Implementation of a wall boundary condition for the solid phase in a gas-particle flow in `twoPhaseEulerFoam`

Mohsen Zarepour

Mechanical Engineering Department,
University of Saskatchewan,
Saskatoon, Canada

January 17, 2022

Outline

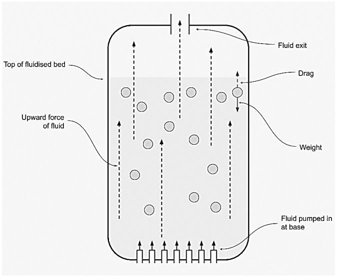
- Gas-solid flow modelling approaches
- A review of the `twoPhaseEulerFoam` solver
- Implementation of the model of the Schneiderbauer et al.
- Test case study

Fluidized bed

The unit works on the principal of **fluidization**.

Advantages:

- Higher rates of heat and mass transfer
- Uniform temperature in the bed
- Low maintenance cost and less labor intensive



Schematic of a fluidized bed

(https://www.baristahustle.com/blog/fluidised-beds/)

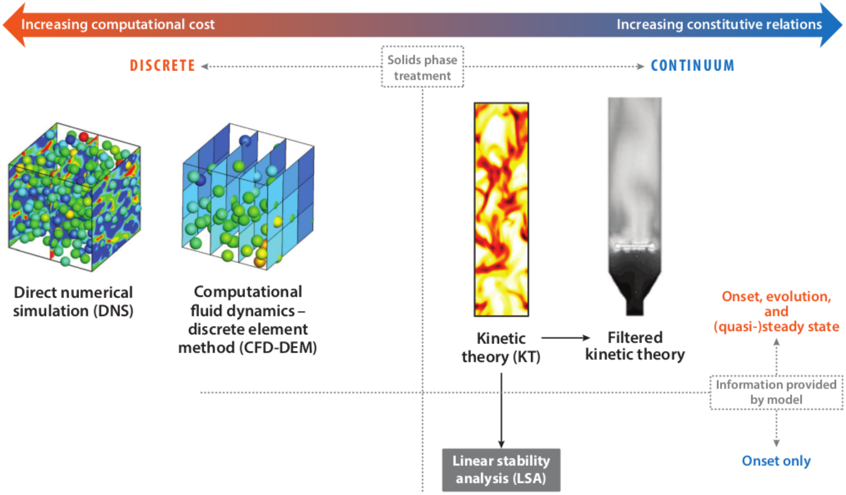
Modelling approaches for gas-particle flows

Methods are categorized based on solids phase treatment.

$$\text{Discrete methods} \quad \begin{cases} \text{Direct Numerical Simulation (DNS)} \\ \text{Discrete Element Method (DEM)} \end{cases}$$

$$\text{Continuum methods} \begin{cases} \text{Two-Fluid Model (TFM)} \\ \text{Filtered Two-Fluid Model (f-TFM)} \end{cases}$$

Models at a glance


Increasing computational cost
Increasing constitutive relations
DISCRETE
Solids phase treatment
CONTINUUM
Direct numerical simulation (DNS)
Computational fluid dynamics – discrete element method (CFD-DEM)
Kinetic theory (KT)
Filtered kinetic theory
Onset, evolution, and (quasi-)steady state
Information provided by model
Onset only
Linear stability analysis (LSA)

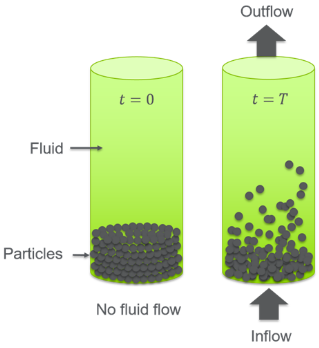
Hierarchy of methods used to study the fluidized beds [1].

Conventional models

Discrete Element Method (DEM)

- Conservation equations are solved for the gas phase
- Newton's 2^{nd} law of motion is solved for the particles

The particle experiences the hydrodynamic forces due to the fluid.



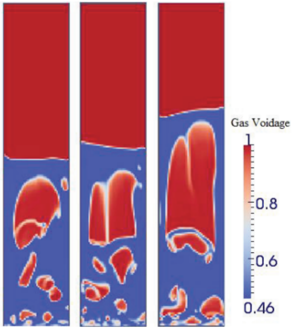
Advantages	Disadvantages
• Modeling motion of individual particles • Straightforward to incorporate physics	• **Computationally demanding**

Conventional models

Two-Fluid model (TFM)

- Both phases are considered as continua
- Conservation equations are solved for both phases

