# Documentation of open-source MFIX–DEM software for gas-solids flows[1]

R. Garg
rahul.garg@gmail.com
National Energy Technology Laboratory
Morgantown, WV, USA

J. Galvin
janine.galvin@netl.doe.gov
National Energy Technology Laboratory
Albany, OR, USA

T. Li
litingwen@gmail.com
National Energy Technology Laboratory
Morgantown, WV, USA

S. Pannala
pannalas@ornl.gov
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, 37831, USA

February 1, 2010

[1]Refer to this document as: R. Garg, J. Galvin, T. Li, and S. Pannala Documentation of open-source MFIX–DEM software for gas-solids flows, From URL https://mfix.netl.doe.gov/documentation/dem_doc_2010.pdf

## Table of Contents

# 1 Introduction and Background

Multiphase flows are commonly observed in nature, such as rain drops in air, snowfall, volcanoes, and sandstorms, and in various industries, such as energy production, chemical processing, and pharmaceuticals. Examples include life-saving flu vaccines which may be delivered to the human body in the form of aerosols, or as a fine powder, and internal combustion engines where finely-atomized fuel spray is injected into compressed air for efficient combustion and hence, less atmospheric pollution. Two-phase flow of gas and liquid exists in oil–gas pipelines and wells, oil refineries, air-lift pumps and steam boilers. Given their ubiquity, a better understanding of the physical phenomena occurring in multiphase flows is needed so that current applications can be made more efficient and environmentally friendly. The scope of the open source code described herein is limited to two-phase flows consisting of dispersed solid particles (differing in size and density) in a carrier phase that may be either a liquid or gas. The following discussion on statistical descriptions, however, is more broad and concerns two-phase flows where the dispersed phase may be solid, liquid or gas, and the carrier phase is liquid or gas.

Two-phase flows are inherently random in nature due to the presence of embedded particles. Therefore a two-phase flow cannot be meaningfully characterized with only *one realization* (Drew, 1983). Consequently, two-phase flows are typically described using statistical techniques. The most common statistical descriptions for two–phase flows can be classified into two broad categories: (i) Eulerian–Eulerian (EE) and (ii) Lagrangian–Eulerian (LE) representations. In the EE representation, the two phases are assumed to be interpenetrating continua. A continuum description, which is typically comprised by a set of conservation equations, such as a mass and momentum balances, is adopted for both the carrier phase and the dispersed phase (e.g., solid particles, droplets, bubbles). Various averaging approaches have been applied to obtain the continuum conservation equations. The earliest averaging techniques consist of both time and space averaging (and its variants based on the choice of averaging region) (Frankl, 1953; Teletov, 1958; Anderson and Jackson, 1967; Drew, 1971). Later the concept of ensemble averaging (Drew, 1983; Kataoka and Serizawa, 1989; Drew and Passman, 1998), which is defined as the process of averaging a quantity over several independent realizations, was applied to derive the continuum equations.

In the LE statistical description (Williams, 1958; Subramaniam, 2000), a continuum description is employed for the carrier phase and its form is generally identical to that in the EE representation. In contrast, the dispersed phase is treated as being composed of discrete entities (or particles [1]) which can be statistically represented by the single particle distribution function. In spray literature this function is referred to as the droplet distribution function (ddf) and evolution of the ddf results in the famous spray equation (Williams, 1958). It is worth noting that by taking moments of the evolution equation, the single particle distribution function can also be used to derive the continuum conservation equation for the solid-phase. The kinetic theory for granular flows (KTGF) (Savage and Jeffrey, 1981; Garzo et al., 2007) follows this approach for describing solid particulates. In particular, continuum equations for solid-phase are obtained and constitutive relations for the transport coefficients (such as the coefficients of viscosity, thermal diffusion coefficients, etc.) can be found in terms of hydrodynamic variables (such as particle number density, inelasticity, granular temperature, etc.).

From a numerical viewpoint, statistical descriptions notwithstanding, simulations that solve the continuum conservation equations for both phases have traditionally been called as EE, Euler-granular, or two fluid method (TFM) simulations. On the other hand, simulations that consider the carrier phase as a continuum and the dispersed phase as made up of discrete entities have

[1] In this context particle may mean any dispersed–phase element, including solid particles, droplets or bubbles.

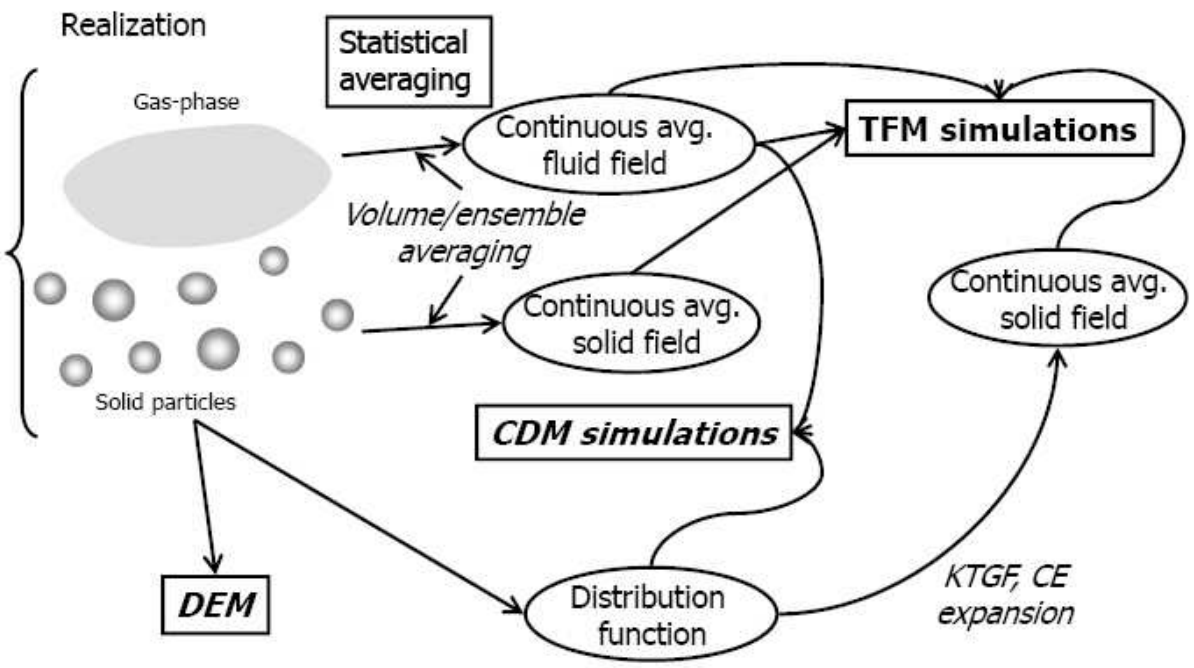



Figure 1: A schematic showing a realization of two–phase flow along with various statistical descriptions and resulting simulation types.

traditionally been called LE simulations. As discussed above and also shown by the schematic in Fig. 1, continuum conservation equations for the dispersed phase can also be obtained from the Lagrangian description. Therefore, calling all simulations that solve continuum conservation equations for both phases as EE fails to distinguish between the underlying statistical descriptions (LE or EE). Therefore, to distinguish statistical descriptions from numerical viewpoint, simulations that consider both phases as continua will hereinafter be referred to as TFM simulations, while simulations that consider the carrier phase as a continuum and the dispersed phase as discrete particles (such as simulations of the current effort) will be referred to as continuum discrete method (CDM).

