tag{12}$$

which is equivalent to:

$$\begin{aligned}\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}-\frac{\partial}{\partial x_j}\left(\nu_{eff}\frac{\partial u_i}{\partial x_j}\right)+\frac{\partial R^c_{ij}}{\partial x_j}\\ -\frac{1}{\varepsilon_f}\frac{\partial \varepsilon_f}{\partial x_i}\nu_{eff}\frac{\partial u_i}{\partial x_j}+\frac{1}{\varepsilon_f}\frac{\partial \varepsilon_f}{\partial x_i}R^c_{ij}\end{aligned} \tag{13}$$

Eq. (13) is also equivalent to:

$$\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}+\frac{\partial}{\partial x_j}\left(-\nu_{eff}\frac{\partial u_i}{\partial x_j}+R^c_{ij}\right)+\frac{1}{\varepsilon_f}\frac{\partial \varepsilon_f}{\partial x_i}\left(-\nu_{eff}\frac{\partial u_i}{\partial x_j}+R^c_{ij}\right) \tag{14}$$

Rusche took $-\nu_{eff}\frac{\partial u_i}{\partial x_j}+R^c_{ij}$ as the stress tensor.

$$-\nu_{eff}\frac{\partial u_i}{\partial x_j}+R^c_{ij}=\tau_{ij}+R_{ij} \tag{15}$$

Then we proved the U equation in lammpsFoam is consistent with Eq. (11) except for the gravity term (The gravity term is moved to the pressure equation).

The code of twoPhaseEulerianFoam is:

```
U2Eqn =
(
    fvm::ddt(alpha2, U2)
  + fvm::div(alphaPhi2, U2)
  - fvm::Sp(fvc::ddt(alpha2) + fvc::div(alphaPhi2), U2)
  + phase2.turbulence().divDevReff(U2)
 ==
  - fvm::Sp(dragCoeff/rho2, U2)
  + (
        liftForce
      + wallLubricationForce
      + turbulentDispersionForce
    )/rho2
  - virtualMassCoeff/rho2
   *(
        fvm::ddt(U2)
      + fvm::div(phi2, U2)
      - fvm::Sp(fvc::div(phi2), U2)
      - DDtU1
    )
);
mrfZones.addCoriolis(alpha2 + virtualMassCoeff/rho2, U2Eqn);
U2Eqn.relax();
);
```

The second half of UEqn.H is the expression of various forces of in the simulations. The top half of twoPhaseEulerianFoam can be interpreted as:

$$\frac{\partial(\varepsilon_f u_i)}{\partial t} + \frac{\partial(\varepsilon_f u_i u_j)}{\partial x_j} - u_i \frac{\partial(\varepsilon_f)}{\partial t} - u_i \frac{\partial(\varepsilon_f u_j)}{\partial x_j} + stress, \tag{16}$$

which is equivalent to:

$$\varepsilon_f \frac{\partial u_i}{\partial t} + \varepsilon_f u_j \frac{\partial u_i}{\partial x_j} + stress \tag{17}$$

Given more studies of the turbulent stress term, this term is mostly taken as $-\frac{\partial}{\partial x_j}\left(\nu_{eff}\varepsilon_f \frac{\partial u_i}{\partial x_j}\right)$ instead of $-\varepsilon_f \frac{\partial}{\partial x_j}\left(\nu_{eff}\frac{\partial u_i}{\partial x_j}\right)$. A further study shows that this difference when dealing with the shear stress is because of the assumptions of the fluid-solid flow and fluid-fluid flow is different.

# 3 Treatment of the stress tensor

## 3.1 Brief Review

It seems that the governing equations are different in some CFD–DEM simulations, especially in the handling of the stress tensor. There might be no definite answer to this problem, but van Wachem et al.'s study [**?**] may provide some evidence for this open question.