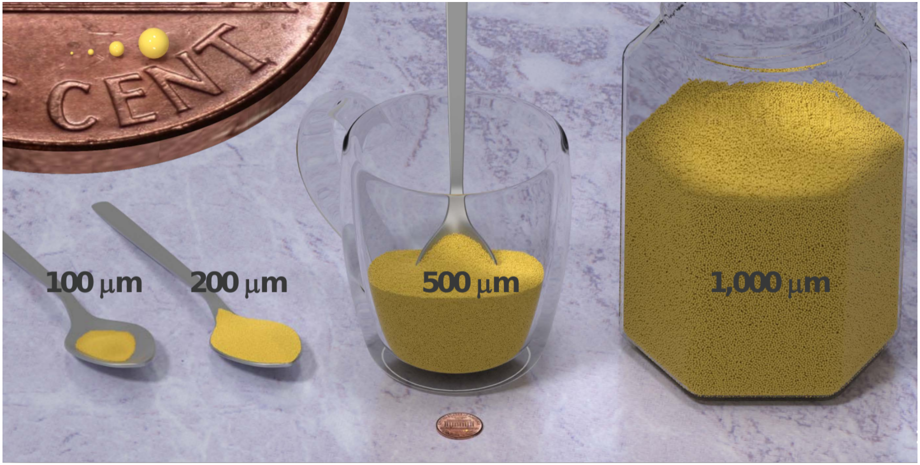
Constitutive relations are implemented to calculate the particle stresses and to close the set of equations.



Advantages	Disadvantages
• Computationally more efficient	• **Requires constitutive relations**

Model selection

What can be modeled with 1 million particles?



TFM is a better choice for modelling most of lab-scale and almost all of pilot-scale bed sizes [2].

General formulation of the TFM

Considering m and n are gas or particle phase:

Continuity:

$$\frac{\partial}{\partial t}(\rho_m \alpha_m) + \nabla \cdot (\rho_m \alpha_m \boldsymbol{u}_m) = 0$$

Momentum:

$$\frac{\partial}{\partial t}(\rho_m \alpha_m \boldsymbol{u}_m) + \nabla \cdot (\rho_m \alpha_m \boldsymbol{u}_m) = \underbrace{-\alpha_m \nabla p - \nabla p_m}_{\text{normal stresses}} + \underbrace{\nabla \cdot (\alpha_m \boldsymbol{\tau}_m)}_{\text{shear stresses}} + \underbrace{\rho_m \alpha_m \boldsymbol{g}}_{\text{grvitational force}} + \underbrace{K_{mn}(\boldsymbol{u}_n - \boldsymbol{u}_m)}_{\text{drag force}}$$

- **Drag** is the only interphase force considered. **Lift**, **virtual mass**, **wall lubrication**, and **interphase turbulence dispersion** forces, on the other hand, may be modelled using the `twoPhaseEulerFOAM`.
- Modelling the particles phase stress tensor and interphase forces are among the main challenges in this field of research.

twoPhaseEulerFoam

The twoPhaseEulerFoam is a solver for a system of two compressible fluid phases with one dispersed phase.

It is a good candidate for simulating **bubble columns** and **fluidized beds** in gas-liquid and gas-particle systems, respectively.

It solves the energy equation for both phases and couples them using one-film resistance heat transfer model.

Solver capabilities

- Both laminar and turbulence models may be utilised for the fluid phase.
- Inviscid solid phase assumption or a viscous solid phase based on the Kinetic theory of granular flows may be utilised for the solid phase.

The algorithm of twoPhaseEulerFoam

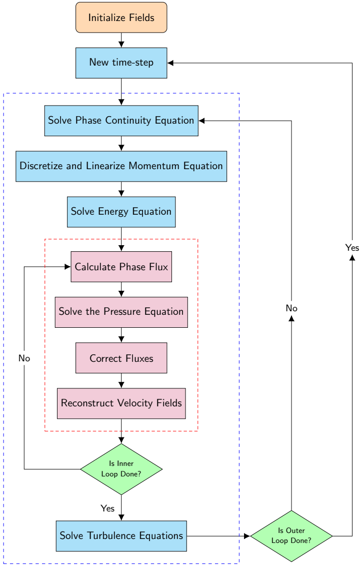
twoPhaseEulerFoam.C

```
Info<< "\nStarting time loop\n" << endl;

while (runTime.run())
{
   ...
    ++runTime;
    Info<< "Time = " << runTime.timeName() << nl << endl;

    // --- Pressure-velocity PIMPLE corrector loop
    while (pimple.loop())
    {
        fluid.solve();
        fluid.correct();

        #include "contErrs.H"
        ...
        #include "pU/UEqns.H"
        #include "EEqns.H"
        #include "pU/pEqn.H"
        #include "pU/DDtU.H"
    ...
        if (pimple.turbCorr())
        {
            fluid.correctTurbulence();
        }
    }
    ...
}
```


Initialize Fields
New time-step
Solve Phase Continuity Equation
Discretize and Linearize Momentum Equation
Solve Energy Equation
Calculate Phase Flux
Solve the Pressure Equation
Correct Fluxes
Reconstruct Velocity Fields
Is Inner Loop Done?
No
Yes
Solve Turbulence Equations
Is Outer Loop Done?
No
Yes

Continuity equation

- Continuity is achieved by solving the continuity equation of the solid phase.
- To keep the solid phase continuity equation highly linked with the pressure and momentum equations, it is not solved in its original format.