A number of open source and commercial codes are capable of doing both TFM and CDM simulations. For example, commercially available codes like Fluent, and open source codes like CFDlib (Kashiwa and Rauenzahn, 1994), OpenFoam, and MFIX (Syamlal et al., 1993; Syamlal, 1998), are all capable of performing TFM simulations for chemically reacting multiphase flows. Similarly, commercially available codes like Fluent and Barracuda, and open sources codes like MFIX–DEM, KIVA (Amsden et al., 1989), Fluent DPM (Discrete Particle Method) and dense-phase DPM modules, and OpenFoam, are all capable of CDM simulations. In regard to TFM simulations, while all the above-mentioned codes solve for similar forms of the governing equations, they primarily differ in their closures for various submodels (such as solid stresses, interphase drag, etc.) and their numerical treatment. In regard to CDM simulations, like the TFM simulations all the codes solve similar forms of the governing equations for the carrier phase (with differences in numerical treatment, closure models, etc). For the dispersed phase, however, all codes except

for MFIX–DEM employ a parcel-based (also called as computational/notional/nominal particles based approach in literature) approach. In the parcel approach a finite number of parcels are tracked rather than using actual individual particles as dictated by the number density of the solid-phase. Each parcel may represent either a fractional number of real particle or many real particles grouped together to form a single parcel. For example, in very dilute regions of spray applications, many parcels are used to represent one real particle in order to mitigate the high statistical errors that would be associated with very few real particles. On the other extreme, in very dense fluidized bed like applications or device-scale problems, many real particles are grouped together and are represented by a single parcel in order to reduce the high computational cost with tracking the real number of particles. Since the particles in this approach are represented by statistically weighted parcels, the collisions between parcels (unlike collisions between particles) cannot be directly resolved necessitating the use of indirect collision models. For example stochastic collision models, such as the droplet collision algorithm of O'Rourke and Amsden (1987) (used in Fluent DPM and OpenFoam) or the less expensive (but similar) no time counter algorithm of Bird (1994), have been used for calculating collisions in CDM simulations of *dilute* gas-solids flows. For CDM simulations of *dense* gas-solids flows, the collisions between parcels have been modeled by an ad-hoc solid stress term that prevents the solids from over packing (e.g., Fluent's dense-phase DPM and Barracuda). In MFIX–DEM code (for CDM simulations) the dispersed phase is represented by actual individual particles and the collisions are directly resolved using the soft-sphere (based on a spring-dashpot model) approach of Cundall and Strack (1978). While simulations using MFIX–DEM are limited to small problem sizes due to high computational cost incurred in the particle neighbor search algorithm, this approach (using actual particles) does serve as a good tool to verify and also develop new closures for various submodels used in TFM simulations.

The above discussion focused on TFM and CDM simulations, which are designed to model two-phase flows having both a carrier and disperse phase. Advanced codes are also available for studying pure particulate (or granular) flows in the absence of a carrier phase. In all such codes the solid-phase is represented by actual particles and collisions are directly resolved. Such simulations are generally referred to as Discrete Element Method (DEM) simulations (cf. Fig. 1). Examples of DEM simulation codes include open source codes, such as LAMMPS (Silbert et al., 2001) and Yade (Galizzi and Kozicki, 2005), and commercial codes, such as EDEM (http:www.dem-solutions.comindex.php) and Itasca (Itasca, 2010). Efforts to couple such standalone DEM codes to existing CFD solvers have recently been undertaken with the goal to leverage the combined abilities that were originally developed for each code individually. For example, EDEM code provides users the ability to couple its DEM modules with other CFD codes such as Fluent. Very recently OpenFoam has been coupled to Yade (Chen, 2009) and LAMMPS. However, these OpenFoam codes for CDM type simulations are still in the beta stage and are unavailable to the CFD community. Although, EDEM, a commercial code, provides coupling hooks with other commercial CFD codes such as Fluent, the inability of users to readily understand and modify the source code limits it to mostly end/expert users.

The open source MFIX–DEM code can be used for DEM, CDM, and TFM simulations from a single source code. A basic structure for DEM and CDM simulations has existed in MFIX for several years. Despite this time and even though MFIX provides an excellent opportunity to run different statistical descriptions from one software platform, MFIX has not been as widely used for CDM and DEM simulations as it has for TFM simulations. We are building on the MFIX–DEM developments that resulted in two theses (Boylakunta, 2003; Weber, 2004) and the previous documentation (Boyalakuntla and Pannala, 2006).

The MFIX–DEM code has recently been extensively debugged. The spring-dashpot model has been rigorously verified by performing a series of mple tests, such as a freely falling particle, two

stacked particles compressed between two boundaries, and a particle sliding on a rough surface. The purpose of these simple tests is to verify independently each component of the spring-dashpot model. For the CDM simulations, qualitative and quantitative analysis of particles in vortex flows has been performed to validate the accuracy of gas-solids coupling. This report documents the current MFIX–DEM code along with pointers to the code and a discussion on the theory. The document will continue to evolve as more verification/validation cases are added and new features are incorporated. If you have any comments or suggestions, please feel free to send them to the authors so that we can continue to improve the code capabilities and the documentation so that the code can be easily used and reviewed.

In the next section, the details of CDM and DEM simulations are provided in a manner that is consistent with the MFIX–DEM implementation.

# 2 Governing Equations

In MFIX–DEM, the gas–phase governing equations for mass and momentum conservation are similar to those in traditional gas–phase CFD but with additional coupling terms due to drag from the solids–phase. The solids-phase is modeled using discrete particles. The current implementation of MFIX–DEM is restricted only to hydrodynamics (no chemistry or heat and mass transfer abilities). Work is under way to extend the DEM implementation to include these additional physics. Below is a list of the governing equations along with the numerical implementation, including the coupling procedure.

## 2.1 Gas-phase

The governing equations, implemented in MFIX (Syamlal et al., 1993), for the gas–phase continuity and momentum conservation in the absence of phase change, chemical reactions, growth, aggregation, breakage phenomena, are:

$$\frac{\partial(\varepsilon_{\rm g}\rho_{\rm g})}{\partial t} + \boldsymbol{\nabla}\cdot(\varepsilon_{\rm g}\rho_{\rm g}\mathbf{v}_{\rm g}) = 0 \quad ; \tag{1}$$

and

$$\frac{D}{Dt}(\varepsilon_{\rm g}\rho_{\rm g}\mathbf{v}_{\rm g}) = \boldsymbol{\nabla}\cdot\overline{\overline{S}}_{\rm g} + \varepsilon_{\rm g}\rho_{\rm g}\mathbf{g} - \sum_{m=1}^{M}\mathbf{I}_{{\rm g}m} \quad . \tag{2}$$

In the above equation, $\varepsilon_{\rm g}$ is the gas–phase volume fraction, $\rho_{\rm g}$ is the thermodynamic density of the gas phase, $\mathbf{v}_{\rm g}$ is the volume–averaged gas–phase velocity, $\mathbf{I}_{{\rm g}m}$ is the momentum transfer term between the gas and the $m^{\rm th}$ solid phase, and $\overline{\overline{S}}_{\rm g}$ is the gas–phase stress tensor given by

$$\overline{\overline{S}}_{\rm g} = -P_{\rm g}\overline{\overline{I}} + \overline{\overline{\tau}}_{\rm g}, \tag{3}$$

where $P_{\rm g}$ is the gas–phase pressure. Also, $\overline{\overline{\tau}}_{\rm g}$ is the gas–phase shear stress tensor,

$$\overline{\overline{\tau}}_{\rm g} = 2\mu_{\rm g}\overline{\overline{D}}_{\rm g} + \lambda_{\rm g}\boldsymbol{\nabla}\cdot{\rm tr}(\overline{\overline{D}}_{\rm g})\overline{\overline{I}}, \tag{4}$$

where $\overline{\overline{D}}_{\rm g} = \frac{1}{2}\left[\boldsymbol{\nabla}\mathbf{v}_{\rm g} + (\boldsymbol{\nabla}\mathbf{v}_{\rm g})^T\right]$ is the strain rate tensor, and $\mu_g$ and $\lambda_g$ are the dynamic and second coefficients of viscosity of the gas phase. Discussion and definition of the interphase momentum transfer term is reserved until section 2.2.4.