In van Wachem et al.'s study, Anderson and Jackson [**?**] derived the continuum equations of motion for gas-particle flow. However, Ishii's assumptions of the derivation [**?**] the fluid–fluid governing equations are slightly different, which is usually used to study the bubble flow problem. The summary of different forms of equations are shown in Fig. 1

In van Wachem et al's study, both methods are validated. Despite this, it is only accepted that CFD–DEM simulations using $\varepsilon_f \nabla \cdot \tau$ as the stress tensor, and fluid–fluid simulations using $\nabla \cdot \varepsilon_f \tau$. However, most CFD–DEM simulations do not follow this rule, shown is Table 1.

**Table 1: Summary of the fluid–solid and fluid–fluid simulations by different cases**

| author | modeling method | | stress tensor | |
|---|---|---|---|---|
| | fluid–solid | fluid–fluid | $\varepsilon_f \nabla \cdot \tau$ | $\nabla \cdot \varepsilon_f \tau$ |
| Xiao et al. [**?**] | ✓ | | ✓ | |
| A.B. Yu (JFM, 2010) | ✓ | | ✓ | |
| A.B. Yu (CES, 1997) | ✓ | | | ✓ |
| A.B. Yu (CES, 2007) | ✓ | | | ✓ |
| Kafui et al. [**?**] | ✓ | | | ✓ |
| Mueller et al. | ✓ | | | ✓ |
| Rusche [**?**] | | ✓ | | ✓ |
| bubbleFoam | | ✓ | | ✓ |
| twoPhaseEulerFoam | | ✓ | | ✓ |
| Tsuji et al. (1993) | ✓ | | ignored | |

Table 1
The governing equations for fluid–fluid and fluid–solid flows

Continuity equations (equal for both phases)

$$\frac{\partial \epsilon}{\partial t} + \nabla(\epsilon \boldsymbol{v}) = 0$$

Momentum equations for fluid–solid flow

$$\rho_{\mathrm{g}}\epsilon_{\mathrm{g}}\left[\frac{\partial \boldsymbol{v}_{\mathrm{g}}}{\partial t} + \boldsymbol{v}_{\mathrm{g}} \cdot \nabla \boldsymbol{v}_{\mathrm{g}}\right] = -\epsilon_{\mathrm{g}}\nabla P + \epsilon_{\mathrm{g}}\nabla \cdot \bar{\bar{\tau}}_{\mathrm{g}} + \epsilon_{\mathrm{g}}\rho_{\mathrm{g}}\boldsymbol{g} - \boldsymbol{I}$$

$$\rho_{\mathrm{s}}\epsilon_{\mathrm{s}}\left[\frac{\partial \boldsymbol{v}_{\mathrm{s}}}{\partial t} + \boldsymbol{v}_{\mathrm{s}} \cdot \nabla \boldsymbol{v}_{\mathrm{s}}\right] = -\epsilon_{\mathrm{s}}\nabla P + \epsilon_{\mathrm{s}}\nabla \cdot \bar{\bar{\tau}}_{\mathrm{g}} + \nabla \cdot \bar{\bar{\tau}}_{\mathrm{s}} - \nabla P_{\mathrm{s}} + \epsilon_{\mathrm{s}}\rho_{\mathrm{s}}\boldsymbol{g} + \boldsymbol{I}$$

Momentum equations for fluid–fluid flow

$$\rho_i\epsilon_i\left[\frac{\partial \boldsymbol{v}_i}{\partial t} + \boldsymbol{v}_i \cdot \nabla \boldsymbol{v}_i\right] = -\epsilon_i\nabla P + \nabla \cdot \epsilon_i\bar{\bar{\tau}}_i + \epsilon_i\rho_i\boldsymbol{g} - \boldsymbol{I}$$

$$\rho_j\epsilon_j\left[\frac{\partial \boldsymbol{v}_j}{\partial t} + \boldsymbol{v}_j \cdot \nabla \boldsymbol{v}_j\right] = -\epsilon_j\nabla P + \nabla \cdot \epsilon_j\bar{\bar{\tau}}_j + \epsilon_j\rho_j\boldsymbol{g} + \boldsymbol{I}$$

**Figure 1: The equations for fluid–solid flow and fluid–fluid flow summarized by van Wachem.**

### 3.2 Conclusions

1. CFD–DEM simulations should use $\varepsilon_f \nabla \cdot \tau$ as the stress tensor, but in many simulations, people use $\nabla \cdot \varepsilon_f \tau$. In some cases, the stress tensor is even ignored.