Adding zero-sum terms to the original equation and doing some manipulation:

$$\frac{\partial \alpha_s}{\partial t} + \nabla \cdot (\alpha_s \alpha_g \boldsymbol{u}_r) + \nabla \cdot (\alpha_s \boldsymbol{u}) = \alpha_s \nabla \cdot \boldsymbol{u} + \alpha_g \nabla \cdot \boldsymbol{u}_s - \alpha_s \nabla \cdot \boldsymbol{u}_g$$
$$\boldsymbol{u}_r = (\boldsymbol{u}_s - \boldsymbol{u}_g), \quad \boldsymbol{u} = \alpha_s \boldsymbol{u}_s + \alpha_g \boldsymbol{u}_g$$

This equation is solved using the **MULES**, Multi-dimentional Universal Limited Explicit Solver, that solves a convective-only transport equation and guarantee boundedness in the solution.

twoPhaseSystem.C

```
...
MULES::explicitSolve
    (
        geometricOneField(),
        alpha1,
        phi_,
        alphaPhic10,
        (alphaSubCycle.index()*Sp)(),
        (Su - (alphaSubCycle.index() - 1)*Sp*alpha1)(),
        UniformField<scalar>(phase1_.alphaMax()),
        zeroField()
    );
...
```

Further discussion on the MULES and how it solves the continuity equation can be found in project of *Alessandro Manni*, 2014.

Momentum equation

- The twoPhaseEulerFoam merely sets the momentum equations up and does not solve them.
- The solution of velocity is not necessary as the flux across the face is predicted and then corrected by the new pressure.

The momentum equations is discretized and linearized into the form outlined by Patankar [3]:

$$A_P \boldsymbol{u}_P = \sum A_{nb} \boldsymbol{u}_{nb} + b - \alpha \nabla P_{rgh}$$

$P_{rgh} = P - \rho g h$ is the modified pressure that is shared between the phases and excludes the static pressure. The following linearization as written in Rusche [4] is employed:

$$\boldsymbol{u}_i = \mathcal{A}_i^{-1} \boldsymbol{\mathcal{H}}_i - \mathcal{A}_i^{-1} \alpha_i \nabla P_{rgh}$$

Pressure equation

The $\boldsymbol{\mathcal{H}}$ vector and $\mathcal{A}$ matrix of linearized momentum equations must be interpolated to the face in order to solve the pressure equation, whereas the pressure is determined on the cell centre [5].
Using the continuity equation and the linearized momentum equation, the total pressure equation is obtained as:

$$\nabla \cdot \left(\frac{\boldsymbol{\mathcal{H}}}{\mathcal{A}}\right) - \nabla^2 \left(\left(\alpha_s^2 \mathcal{A}_s^{-1} + \alpha_g^2 \mathcal{A}_g^{-1}\right) P_{rgh}\right) + \frac{\alpha_s}{\rho_s}\frac{D\rho_s}{Dt} + \frac{\alpha_g}{\rho_g}\frac{D\rho_g}{Dt} = 0$$

Solving the pressure equation iteratively, the face fluxes are reconstructed and each phase velocity is computed respectively.

Wall treatment of particle phase

Note that:

- There is no a general consensus of modelling the wall and particles phase interaction in gas-particles flows; **No-slip**, **free-slip** and **partial-slip** boundary conditions are utilised. However, the partial-slip boundary condition appears to be the most realistic of the three as particles tend to slide over the wall.
- Various wall boundary conditions have been proposed in the literature to account for the partial slide of the particles.

In this study, the **Schneiderbauer et al.** model as an example of a partial-slip boundary condition, was implemented in `twoPhaseEulerFoam`.

Schneiderbauer model

- It is among the newest proposed models for wall treatment of particles.
- It includes a sliding and sticking collisions into a single formula.
- It is capable of taking an unsteady condition of moving walls to account.

Model of Schneiderbauer et al.

Wall shear stress:

$$\mu_s \frac{\partial u_{sl}}{\partial x} = -\frac{1}{2}\mu_w \alpha_s \rho_s g_0 \left(1+e_w\right) \Theta_s \mathrm{erf}\left(\overline{u}\right) \frac{u_{sl}}{\|u_{sl}\|},$$