## 2.2 Solid-phase: Discrete Element Method (DEM)

In the DEM approach, the $m^{\text{th}}$ solid–phase is represented by $N_m$ spherical particles with each particle having diameter $D_m$ and density $\rho_{sm}$. Solid phases are differentiated based according to radii and densities. Accordingly, the diameter and density of the $m^{\text{th}}$solid–phase is denoted by $D_m$ and $\rho_{sm}$, respectively. For total of $M$ solid phases, the total number of particles is equal to $N = \sum_{m=1}^{M} N_m$. These $N$particles are represented in a Lagrangian frame of reference at time $t$ by $\{\mathbf{X}^{(i)}(t), \mathbf{V}^{(i)}(t), \boldsymbol{\omega}^{(i)}(t), D^{(i)}, \rho^{(i)}\, i = 1, \ldots, N\}$, where $\mathbf{X}^{(i)}(t)$ denotes the $i^{th}$ particle's position, $\mathbf{V}^{(i)}(t)$ and $\boldsymbol{\omega}^{(i)}$ denote its linear and angular velocities, $D^{(i)}$ denotes its diameter, and $\rho^{(i)}$ represents its density. It is implicit that if a particle belongs to $m^{\text{th}}$ solid–phase, then its diameter and density are, respectively, equal to $D_m$ and $\rho_{sm}$ (i.e., equal to the diameter and density of the $m^{\text{th}}$ solid–phase). The mass $m^{(i)}$ and moment of inertia $I^{(i)}$ of the $i^{\text{th}}$ particle are equal to $\rho^{(i)}\dfrac{\pi {D^{(i)}}^3}{6}$ and $\dfrac{m^{(i)}{D^{(i)}}^2}{10}$, respectively. The position, linear and angular velocities of the $i^{\text{th}}$ particle evolve according to Newton's laws as:

$$\frac{d\mathbf{X}^{(i)}(t)}{dt} = \mathbf{V}^{(i)}(t), \tag{5}$$

$$m^{(i)}\frac{d\mathbf{V}^{(i)}(t)}{dt} = \mathbf{F}_{\text{T}}^{(i)} = m^{(i)}\mathbf{g} + \mathbf{F}_{\text{d}}^{(i\in k,m)}(t) + \mathbf{F}_{\text{c}}^{(i)}(t), \tag{6}$$

$$I^{(i)}\frac{d\boldsymbol{\omega}^{(i)}(t)}{dt} = \mathbf{T}^{(i)} \tag{7}$$

$$\tag{8}$$

where $\mathbf{g}$ is the acceleration due to gravity, $\mathbf{F}_{\text{d}}^{(i\in k,m)}$ is the total drag force (pressure + viscous) on $i^{\text{th}}$ particle residing in $k^{\text{th}}$ cell and belonging to the $m^{\text{th}}$ solid–phase, $\mathbf{F}_{\text{c}}^{(i)}$ is the net contact force acting as a result of contact with other particles, $\boldsymbol{\eta}$ is the outward pointing normal unit vector along the particle radius, $\mathbf{T}^{(i)}$ is the sum of all torques acting on the $i^{\text{th}}$ particle, and $\mathbf{F}_{\text{T}}^{(i)}$ is the net sum of all forces acting on the $i^{\text{th}}$ particle. The next four subsections discuss in detail the calculation of the contact and drag forces.

### 2.2.1 Contact Forces

The advantage of the DEM approach over that of solving continuum equations for solid–phase lies in its explicit treatment of particle–particle collisions. For two–phase flows, hard–sphere (based on the event driven algorithm, first proposed by Allen and Tildesley (1989)) and soft–sphere (based on the spring–dashpot model, first proposed by (Cundall and Strack, 1978)) models are the two most commonly used approaches. In the hard–sphere approach, collisions are binary and instantaneous, whereas the soft–sphere approach imposes no such restrictions, thus it is possible to have enduring, multi–particle contacts. In the event driven (hard–sphere) approach, the time step is determined by the minimum collision time between any one pair of particles — which is directly proportional to the mean free path or inversely proportional to the particle volume fraction. Therefore, the hard–sphere approach is most suitable for dilute systems, since in denser systems the minimum collision time becomes much smaller than other time scales. Also, in dense regions, momentum transfer occurs more through enduring contacts (the so called quasi–static regime) than through binary collisions. Even in gas–particle systems that are nominally dilute, the preferential concentration of

particles to the high strain rate regions of gas flow can result in locally dense regions which require very small time steps to resolve. The time step in the soft–sphere approach, although small and a function of the spring stiffness, does not vary with the volume fraction. Although the hard–sphere approach may be a good alternative in some systems, the soft–sphere approach is generally more robust due to the independence of the time step size from the volume fraction.

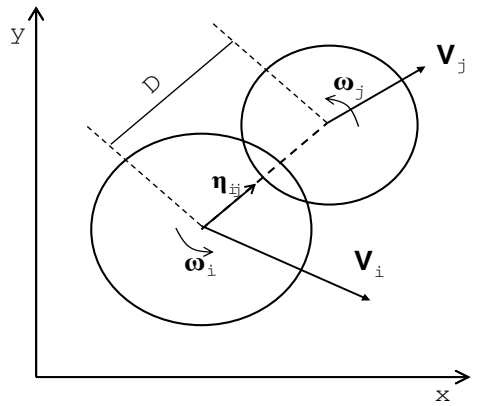



Figure 2: Schematic of two particles $i$ and $j$ having diameters $D_i$ and $D_j$ in contact. Particles have linear and angular velocities equal to $\mathbf{V}_i, \mathbf{V}_j$ and $\omega_i, \omega_j$, respectively. Overlap $\delta_{\rm n} = 0.5(D_i + D_j) - D$. $\eta_{ij}$ is the vector along the line of contact pointing from particle $i$ to particle $j$.