2. Both $\varepsilon_f \nabla \cdot \tau$ and $\nabla \cdot \varepsilon_f \tau$ seem acceptable in CFD–DEM simulations. In some simulations, the equations derived from Anderson and Jackson are directly written as $\nabla \cdot \varepsilon_f \tau$. (See A.B. Yu 1997, Mueller et al.)

3. Another evidence that both $\varepsilon_f \nabla \cdot \tau$ and $\nabla \cdot \varepsilon_f \tau$ are acceptable is: in some of Wachem's simulations, he would even be confused with the stress tensor. (See AIChE, 2012)

4. In fluid–fluid flow, the stress tensor is mostly taken as $\nabla \cdot \varepsilon_f \tau$.

## 4 Large Eddy simulations in OpenFOAM

Since LES will be implemented in the new lammpsFoam solver, this section serves as an introduction part. It demonstrates how LES is simulated in OpenFOAM.

### 4.1 Introduction

Let's start from Navier–Stokes Equations:

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial u_i}{\partial x_j}\right). \tag{18}$$

If the velocity $u$ and the pressure $p$ are filtered, we have:

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial \overline{u_i u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \overline{u_i}}{\partial x_j}\right). \tag{19}$$

If we define the residual stress tensor $\tau_{ij} = \overline{u_i u_j} - \overline{u_i}\,\overline{u_j}$, we have

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial \overline{u_i}\,\overline{u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \overline{u_i}}{\partial x_j}\right) + \frac{\partial \tau_{ij}}{\partial x_j}. \tag{20}$$

Usually, we employ Boussinesq hypothesis and

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = 2\nu_t \overline{S_{ij}}, \tag{21}$$

Where $\overline{S_{ij}} = 1/2(\partial \overline{u_i}/\partial x_j + \partial \overline{u_j}/\partial x_i)$. Then we have:

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial \overline{u_i}\,\overline{u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left([\nu + \nu_t]\frac{\partial \overline{u_i}}{\partial x_j}\right). \tag{22}$$

In this case, the pressure is modified to include the trace term $-1/3\tau_{kk}\delta_{ij}$.

## 4.2 Filtered Navier–Stokes Equations in OpenFOAM

In pisoFoam.C, we have:

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  + turbulence->divDevReff(U)
);
```

In a representative LES model, say GenEddyVisc.C, we have:

```
tmp<fvVectorMatrix>
GenEddyVisc::divDevReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(nuEff(), U)
      - fvc::div(nuEff()*dev(T(fvc::grad(U))))
    );
}
```

Since the term:

```
- fvc::div(nuEff()*dev(T(fvc::grad(U))))
```

will be zero if the fluid mass is conservative (see derivation), we have:

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nuEff(), U)
);
```

which is consistent with Eq. (22).

### 4.3 Subgrid scale modeling

According to the study performed by Fureby et al. [**?**], the SGS stress tensor is represented as:

$$\begin{aligned} \tau_{ij} &= \frac{2}{3}kI_{ij} - 2\nu_t S_{ij,D}, \\ k &= \frac{1}{2}\mathrm{tr}(\tau_{ij}), S_{ij,D} = S_{ij} - \frac{1}{3}\mathrm{tr}(S_{ij})I_{ij} \end{aligned} \tag{23}$$

According to Smagorinsky model,

$$k = c_I \Delta^2 |S_{ij}|^2, \nu_t = c_D \Delta^2 |S_{ij}|, \tag{24}$$

Then the SGS stress tensor can be modeled.