Particle granular temperature flux:

$$\begin{aligned}\kappa_s \frac{\partial \Theta_s}{\partial x} = -\frac{\alpha_s \rho_s g_0 \left(1+e_w\right)\Theta_s \sqrt{3\Theta_s}}{2\sqrt{6\pi}} \Bigg\{ & 2\left(e_w - 1\right) + 3\mu_w^2 \left(1+e_w\right) \\ & + \frac{\mu_w^2}{\overline{\mu}_0{}^2} \exp\left(-\overline{u}^2\right) \Big[6r^2 \left(1 + e_w - \overline{\mu}_o\right) \\ & + 7\left(1+e_w\right) - 4\frac{\overline{\mu}_0}{\mu_w}\left(1+\mu_w\right) - 3\overline{\mu}_0^2 \left(1+e_w\right)\Big]\Bigg\}\end{aligned}$$

$$\overline{\mu}_0 = \frac{\mu_0}{1+\beta_0}, \quad \mu_0 = \frac{7}{2}(1+e_w)\mu_w, \quad \overline{u} = \sqrt{\frac{3}{2}}\frac{r}{\overline{\mu}_0}, \quad r = \frac{\|u_{sl}\|}{\sqrt{3\Theta_s}},$$

β_0: Tangential restitution coefficient, μ_w: Wall friction coefficient, e_w: Restitution coefficient, u_{sl}: Slip velocity

Implementation of the model

- The phaseCompressibleTurbulenceModels directory that contains the Johnson and Jackson model is copied to the user src directory. Assuming the terminal is opened in the user src directory.

```
    cp -r $FOAM_SRC/phaseSystemModels/twoPhaseEuler/phaseCompressibleTurbulenceModels .
```

- The phasePressureModel directory and phaseCompressibleTurbulenceModels.C are deleted from the phaseCompressibleTurbulenceModels.
- Navigating to the kineticTheoryModels directory, all directories are deleted except the derivedFvPatchFields.
- Navigating to the derivedFvPatchFields directory, the JohnsonJacksonParticleSlip and JohnsonJacksonParticleTheta directories are renamed to SchneiderBauerParticleSlip and SchneiderBauerParticleTheta, respectively.
- Going into the SchneiderBauerParticleSlip and SchneiderBauerParticleTheta directories, the JohnsonJackson keyword is changed to the SchneiderBauer in all included files names. Replace every occurrence of "JohnsonJackson" keyword with "SchneiderBauer" in .H and .C files.

Rearranging the wall shear stress equation

The wall shear stress model can be implemented in the code as a `partialSlipFvPatchVectorField`.

Boundary slip velocity: $$\underbrace{u_m}_{\text{velocity on wall}} = \left(1 - \frac{c}{c+\Delta}\right)\left(\boldsymbol{I} - (\overrightarrow{n} \otimes \overrightarrow{n})\right) \cdot \underbrace{\boldsymbol{u_c}}_{\text{velocity on adjacent cell center}}$$

c is the slip value fraction, Δ is the reciprocal of distance between the points and `valueFraction` $= \dfrac{c}{c+\Delta}$.

$$(\boldsymbol{I} - \overrightarrow{n} \otimes \overrightarrow{n}) \cdot \boldsymbol{u_c} = \boldsymbol{u_c} - (\boldsymbol{u_c} \cdot \overrightarrow{n}) \cdot \overrightarrow{n}$$

Rearranging the wall shear stress equation

Extracting slip value fraction to calculate the `valueFraction`:

Wall shear stress:

$$\mu_s \frac{\partial u_{sl}}{\partial x} = \underbrace{-\frac{1}{2} \frac{\mu_w \alpha_s \rho_s g_0 \left(1 + e_w\right) \Theta_s \operatorname{erf}\left(\overline{u}\right)}{\|u_{sl}\|}}_{c\text{: Slip value fraction}} u_{sl}$$

Slip value fraction:

$$c = -\frac{1}{2} \frac{\mu_w \alpha_s \rho_s g_0 \left(1 + e_w\right) \Theta_s}{\|u_{sl}\|} \operatorname{erf} \underbrace{\left(\overline{u}\right)}$$

$$\texttt{uBar\_} = \frac{(1 + \beta_0)\, U_{sl} \sqrt{1.5}}{3.5(1 + e_w)\mu_w \sqrt{3\Theta_s}}$$

Implementation of the wall shear stress in the code

1. Declaration of the new variables and modification of the constructors

SchneiderBauerParticleSlipFvPatchVectorField.H

```
...
class SchneiderBauerParticleSlipFvPatchVectorField
:
    public partialSlipFvPatchVectorField
{
    // Private data

        //- Intermediate variables
        dimensionedScalar mu0_;

        //- Tangential Restitution Coefficient
        dimensionedScalar betha0_;

        //- Particle-wall restitution Coefficient
        dimensionedScalar restitutionCoefficient_;

        //- Wall friction coefficient
        dimensionedScalar muw_;


public:

    //- Runtime type information
    TypeName("SchneiderBauerParticleSlip");
    ...
```