Below the soft–sphere collision approach implemented in MFIX–DEM code is detailed. As shown by the schematic in Fig. 2, consider two particles $i$ and $j$ in contact that have diameters equal to $D^{(i)}$ and $D^{(j)}$ and are located at $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$. The particle $i$ is moving with linear and angular velocities equal to $\mathbf{V}^{(i)}$ and $\boldsymbol{\omega}^{(i)}$, respectively. Similarly, the particle $j$ is moving with linear and angular velocities equal to $\mathbf{V}^{(j)}$ and $\boldsymbol{\omega}^{(j)}$, respectively. The normal overlap between the particles is calculated as

$$\delta_n = 0.5\left(D^{(i)} + D^{(j)}\right) - \left|\mathbf{X}^{(i)} - \mathbf{X}^{(j)}\right|. \tag{9}$$

The unit vector along the line of contact pointing from particle $i$ to particle $j$ is

$$\boldsymbol{\eta}_{ij} = \frac{\mathbf{X}^{(j)} - \mathbf{X}^{(i)}}{\left|\mathbf{X}^{(j)} - \mathbf{X}^{(i)}\right|}, \tag{10}$$

and the relative velocity of the point of contact becomes

$$\mathbf{V}_{ij} = \mathbf{V}^{(i)} - \mathbf{V}^{(j)} + \frac{1}{2}\left(L^{(i)}\boldsymbol{\omega}^{(i)} + L^{(j)}\boldsymbol{\omega}^{(j)}\right) \times \boldsymbol{\eta}_{ij}, \tag{11}$$

where $L^{(i)}$ and $L^{(j)}$ are the distance of the ontact point from the center of particles $i^{\rm th}$ and $j^{\rm th}$, respectively. They are given by

$$\mathrm{L}^{(i)} = \frac{\left|\mathbf{X}^{(j)} - \mathbf{X}^{(i)}\right|^2 + {r^{(i)}}^2 - {r^{(j)}}^2}{2\left|\mathbf{X}^{(j)} - \mathbf{X}^{(i)}\right|}, \tag{12}$$

and

$$\mathrm{L}^{(j)} = \left|\mathbf{X}^{(j)} - \mathbf{X}^{(i)}\right| - \mathrm{L}^{(i)}, \tag{13}$$

where $r^{(i)} = 0.5D^{(i)}$ and $r^{(j)} = 0.5D^{(j)}$ are the particle radii.

Therefore, the normal $\mathbf{V}_{\mathrm{n}ij}$ and tangential $\mathbf{V}_{\mathrm{t}ij}$ components of contact velocity, respectively, are

$$\mathbf{V}_{\mathrm{n}ij} = \mathbf{V}_{ij} \cdot \boldsymbol{\eta}_{ij}\, \boldsymbol{\eta}_{ij} \equiv \left(\mathbf{V}^{(i)} - \mathbf{V}^{(j)}\right) \cdot \boldsymbol{\eta}_{ij}\, \boldsymbol{\eta}_{ij}, \tag{14}$$

and

$$\mathbf{V}_{\mathrm{t}ij} = \mathbf{V}_{ij} - \mathbf{V}_{ij} \cdot \boldsymbol{\eta}_{ij}\, \boldsymbol{\eta}_{ij}. \tag{15}$$

The tangent to the plane of contact $\boldsymbol{t}_{ij}$ is

$$\boldsymbol{t}_{ij} = \frac{\mathbf{V}_{\mathrm{t}ij}}{|\mathbf{V}_{\mathrm{t}ij}|}. \tag{16}$$

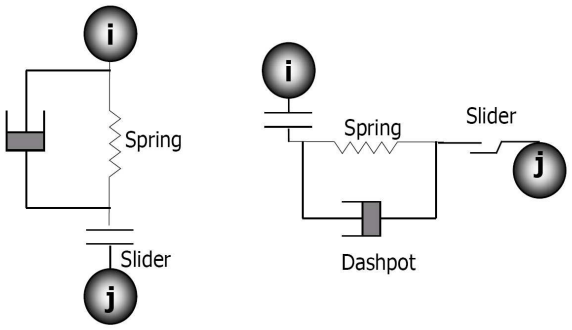



Figure 3: Schematic of the spring–dashpot system used to model particle contact forces in soft–sphere approach.

In soft–sphere approach, the overlap between the two particles is represented as a system of springs and dashpots (Fig. 3) in both normal and tangential directions. The spring causes the rebound off the colliding particles and the dashpot mimics the dissipation of kinetic energy due to inelastic collisions. The spring stiffness coefficients in the tangential and normal directions are $k_{\mathrm{t}}$ and $k_{\mathrm{n}}$, respectively. Similarly, the dashpot damping coefficients in the tangential and normal directions are $\eta_{\mathrm{t}}$ and $\eta_{\mathrm{n}}$, respectively. The spring stiffness and dashpot damping coefficients are essentially a function of the solid–phases the colliding particles belong to. For example, if the $i^{\mathrm{th}}$ particle belongs to $m^{\mathrm{th}}$ solid–phase and the $j^{\mathrm{th}}$ particle belongs to $\ell^{\mathrm{th}}$ solid–phase, then the spring stiffness coefficients are given by $k_{\mathrm{n}m\ell}$ and $k_{\mathrm{t}m\ell}$. Similarly, the dashpot damping coefficients are given by $\eta_{\mathrm{n}m\ell}$ and $\eta_{\mathrm{t}m\ell}$. However, in order to keep the formulation simple, the subscripts $(m, \ell)$ are dropped and it is noted that the spring stiffness and dashpot damping coefficients will depend on the solid–phases the colliding particles belong to.

The normal and tangential components of the contact force $\mathbf{F}_{ij}$, at time $t$, are decomposed into the spring (conservative) force $\mathbf{F}^{\mathrm{S}}_{ij}$ and the dashpot (dissipative) force $\mathbf{F}^{\mathrm{D}}_{ij}$ as

$$\mathbf{F}_{\mathrm{n}ij}(t) = \mathbf{F}^{\mathrm{S}}_{\mathrm{n}ij}(t) + \mathbf{F}^{\mathrm{D}}_{\mathrm{n}ij}(t), \tag{17}$$

and

$$\mathbf{F}_{\mathrm{t}ij}(t) = \mathbf{F}^{\mathrm{S}}_{\mathrm{t}ij}(t) + \mathbf{F}^{\mathrm{D}}_{\mathrm{t}ij}(t). \tag{18}$$

The normal spring force $\mathbf{F}^{\mathrm{S}}_{\mathrm{n}ij}$ at any time during the contact is calculated based on the overlap $\delta_n$ between the particles and is given by

$$\mathbf{F}^{\mathrm{S}}_{\mathrm{n}ij} = -k_n \delta_n \boldsymbol{\eta}_{ij}. \tag{19}$$

For tangential spring force, a time history is maintained. At the initiation of the contact the

A time history of the tangential spring force is maintained once the contact initiates. At any time during the contact, the tangential spring force is given by

$$\mathbf{F}^{\mathrm{S}}_{\mathrm{t}ij} = -k_t \boldsymbol{\delta}_t \tag{20}$$

where $\boldsymbol{\delta}_t$ is the tangential displacement. At the initiation of the contact the tangential displacement is calculated as

$$\boldsymbol{\delta}_t = \mathbf{V}_{\mathrm{t}ij} \min\left(\frac{|\boldsymbol{\delta}_n|}{\mathbf{V}_{ij}\cdot\boldsymbol{\eta}_{ij}}, \Delta t\right). \tag{21}$$

At time $(t+\Delta t)$ the tangential displacement is calculated as

$$\boldsymbol{\delta}_t(t+\Delta t) = \boldsymbol{\delta}_t(t) + \mathbf{V}_{\mathrm{t}ij}\Delta t. \tag{22}$$

In the above expression the accumulated tangential displacement $\boldsymbol{\delta}_t(t)$ at time $t$ will not necessarily lie on the tangent plane at $t+\Delta t$. Therefore, the above expression for tangential displacement is further corrected to ensure that the tangential displacement lies in the current tangent plane. The corrected tangential displacement is obtained by subtracting the normal component of $\boldsymbol{\delta}_t(t+\Delta t)$ from $\boldsymbol{\delta}_t(t+\Delta t)$ itself, which is given as

$$\boldsymbol{\delta}_t(t+\Delta t) = \boldsymbol{\delta}_t(t+\Delta t) - \left(\boldsymbol{\delta}_t(t+\Delta t)\cdot\boldsymbol{\eta}_{ij}\right)\boldsymbol{\eta}_{ij}. \tag{23}$$