Here is how the SGS model in implemented in OpenFOAM

```
B = 2/3*k*I - 2*nuSgs*dev(D)
Beff = 2/3*k*I - 2*nuEff*dev(D)

D = symm(grad(U));
k = (2*ck/ce)*delta^2*||D||^2
nuSgs = ck*sqrt(k)*delta
nuEff = nuSgs + nu
```

The defination of $k$ and $\nu_t$ are the same as the reference since:

$$\begin{aligned} \nu_t &= c_k \sqrt{2c_k/c_e} \Delta^2 |S_{ij}|, \\ c_D &= c_k \sqrt{2c_k/c_e} \approx 0.042. \end{aligned} \tag{25}$$

It is also consistent with Pope's defination [**?**]. (The $c_D$ recommended by Pope is 0.028, but it is because the definations of $|S_{ij}|$ are not consistent in both studies. The equations for $\nu_t$ are consistent.) The SGS model in OpenFOAM is consistent with the turbulent theories.

### 4.4 LES–DEM Equations

From Fang et al.'s work [**?**] (I copied the original equations):

$$\frac{\partial(\bar{\varepsilon}\overline{u_i})}{\partial t} + \frac{\partial(\bar{\varepsilon}\overline{u_i}\overline{u_j})}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_i}\left(\nu\frac{(\partial\bar{\varepsilon}\overline{u_i})}{\partial x_j}\right) + \frac{1}{\rho}\frac{\partial(\bar{\varepsilon}\tau_{ij})}{\partial x_j}. \tag{26}$$

the sub-grid stress tensor is taken as:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = 2\mu_t\overline{S_{ij}}, \tag{27}$$

Where $\overline{S_{ij}} = 1/2(\partial\overline{u_i}/\partial x_j + \partial\overline{u_j}/\partial x_i)$. The turbulent viscosity is defined as:

$$\nu_t = (C_s\Delta)^2\sqrt{2\overline{S_{ij}}\,\overline{S_{ij}}}. \tag{28}$$

In my perspective, there might be a few mistakes in the equations: The correct form of Navier–Stokes equations is:

$$\frac{\partial\bar{\varepsilon}\overline{u_i}}{\partial t} + \frac{\partial\bar{\varepsilon}\overline{u_i}\overline{u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu\frac{\partial\bar{\varepsilon}\overline{u_i}}{\partial x_j}\right) + \frac{1}{\rho}\frac{\partial\bar{\varepsilon}\tau_{ij}}{\partial x_j}. \tag{29}$$

Also, the implementation of this equation is also straightforward.

# 5 Useful Advices

In the joint group meeting on 09/05/14, Dr. Liu suggested we check the forces added to the particles. I looked through the code and noticed that:

```
// local pressure gradient is used though (no weighting).
pDrag_[particleI] =
    Jd_[particleI]*(1.0 - pAlpha_[particleI])
   *p.Vol()*Uri_[particleI]          // Drag
  - gradp[p.cell()]*p.Vol();         // Buoyancy
```

the drag and buoyancy is added to the particles, but the added mass is not considered. For particle–gas flow, the added mass is negligible, but for sediment transport simulations, this is not.

Another useful advice from Dr. Calantoni is that the distribution of bottom particles should not be regular. Adding a random height of the particles may help.

# 6 Plan for new implementations

After we talked with Dr. Liu and Dr. Calantoni, we found out that we can still improve the simulation in several ways.

1. Add the virtual mass force in the particle motion, though it is not considered in Schmeeckle's simulations [**?**].
2. Change the initial distribution of the "frozen" particles, add a random offset to them.
3. Verification of the code and validation of the cases.

If it is still not working, the turbulence stress should be added.

1. Keep the source code of lammpsFoam. Update it to 2.3.x version.
2. Change the UEqn part, make it looks like the new version. (The meaning of each term will be more explicit after this.)
3. Make use of the two-phase turbulence models in 2.3.x version, change the stress tensor in lammpsFoam. (The stress tensor in two-phase turbulence models is $\nabla \cdot \varepsilon_f \tau$ in OpenFOAM. Improve the two-phase stress tensor to the correct one.)
4. Verification and Validation.