2. Redefinition of the constructors

SchneiderBauerParticleSlipFvPatchVectorField.C

```
...
Foam::SchneiderBauerParticleSlipFvPatchVectorField::
SchneiderBauerParticleSlipFvPatchVectorField
(
    const fvPatch& p,
    const DimensionedField<vector, volMesh>& iF,
    const dictionary& dict
)
:
    partialSlipFvPatchVectorField(p, iF),
    betha0_("tangentialRestitutionCoefficient", dimless,
       dict),
    restitutionCoefficient_("restitutionCoefficient",
       dimless, dict),
    muw_("wallFrictionCoefficient", dimless, dict),
    {

    if
    (
        (betha0_.value() < 0)
     || (betha0_.value() > 1)
    )
    ...
}
    ...
```

Implementation of the wall shear stress in the code

3. Definition of new variables required to calculate the slip velocity

SchneiderBauerParticleSlipFvPatchVectorField.C

```
...
//- Intermediate Field
    const scalarField uBar_
    (
    (scalar(1)+betha0_.value())*
    mag(U)*
    sqrt(1.5)/
    (3.5*
    (scalar(1)+restitutionCoefficient_.value())*
    muw_.value()*
    sqrt(3*Theta))
    );
    // calculate the slip value fraction
    scalarField c
    (
    erf(uBar_)*
    0.5*
    (scalar(1)+restitutionCoefficient_.value())
    *alpha
    *gs0
    *Theta
    *muw_.value()
    /max((nu-nuFric)*mag(U), SMALL)
    );
    this->valueFraction() = c/(c + patch().deltaCoeffs());
    partialSlipFvPatchVectorField::updateCoeffs();
    ...
```

Slip value fraction:

$$\mathtt{c} = -\frac{1}{2}\frac{\mu_w \alpha_s \rho_s g_0 \left(1+e_w\right)\Theta_s}{\|u_{sl}\|}\operatorname{erf}\underbrace{(\overline{u})}_{\mathtt{uBar\_}=\frac{(1+\beta_0)U_{sl}\sqrt{1.5}}{3.5(1+e_w)\mu_w\sqrt{3\Theta_s}}}$$

Value fraction:

$$\mathtt{valueFraction} = \frac{c}{c+\Delta}$$

Rearranging the granular energy flux equation

The granular energy flux model can be implemented in the code as a `mixedFvPatchVectorField`.

Mixed (Robin) boundary condition is a linear blend of fixed value and gradient boundary conditions.

Mixed boundary condition:

$$\underbrace{\Theta_m}_{\text{Granular temperature on wall}} = \texttt{valueFraction} \times \texttt{refValue} + (1 - \texttt{valueFraction}) \times (\underbrace{\Theta_c}_{\text{Granular temperature on adjacent cell center}} + \texttt{refGrad} \times \Delta)$$

Rearranging the granular energy flux equation

Particle granular temperature flux:

$$\kappa_s \frac{\partial \Theta_s}{\partial x} = -\frac{\alpha_s \rho_s g_0 (1+e_w)\Theta_s \sqrt{3\Theta_s}}{2\sqrt{6\pi}} \Big\{ 2(e_w - 1) + 3\mu_w^2 (1+e_w) + \frac{\mu_w^2}{\overline{\mu}_0{}^2} \exp\left(-\overline{u}^2\right) \Big[6r^2 (1 + e_w - \overline{\mu}_o) + 7(1+e_w) - 4\frac{\overline{\mu}_0}{\mu_w}(1+\mu_w) - 3\overline{\mu}_0^2 (1+e_w) \Big] \Big\}$$

Adopting the Schneiderbauer model's flux of granular energy to be utilised as a `mixed` boundary condition:

$$\frac{\partial \Theta_s}{\partial x} = \underbrace{-\frac{K_2 K_3 U_{sl}^2}{3} \frac{\alpha_s g_0 (1+e_w)\sqrt{3\Theta_s}}{2\sqrt{6\pi}\kappa_s'}}_{c \cdot \texttt{refValue}} - \underbrace{(K_1 + K_2 K_4) \frac{\alpha_s g_0 (1+e_w)\sqrt{3\Theta_s}}{2\sqrt{6\pi}\kappa_s'}}_{c} \Theta_s$$

$$\texttt{c} = (K_1 + K_2 K_4) \frac{\alpha_s g_0 (1+e_w)\sqrt{3\Theta_s}}{2\sqrt{6\pi}\kappa_s'} \qquad \texttt{refValue} = -\frac{K_2 K_3 U_{sl}^2}{3(K_1 + K_2 K_4)}$$

$$\texttt{valueFraction} = \frac{c}{c + \Delta} \qquad \texttt{refGrad} = 0$$

$\boldsymbol{K_1}$, $\boldsymbol{K_2}$, $\boldsymbol{K_3}$, and $\boldsymbol{K_4}$ are intermediate variables based on the parameters of wall-particles interaction.