For the case of finite Coulomb friction between particles [2], if the following holds at any time during the contact,

$$|\mathbf{F}_{\mathrm{t}ij}| > \mu\,|\mathbf{F}_{\mathrm{n}ij}|, \tag{24}$$

then the sliding is assumed to occur and the tangential contact force is given by

$$\mathbf{F}_{\mathrm{t}ij} = \begin{cases} -\mu\,|\mathbf{F}_{\mathrm{n}ij}|\,\mathbf{t}_{ij} & \text{if } \mathbf{t}_{ij} \neq 0 \\ -\mu\,|\mathbf{F}_{\mathrm{n}ij}|\,\frac{\boldsymbol{\delta}_t}{|\boldsymbol{\delta}_t|} & \text{if } \mathbf{t}_{ij} = 0, \boldsymbol{\delta}_t \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

It is important to note that the $i^{\text{th}}$ particle in the contact $i-j$ pair experiences a contact force equal to $\mathbf{F}_{ij}$ and the $j^{\text{th}}$ particle, according to Newton's third law of motion, experiences an equal and opposite contact force (i.e. $-\mathbf{F}_{ij}$). Therefore, the total contact force $\mathbf{F}^{(i)}_{\mathrm{c}}(t)$ at any time on the $i^{\text{th}}$ particle is given as

$$\mathbf{F}^{(i)}_{\mathrm{c}}(t) = \sum_{\substack{j=1 \\ j\neq i}}^{N} \left(\mathbf{F}^{\mathrm{S}}_{ij}(t) + \mathbf{F}^{\mathrm{D}}_{ij}(t)\right) \tag{26}$$

and the total torque acting on $i^{\text{th}}$ particle is calculated by

$$\mathbf{T}^{(i)}(t) = \sum_{\substack{j=1 \\ j\neq i}}^{N} \left(\mathrm{L}^{(i)}\boldsymbol{\eta}_{ij} \times \mathbf{F}_{\mathrm{t}ij}(t)\right) \tag{27}$$

[2] Like for the spring stiffness and dashpot damping coefficients, the friction coefficient $\mu_{m\ell}$ will also depend on the solid–phases the colliding particles belong to. However, for the sake of clarity, the subscripts are omitted in favor of just $\mu$.

### 2.2.2 Relationship between dashpot coefficients and coefficients of restitution

For collisions between particles belonging to the $m^{\text{th}}$ and $\ell^{\text{th}}$ solid–phases, the normal dashpot damping coefficient $\eta_{\text{n}m\ell}$ is related to the normal coefficient of restitution $e_{\text{n}m\ell}$ (Silbert et al., 2001) by

$$e_{\text{n}m\ell} = \exp\left(-\frac{\eta_{\text{n}m\ell}\, t^{\text{col}}_{\text{n},m\ell}}{2m_{\text{eff}}}\right), \tag{28}$$

where $m_{\text{eff}} = m_m m_\ell/(m_m + m_\ell)$ is the effective mass and $t^{\text{col}}_{\text{n},m\ell}$ is the collision time between $m$ and $\ell$ solid phases. It is given by

$$t^{\text{col}}_{\text{n},m\ell} = \pi\left(\frac{k_{\text{n}m\ell}}{m_{\text{eff}}} - \frac{\eta^2_{\text{n}m\ell}}{4m^2_{\text{eff}}}\right)^{-1/2}. \tag{29}$$

From the above two expressions, $\eta_{\text{n}ml}$ is obtained as

$$\eta_{\text{n}m\ell} = \frac{2\sqrt{m_{\text{eff}} k_{\text{n}m\ell}}\,|\ln e_{\text{n}m\ell}|}{\sqrt{\pi^2 + \ln^2 e_{\text{n}m\ell}}}, \tag{30}$$

and a similar expression can be written for $\eta_{\text{t}m\ell}$.

The time step $\Delta t$ is typically taken to be equal to one by fifty of the minimum collision time (i.e. $\Delta t = \min(t_{\text{col},m\ell}/50)$). Specification of spring stiffness coefficients in DEM simulations is problematic. If values close to the real physical values are chosen, then the time step will become very small, prohibiting any large–scale study. Therefore a value of normal spring stiffness coefficient$\sim 10^5$, is usually specified. The tangential spring stiffness coefficient is set equal to two–fifths of the normal stiffness coefficient (i.e., $k_{\text{t}ml} = 2/5k_{\text{n}}, \forall m, l$). The tangential damping coefficient is generally taken to be half of normal damping coefficient (i.e., $\eta_{\text{t}ml} = 0.5\eta_{\text{n}ml}, \forall m, l$). In gas–particle flows, since the drag force also opposes the particle velocity, a spring stiffness less than that used in pure granular flows can be utilized.

For $M$ solid–phases, the coefficients of restitution will be $M \times M$ symmetric matrices. For example, the coefficient of normal restitution matrix can be written as

$$[e_{\text{n}}] = \begin{bmatrix} e_{\text{n}11} & e_{\text{n}12} & \dots & e_{\text{n}1M} \\ e_{\text{n}21} & e_{\text{n}22} & \dots & e_{\text{n}2M} \\ \vdots & \vdots & \vdots & \vdots \\ e_{\text{n}M1} & e_{\text{n}M2} & \dots & e_{\text{n}MM} \end{bmatrix}, \tag{31}$$

and likewise for the tangential coefficient of restitution. Since the above matrix is symmetric, the user needs to input only $M(M-1)/2$ (top diagonal or lower diagonal) values for normal coefficient of restitution between particle–particle collisions. These $M(M-1)/2$ values can be specified in the "mfix.dat" file by the array name "DES_EN_INPUT". The values are specified for the top diagonal entries of the above matrix from left to right in the following order

$$\{\text{DES\_EN\_INPUT}\} = \{e_{\text{n}11}, e_{\text{n}12}, \dots, e_{\text{n}1M}, e_{\text{n}22}, e_{\text{n}23}, \dots, e_{\text{n}2M}, \dots e_{\text{n}MM}\} \tag{32}$$

### 2.2.3 Hertzian Model

In MFIX-DEM the linear spring dashpot model, discussed above, is the default model used to describe particle-particle and particle-wall collisions. Alternatively, the Hertzian model, could be used for resolving collisions since it has also been implemented in MFIX-DEM. According to Hertzian contact theory, the normal and tangential spring stiffnesses between contacting particles

$i$ and $j$ belonging to the $m^{\text{th}}$ and $\ell^{\text{th}}$ solid–phases can be calculated from the Young's modulus and Poisson ratio as follows

$$k_{n,ij} = \frac{4}{3}\frac{E_m E_l \sqrt{r^*_{ml}}}{E_m(1-\sigma_l^2) + E_l(1-\sigma_m^2)}\delta_{n,ij}^{\frac{1}{2}}, \tag{33}$$

and

$$k_{t,ij} = \frac{16}{3}\frac{G_m G_l \sqrt{r^*_{ml}}}{G_m(2-\sigma_l) + G_l(2-\sigma_m)}\delta_{n,ij}^{\frac{1}{2}}, \tag{34}$$

where $E_m$ and $E_l$ are the Young's moduli and $\sigma_m$ and $\sigma_l$ are the Poisson ratios for $m^{\text{th}}$ and $l^{\text{th}}$ solid–phase, respectively. $G_m$, $G_l$ are the shear moduli calculated as $G_m = \frac{E_m}{2(1+\sigma_m)}$, $G_l = \frac{E_l}{2(1+\sigma_l)}$, and $\frac{1}{r^*_{ml}} = \frac{1}{r^{(m)}} + \frac{1}{r^{(l)}}$. The damping coefficients are related to the spring stiffness and restitution coefficients as given earlier by Eq. 30.

Similar treatment is applied for the wall-particle contact. By default, the linear spring-dashpot model is used in the DEM simulation. To active the Hertzian model, the key word "DES_COLL_MODEL" must be set to "HERTZIAN" and material properties, including Young's modulus, Poisson ratio, and normal and tangential restitution coefficients, must be specified.

### 2.2.4 Estimation of gas–solid momentum transfer term $\mathbf{I}_{gm}$

In this section since we are interested in calculating the momentum interaction term $\mathbf{I}_{gm}$ between the gas–phase and $m^{\text{th}}$ solid–phase, the discussion is limited to particles belonging to $m^{\text{th}}$ solid–phase.

Consider $i^{\text{th}}$ particle, belonging to $m^{\text{th}}$ solid–phase, that resides in $k^{\text{th}}$ computational cell at time $t$. The drag force on this particle is represented as

$$\mathbf{F}_{\text{d}}^{(i\in k,m)} = -\boldsymbol{\nabla} P_{\text{g}}(\mathbf{X}^{(i)})\mathcal{V}_{\text{m}} + \frac{\beta_{\text{m}}^{(i\in k)}\mathcal{V}_{\text{m}}}{\varepsilon_{sm}}\left(\mathbf{v}_{\text{g}}(\mathbf{X}^{(i)}) - \mathbf{V}^{(i)}\right), \tag{35}$$

where $P_{\text{g}}(\mathbf{X}^{(i)})$ and $\mathbf{v}_{\text{g}}(\mathbf{X}^{(i)})$ are the gas–phase mean pressure $P_{\text{g}}$ and velocity $\mathbf{v}_{\text{g}}$ fields at the particle location, $\mathcal{V}_{\text{m}} = \frac{\pi D_{\text{m}}^3}{6}$ is the particle volume, and $\beta_{\text{m}}^{(i\in k)}$ is the local gas–solid momentum transfer coefficient for particle $i$ residing in $k^{\text{th}}$ cell. An explicit functional form of $\beta_{\text{m}}^{(i\in k)}$ is not known theoretically and, therefore, different correlations deduced from experimental and numerical studies are used to model this term. Nevertheless, a general parametrization for $\beta_{\text{m}}^{(i\in k)}$ that subsumes different models can be written as

$$\beta_{\text{m}}^{(i\in k)} = \beta\left(\rho_{\text{m}}, D_{\text{m}}, \left|\mathbf{V}^{(i)} - \mathbf{v}_{\text{g}}(\mathbf{X}^{(i)})\right|, \rho_{\text{g}}, \mu_{\text{g}}\right). \tag{36}$$

The gas–solid momentum transfer term $\mathbf{I}_{gm}$, at $\mathbf{x}_k$, that enters the gas–phase momentum conservation equation (Eq. 2) is computed as

$$\mathbf{I}_{gm}^k = \frac{1}{\mathcal{V}_{\text{k}}}\sum_{i=1}^{N_{\text{m}}}\mathbf{F}_{\text{d}}^{(i\in k,m)}K(\mathbf{X}_{\text{m}}^{(i)}, \mathbf{x}_k), \tag{37}$$

where $K(\mathbf{X}_{\text{m}}^{(i)}, \mathbf{x}_k)$ is a generic kernel with compact support and determines the influence of the particle force at $\mathbf{X}_{\text{m}}^{(i)}$ on a grid node located at $\mathbf{x}^k$, and $\mathcal{V}_{\text{m}}$ is the geometric volume of the $k^{\text{th}}$ grid cell.

In MFIX–DEM, there are two methods available to calculate the above drag force. In the first method, for a particle residing in $k^{\text{th}}$ cell, rather than computing mean gas–phase velocity at the

particle location $\mathbf{v}_{\rm g}(\mathbf{X}^{(i)})$, a cell–centered value of $\mathbf{v}_{\rm g}$ is used. Similarly, rather than using velocity of each particle $\mathbf{V}^{(i)}$, a local cell averaged velocity of the $m^{\rm th}$ solid–phase $\mathbf{v}_{sm}$ is used. With this simplification, the momentum transfer coefficient for all particles of $m^{\rm th}$ solid–phase that reside in cell $k$ is constant and has the following functional form

$$\beta_{\rm m}^{(\forall i \in k)} = \beta_{\rm m}^{(k)} = \left(\rho_{\rm m}, D_{\rm m}, \left|\mathbf{v}_{sm}\left(\mathbf{x}_k\right) - \mathbf{v}_{\rm g}\left(\mathbf{x}_k\right)\right|, \rho_{\rm g}, \mu_{\rm g}\right), \tag{38}$$

where $\mathbf{x}_k$ is the center of the $k^{\rm th}$ cell. Therefore, the drag force on the $i^{\rm th}$ particle belonging to solid–phase $m$ and residing in cell $k$ is

$$\mathbf{F}_{\rm d}^{(i\in k,m)} = -\boldsymbol{\nabla} P_{\rm g}\left(\mathbf{x}_k\right) \mathcal{V}_{\rm m} + \frac{\beta_{\rm m}^{(k)} \mathcal{V}_{\rm m}}{\varepsilon_{sm}} \left(\mathbf{v}_{\rm g}\left(\mathbf{x}_k\right) - \mathbf{v}_{sm}\left(\mathbf{x}_k\right)\right). \tag{39}$$

Under this approximation of constant drag force on all particles residing in a particular cell, the gas–solid momentum transfer term $\mathbf{I}_{gm}^k$ is estimated in $k^{\rm th}$ cell as

$$\mathbf{I}_{gm}^k = -\varepsilon_{sm} \boldsymbol{\nabla} P_{\rm g}\left(\mathbf{x}_k\right) + \beta_{\rm m}^{(k)} \left(\mathbf{v}_{\rm g}\left(\mathbf{x}_k\right) - \mathbf{v}_{sm}\left(\mathbf{x}_k\right)\right). \tag{40}$$

In the second method to calculate gas–solid momentum transfer term, the mean gas–phase velocity is interpolated to the particle location. Using Eq. 37, the drag force on each particle is then projected back onto to the Eulerian gas–phase grid. However, in order to avoid the complexities in numerical algorithm that will arise as a result of forward and backward interpolation of the gas–phase pressure field, the pressure drag force term is evaluated at the cell center (resulting in equal pressure drag force on all particles residing in a particular cell). Therefore, the gas–solid momentum transfer term $\mathbf{I}_{gm}^k$ is estimated in $k^{\rm th}$ cell as

$$\mathbf{I}_{gm}^k = -\varepsilon_{sm} \boldsymbol{\nabla} P_{\rm g}\left(\mathbf{x}_k\right) + \frac{1}{\mathcal{V}_{\rm k}} \sum_{i=1}^{N_{\rm m}} \frac{\beta_{\rm m}^{(i\in k)} \mathcal{V}_{\rm m}}{\varepsilon_{sm}} \left(\mathbf{v}_{\rm g}(\mathbf{X}_{\rm m}^{(i)}) - \mathbf{V}_{\rm m}^{(i)}\right) K(\mathbf{X}_{\rm m}^{(i)}, \mathbf{x}_k). \tag{41}$$

The first method is the default. In order to turn on the second method, set the variable 'DES_INTERP_ON' to '.TRUE.' in the input file.