Implementation of the wall granular energy flux

1. Declaration of the new variables

SchneiderBauerParticleThetaFvPatchScalarField.H

```
...
class SchneiderBauerParticleThetaFvPatchScalarField
:
    public mixedFvPatchScalarField
{
    // Private data

        //- Tangential restitution coefficient
        dimensionedScalar betha0_;

        //- Wall friction coefficient
        dimensionedScalar muw_;

        //- Particle-wall restitution coefficient
        dimensionedScalar restitutionCoefficient_;


        public:

    //- Runtime type information
    TypeName("SchneiderBauerParticleTheta");
    ...
```

2. Modification of the constructors

SchneiderBauerParticleThetaFvPatchScalarField.C

```
...
Foam::SchneiderBauerParticleThetaFvPatchScalarField::
SchneiderBauerParticleThetaFvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF,
    const dictionary& dict
)
:
    mixedFvPatchScalarField(p, iF),
    betha0_("tangentialRestitutionCoefficient", dimless,
       dict),
    muw_("wallFrictionCoefficient", dimless, dict),
    restitutionCoefficient_("restitutionCoefficient",
       dimless, dict),

{
    if
    (
        (restitutionCoefficient_.value() < 0)
     || (restitutionCoefficient_.value() > 1)
    )
     FatalErrorInFunction
            << "The restitution coefficient has to be
       between 0 and 1" << abort(FatalError);
    ...
```

Implementation of the wall granular energy flux

3. Definition of the intermediate variables

SchneiderBauerParticleThetaFvPatchScalarField.C

```
    ...
    scalar mu0bar_
    (
        3.5*
        (scalar(1)+restitutionCoefficient_.value())*
        muw_.value()/
        (scalar(1)+betha0_.value())
    );

    scalarField uBar_
    (
        sqrt(1.5)*
        mag(U)*
        (scalar(1)+betha0_.value())/
        max(3.5*
        (1+restitutionCoefficient_.value())*
        muw_.value()*
        sqrt(max(3*Theta,SMALL)),SMALL)
    );

    scalar k1_
    (
        2*(restitutionCoefficient_.value()-scalar(1))
        + 3*magSqr(muw_.value())
        *(1+restitutionCoefficient_.value())
    );
```

```
    scalarField k2_
    (
        magSqr(muw_.value()/mu0bar_)
        *exp(-magSqr(uBar_))
    );

    scalar k3_
    (
        6*
        (1+restitutionCoefficient_.value()-mu0bar_)
    );

    scalar k4_
    (
        7*(1+restitutionCoefficient_.value())
        -4*mu0bar_/
        muw_.value()*
        (1+muw_.value())
        -3*magSqr(mu0bar_)*
        (1+restitutionCoefficient_.value())
    );
...
```

Implementation of the wall granular energy flux

4. Definition of the new variables required to calculate granular energy flux

SchneiderBauerParticleThetaFvPatchScalarField.C

```
...
  // calculate the reference value and the value fraction
    this->refValue() =
        (-1)*k2_
        *k3_
        *magSqr(U)
        /(3*(k1_+k2_*k4_));

    this->refGrad() = 0.0;

    scalarField c
    (
        (k1_+k2_*k4_)
        *(1+restitutionCoefficient_.value())
        *gs0
        *sqrt(3*Theta)
        *alpha
        /max(2*sqrt(6*constant::mathematical::pi)*kappa ,
       SMALL)
    );

    this->valueFraction() = c/(c + patch().deltaCoeffs());

    mixedFvPatchScalarField::updateCoeffs();
}
    ...
```

$$\mathsf{c} = (K_1 + K_2 K_4)\,\frac{\alpha_s g_0\,(1+e_w)\,\sqrt{3\Theta_s}}{2\sqrt{6\pi}\kappa_s'}$$

$$\texttt{refValue} = -\frac{K_2 K_3 U_{sl}^2}{3\,(K_1 + K_2 K_4)}$$

$$\texttt{valueFraction} = \frac{c}{c+\Delta}$$

$$\texttt{refGrad} = 0$$

Modifying the write function

The write function in both SchneiderBauerParticleSlipFvPatchVectorField.C and SchneiderBauerParticleThetaFvPatchScalarField.C should be modified in order to let the solver run in parallel or a simulation to be restarted.

SchneiderBauer...Field.C

```
    ...
    void Foam::SchneiderBauerParticleSlipFvPatchVectorField::write
(
    Ostream& os
) const
{
    fvPatchVectorField::write(os);
    os.writeEntry("tangentialRestitutionCoefficient", betha0_);
    os.writeEntry("wallFrictionCoefficient", muw_);
    os.writeEntry("restitutionCoefficient", restitutionCoefficient_);
    writeEntry("value", os);
}
    ...
```

Modifying the Make directory and compiling the code

- Go to the Make folder in phaseCompressibleTurbulenceModels directory and change the contents of files as follows.