# 3 Computational Details

In this section, we will give you a short overview of the computational implementation of the above physics algorithms in the MFIX–DEM code. This should serve as a starting point for understanding the numerical methods, code structure, and implementation. A more detailed Doxygen output can be downloaded from the MFIX website for those who wish to dig deeper into the code (https://mfix.netl.doe.gov/documentation/dem_refman.pdf). In addition, we have placed the html version of Doxygen output at https://mfix.netl.doe.gov/members/develop/doxygen_docs/dem/main.html so that one can browse through the code easily.

Below is the algorithm implemented in MFIX and a graphical version of this call sequence is provided in Fig. 4.

We do not delve into the details of the numerical methods employed for the continuum part of the MFIX–DEM as they are widely covered elsewhere (Syamlal, 1998). The important computational aspects of DEM method for coupled gas-granular flows are time-integration, neighbor search algorithms and interpolation of the continuum quantities to the discrete particle locations and vice-versa. We provide more details of these important aspects below:


Figure 4: Call graph of the main DES routines called in a MFIX–DEM execution

### 3.1 Time Integration

Time integration in DEM is one of the most widely researched area in the broad area of molecular dynamics (Khakimov, 2002; Omelyan et al., 2002; Rougier et al., 2004) . A stable, efficient and energy preserving time-integration scheme is desirable. Limited time-integration options are currently available in MFIX–DEM and this is open to further extension in the future. The default time integration scheme is a first-order technique. In the first-order scheme, the translational velocity, particle center position, and the angular velocity at time $t + \Delta t$ are obtained from values at time $t$ by

$$\mathbf{V}^{(i)}(t + \Delta t) = \mathbf{V}^{(i)}(t) + \frac{\mathbf{F}_{\mathrm{T}}^{(i)}(t)}{m^{(i)}}\Delta t, \tag{42}$$

$$\mathbf{X}^{(i)}(t + \Delta t) = \mathbf{X}^{(i)}(t) + \mathbf{V}^{(i)}(t + \Delta t)\Delta t, \tag{43}$$

and

$$\boldsymbol{\omega}^{(i)}(t + \Delta t) = \boldsymbol{\omega}^{(i)}(t) + \frac{\mathbf{T}^{(i)}(t)}{I^{(i)}}\Delta t, \tag{44}$$

respectively, where $\mathbf{F}_{\mathrm{T}}^{(i)}$ and $\mathbf{T}^{(i)}$ are the total force and torque acting on the particle (cf. Eqns. 6 and 7).

**Algorithm 1** MFIX–DEM Algorithm

1: Read initial gas flow field, particle positions and velocities, boundary conditions etc.: This is done through *mfix* calling *get_data* subroutine. *get_data* in turns calls *init_namelist/des_init_namelist*, *des_allocate_arrays*, and *des_init_arrays*. In addition, *des_check_data* is called to perform some sanity checks.
2: Compute values at next time step: mfix calls *time_march* to compute the gas phase quantities at the next time step. *Time_march* in turn calls *des_time_march* to calculate the updated values for the particles through the following iteration
3: *des_time_march* calling sequence: call *particles_in_cells* to determine the particle location on the background Eulerian grid, call *neighbor* to determine the particle neighbor information, call *calc_force_des* to compute the forces (particle collisions, friction, drag etc.) and integrate in *cfnewvalues* based on the information from all these routines to calculate the new particle values as well as the drag information that would be used for next continuum solve. Note that the integration used for the Lagrangian tracking is currently first–order in time and that needs to be fixed. In addition, the splitting between the continuum and discrete solve is first–order in time and one could use something like Strang–splitting for an higher–order implementation.
4: Iterate in des_time_march till $\Sigma\Delta t_{discrete} = \Delta t_{continuum}$
5: Iterate in time_march till $\Sigma\Delta t_{continuum} \geqslant t_{write}, t_{restart} etc.$
6: Iterate in time_march till $\Sigma\Delta t_{continuum} \geqslant t_{stop}$

In addition to the default first-order time integration scheme, a second-order technique, Adams-Bashforth scheme, is also available. This method can be turned on by setting the user input variable DES_INTG_METHOD to "ADAMS_BASHFORTH" in the user-input file (see Table 2 for a complete list of all available DEM user-input variables). In the Adams Bashforth scheme, the translational velocity, particle center position, and the angular velocity at time $t + \Delta t$ are obtained from values at time $t$ and also $t - \Delta t$ by

$$\mathbf{V}^{(i)}(t + \Delta t) = \mathbf{V}^{(i)}(t) + \frac{0.5}{m^{(i)}} \left(3\mathbf{F}_{\mathrm{T}}^{(i)}(t) - \mathbf{F}_{\mathrm{T}}^{(i)}(t - \Delta t)\right) \Delta t, \tag{45}$$

$$\mathbf{X}^{(i)}(t + \Delta t) = \mathbf{X}^{(i)}(t) + 0.5 \left(3\mathbf{V}^{(i)}(t) - \mathbf{V}^{(i)}(t - \Delta t)\right) \Delta t, \tag{46}$$

and

$$\boldsymbol{\omega}^{(i)}(t + \Delta t) = \boldsymbol{\omega}^{(i)}(t) + \frac{0.5}{I^{(i)}} \left(3\mathbf{T}^{(i)}(t) - \mathbf{T}^{(i)}(t - \Delta t)\right) \Delta t, \tag{47}$$

respectively.

Since the Adams-Bashforth scheme requires values from $t - \Delta t$, it is computationally more expensive than the first-order scheme. Since the time-step in MFIX-DEM's soft-sphere approach is based on spring stiffness coefficient, it is reasonably small. Subsequently, the first-order time stepping scheme is a good option, which is fast and has less memory requirement (Dziugys and Peters, 2001). It should be noted that the first-order scheme can suffer from poor energy conservation. However, energy conservation is not as crucial of an issue in gas-solid systems as it is in molecular or pure granular systems due to the presence of non-conservative forces such as inter-phase drag force (van der Hoef et al., 2008).

### 3.2 Neighbor Search Algorithm

One of the most important and time consuming component of any particle–based simulations is the neighbor search algorithm. In MFIX–DEM code, the user has an option to choose between four

neighbor search algorithms by specifying an appropriate value for "DES_NEIGHBOR_SEARCH" variable in the input file. A value of 1 will use the simplest but also the most expensive "$N^2$" search algorithm, where $N$ is the total number of particles in the domain. Therefore, it should be used either for a small system or for debugging purposes. Input values of 2 and 3 correspond to the "Quadtree" and "Octree" search algorithms. All the above three methods are grid–free methods. In MFIX–DEM, the particles are binned (or marked) according to the cell their center belongs to. The fourth method, referred to as the "Cell–linked list" search algorithm, exploits this existing information and is activated by specifying 4 for "DES_NEIGHBOR_SEARCH". As shown by the 2−D schematic in Fig. 5, if the particle of interest is the one represented by the filled circle, then the particles belonging to 9 (27 for the 3-D case) adjacent cells, along with particles belonging to the same cell as the particle of interest, are considered as potential neighbors. Thus, only these particles are further checked against the particle of interest for a neighbor contact. For any of these search methods, any two particles $i$ and $j$ that are located at $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$, and have radii $R_i$ and $R_j$, are considered neighbors if they satisfy the following condition

$$\left|\mathbf{X}^{(i)} - \mathbf{X}^{(j)}\right| < K(R_i + R_j), \tag{48}$$

where $K$ is a user input variable by the name "FACTOR_RLM" and its default value is equal to 1.2. If "FACTOR_RLM" is specified as one, then only the particles that are either nearly touching or overlapping will be considered as neighbors. For this setting the neighbor search algorithm would have to be called each time step to ensure that the simulation does not miss any possible collision, which would result in high computational expense. Alternatively, a very high value for "FACTOR_RLM" is also not advisable as a particle might end up with more neighbors than the array sizes can accommodate, resulting in run–time segmentation errors.