phaseCompressibleTurbulenceModels/Make/files

```
derived = kineticTheoryModels/derivedFvPatchFields
$(derived)/SchneiderBauerParticleTheta/SchneiderBauerParticleThetaFvPatchScalarField.C
$(derived)/SchneiderBauerParticleSlip/SchneiderBauerParticleSlipFvPatchVectorField.C
LIB = $(FOAM_USER_LIBBIN)/libmyPhaseCompressibleTurbulenceModels
```

- change the contents of options as follows.

phaseCompressibleTurbulenceModels/Make/options

```
EXE_INC = \
-I$(FOAM_SRC)/phaseSystemModels/twoPhaseEuler/twoPhaseSystem/lnInclude \
-I$(LIB_SRC)/finiteVolume/lnInclude \
...
-I$(LIB_SRC)/TurbulenceModels/phaseCompressible/lnInclude \
-I$(FOAM_SRC)/phaseSystemModels/twoPhaseEuler/phaseCompressibleTurbulenceModels/lnInclude

LIB_LIBS = \
-lfiniteVolume \
...
-lcompressibleTwoPhaseSystem \
-lphaseCompressibleTurbulenceModels
```

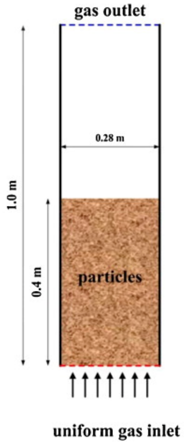
- Execute wmake in the phaseCompressibleTurbulenceModels directory.

Test case description

Characteristics of the simulated fluidized bed.

Parameter	Value
Bed width, m	0.28
Bed height, m	1.0
Bed depth, m	0.025
Initial bed height	0.4
Initial solid packing	0.6
Superficial gas velocity, m/s	0.38
Minimum fluidization velocity, m/s	0.065
Gas density, kg/m^3	Ideal Gas EOS
Gas kinematic viscosity, m^2/s	1.4×10^{-5}
Particle diameter, m	2.75×10^{-4}
Particle density, kg/m^3	2500
Particle–particle restitution coefficient	0.8
Specularity coefficient	0.5
Maximum packing limit, $\varepsilon_s^{\max}$	0.65
minimum frictional solid volume fraction, $\varepsilon_s^{\min}$	0.50

- A uniform gas velocity is employed at the inlet.
- A zero velocity is assigned to the solid phase at the inlet.
- No-slip Condition is applied for the gas phase on the wall.
- The models of Johnson-Jackson and Schneiderbauer et al. are applied for the solid phase at the wall.



Schematic of the bed [6].

Employing the model in the test case

The **wall shear stress** and the **granular energy flux** of the model of Schneiderbauer et al. are called separately in `0/U.particles` and `0/Theta.particles`, respectively.

0/U.particles

```
...
walls
{
   type                       SchneiderBauerParticleSlip;
    value                      $internalField;
    tangentialRestitutionCoefficient 0.33;
    restitutionCoefficient  0.8;
    wallFrictionCoefficient 0.3;

}
...
```

0/Theta.particles

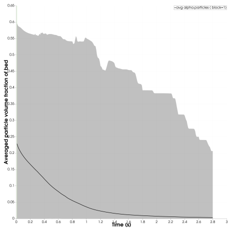
```
...
walls
{
   type                       SchneiderBauerParticleTheta;
    value                uniform 1e-4;
    tangentialRestitutionCoefficient 0.33;
    wallFrictionCoefficient 0.3;
    restitutionCoefficient  0.8;

}
...
```

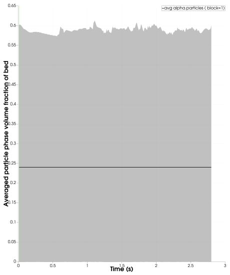
Results computed by `OpenFOAM-v2006`

The mean volume fraction of particles in the bed decreases in `OpenFOAM-v2006`, which is an unusual behaviour.

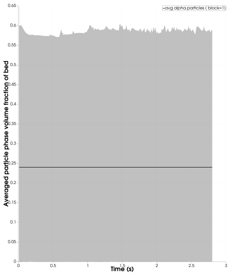
- Apparently, this issue appears to be present in versions 1912, 2006, 2012.
- It should be noted that the case was simulated using the **Johnson and Jackson** model that is originally included in the `twoPhaseEulerFoam` solver.
- Although the source of the problem is not pinpointed, It can be asserted that it occurs when partial slip models, such as the ones used in this study, are implemented for particle-wall treatment.



(a) `OpenFOAM-v2006`



(b) `OpenFOAM-v2106`



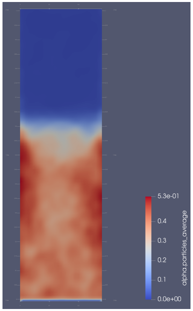
(c) `OpenFOAM-v8`

Figure: Averaged solid phase volume fraction of bed over time.

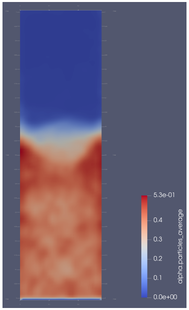
Results computed by OpenFOAM-v2106

The variables were time-averaged over 40 seconds after the initial 5 seconds of the simulation to minimize transient start-up effects.

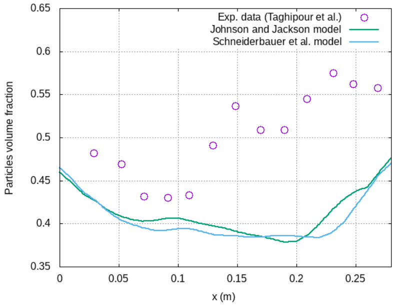
- The concentration of particles near the wall and overall bed behavior predicted by the models are different.
- Symmetrical time-averaged profiles are expected due to the symmetry of the geometry and boundary conditions. The model of Schneiderbauer et al. shows a better symmetry of the the profile in the bed.



(a) Johnson and Jockson



(b) Schneiderbauer et al.

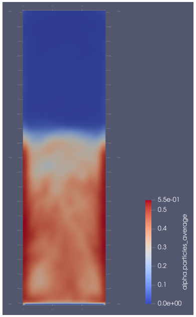


(c) Time-averaged particles volume fraction

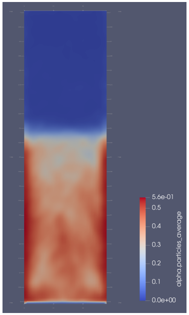
Figure: (a, b):Time-averaged solid phase volume fraction of the bed, and (c): time-averaged solid volume fraction profiles predicted by two models vs. experimental data at height of 20 cm above the inlet.

Results computed by OpenFOAM-v8

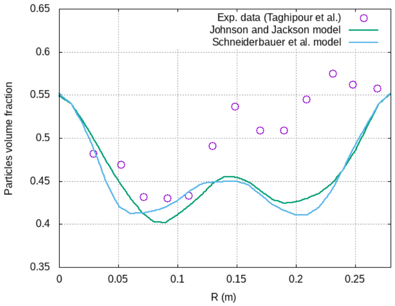
The variables were time-averaged over 40 seconds after the initial 5 seconds of the simulation to minimize transient start-up effects.



(a) Johnson and Jockson



(b) Schneiderbauer et al.



(c) Time-averaged particles volume fraction

Figure: (a, b):Time-averaged solid phase volume fraction of the bed, and (c): time-averaged solid volume fraction profiles predicted by two models vs. experimental data at height of 20 cm above the inlet.

- The results predicted by OpenFOAM-v8 is in a better agreement with the experimental data.

Conclusion

- The model of Schneiderbauer et al. was implemented successfully in `OpenFOAM`.
- The results predicted using the model of Schneiderbauer et al. were in a better agreement with the experimental data than those predicted by the model of Johnson and Jackson.
- The results predicted by `OpenFOAM-v8` conform better with the experimental data than those predicted by `OpenFOAM-v2106`.
- The `twoPhaseEulerFoam` solver of versions 1912, 2006, and 2012 do not implement the partial-slip type boundary conditions correctly.
- A thorough analysis of boundary condition models necessitates more realistic simulations with finer grids and longer simulation times, as well as a thorough post-analysis to investigate 3D bubble characteristics, velocity fields, flux of granular energy, and shear stresses near walls.

Implementation of a wall boundary condition for the solid phase in a gas-particle flow in `twoPhaseEulerFoam`

Mohsen Zarepour

Mechanical Engineering Department,
University of Saskatchewan,
Saskatoon, Canada

January 17, 2022

References I

[1] William D. Fullmer and Christine M. Hrenya. "The Clustering Instability in Rapid Granular and Gas-Solid Flows". In: *Annual Review of Fluid Mechanics* 49.1 (Jan. 2017), pp. 485–510. DOI: 10.1146/annurev-fluid-010816-060028.

[2] Jeff Dietiker. *MFiX suite Development Update*. Tech. rep. National Energy Technology Laboratory (NETL), Pittsburgh, PA, Morgantown, WV . . ., 2019.

[3] BR Baliga and SV Patankar. "A new finite-element formulation for convection-diffusion problems". In: *Numerical Heat Transfer* 3.4 (1980), pp. 393–409.

[4] Henrik Rusche. "Computational fluid dynamics of dispersed two-phase flows at high phase fractions". PhD thesis. Imperial College London (University of London), 2003.

[5] CM Rhie and W Li Chow. "Numerical study of the turbulent flow past an airfoil with trailing edge separation". In: *AIAA journal* 21.11 (1983), pp. 1525–1532.

References II

[6] Yefei Liu and Olaf Hinrichsen. "CFD modeling of bubbling fluidized beds using OpenFOAM®: Model validation and comparison of TVD differencing schemes". In: *Computers & chemical engineering* 69 (2014), pp. 75–88.