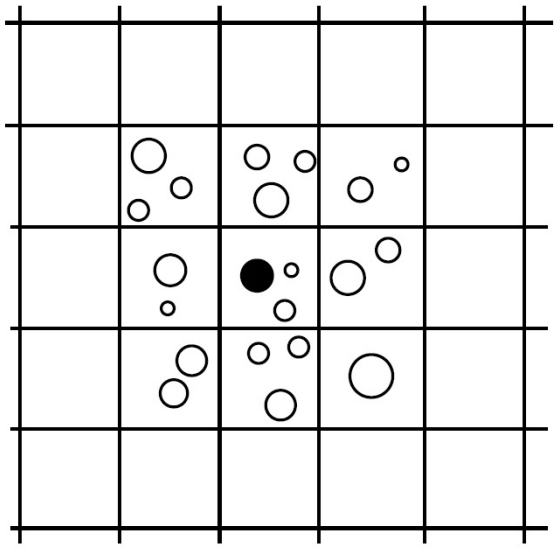


Figure 5: 2-D Schematic for "cell–linked list" neighbor search algorithm. Hollow and filled circles represent particles of different radii.

Another important parameter is the frequency at which the neighbor search algorithm is called. In the MFIX–DEM implementation, the neighbor–search algorithm is called every time the code enters the DES modules from the Eulerian solver. Once in the DES modules, the neighbor search

algorithm is called after every "NEIGHBOR_SEARCH_N" number of DES iterations. The default value for "NEIGHBOR_SEARCH_N" is equal to 25. Between "NEIGHBOR_SEARCH_N" DES iterations, if any particle moves by more than "NEIGHBOR_SEARCH_RAD_RATIO" (user input, default value = 1.0) times its radius, then the neighbor search algorithm is called. Since the system dynamics for every problem are not known a priori, this second test (based on "NEIGHBOR_SEARCH_RAD_RATIO") is critical and ensures against simulations becoming unstable due to large particle overlaps which might occur if a high value for "NEIGHBOR_SEARCH_N" is specified. Out of the four options available for neighbor search, we recommend using the cell-linked list search algorithm for production runs (DES_NEIGHBOR_SEARCH=4). Quadtree and Octree have not been extensively tested with the recently debugged MFIX-DEM code, so the user is very strongly encouraged to run sanity checks before using Quadtree or Octree. The computationally most expensive "$N^2$" search algorithm should be used sparingly, preferrably for debugging purposes.

### 3.3 Gas-Solids Coupling

Since the velocity of solid-phase is evolved by explicit time integration in MFIX-DEM, the gas-solids coupling is a little different from the one descibed in MFIX numerics manual (Syamlal, 1998). The numerical details of the gas-solids coupling in MFIX-DEM are discussed in Appendix A.

## 4 DEM Verification Tests

We wish to perform a series of verification studies for pure granular flows as well as gas–particle flows. The MFIX–DEM code is extremely complex with the interaction between the fluid–solver, particle–solver, collision–algorithms, boundaries etc. In addition, the fluid–solver is on a staggered–grid with scalar quantities solved on the cell centers while the velocities are computed on the cell faces. With all the above complexities, limited verification may be performed by visually comparing the code segments to the equations being solved. In addition, a series of verification tests were performed to probe for the accuracy of each of the units of this complex model. Additional tests may be added in the future as they become available or as new features are incorporated.

### 4.1 Freely Falling Particle

**Directory: mfix/tests/dem-tests/freely-falling-particle**

In this case, a single smooth (frictionless) spherical particle freely falling under gravity from its initial position bounces upon collision with a fixed wall. For a schematic of the problem see Figure 5. The translational motion of the particle can be described in three stages: free fall, contact and rebound. Following the work (Chen et al., 2007), an analytic expression for particle motion during each stage is obtained.

#### 4.1.1 Stage I: Free fall

The expression describing the particle motion during free fall is obtained from the force balance on the particle

$$\ddot{y} = -g \tag{49}$$

$$\dot{y} = \int \ddot{y} dt = -gt \tag{50}$$

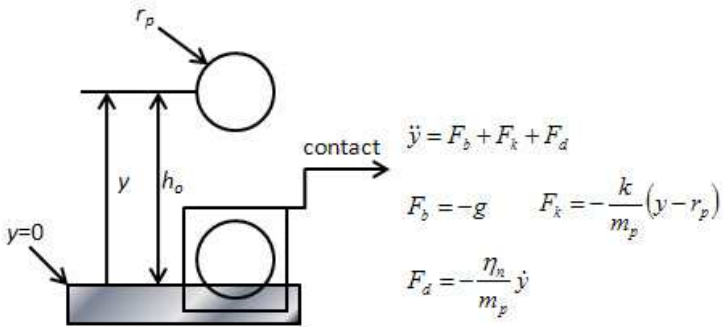



Figure 6: Schematic for the free fall verification case: a smooth spherical particle falling onto a fixed wall. The forces acting on the particle during contact are also presented.

$$y = \int \dot{y} dt = h_{\mathrm{o}} - \frac{1}{2} g t^2 \tag{51}$$

with the initial conditions $\dot{y}(t = 0) = 0$ and $y(t = 0) = h_{\mathrm{o}}$ and where $g$ is the acceleration due to gravity, $h_{\mathrm{o}}$ is the initial distance of the particle center from the wall, $y$ is the particle's center position with respect to the wall, $\dot{y}$ is the velocity, and $\ddot{y}$ is the acceleration.

#### 4.1.2 Stage II: Contact

The time at which the particle contacts the fixed wall ($t_{\mathrm{c}}$) signifies the end of the free fall stage and the beginning of the contact stage. This time corresponds to the particle center position equal to the particle radius (i.e., $y(t = t_{\mathrm{c}}) = r_{\mathrm{p}}$) and its value can be found via Eq. 51:

$$t_{\mathrm{c}} = \sqrt{2 \left( h_{\mathrm{o}} - r_{\mathrm{p}} \right) / g}. \tag{52}$$

Using the expression for $t_{\mathrm{c}}$ and equation 50, the velocity just prior to contact can be described as

$$v_{\mathrm{c}} = -\sqrt{2 g h_{\mathrm{o}}} = -\sqrt{2 g \left( y_o - r_{\mathrm{p}} \right)} \tag{53}$$

In this case, the particle-wall collision is treated using a soft-sphere approach, specifically the linear spring-dashpot model discussed earlier. Accordingly, expression for particle acceleration during contact is given by

$$\ddot{y} = -g - \frac{k_{\mathrm{n}}}{m_{\mathrm{p}}} \left( y - r_{\mathrm{p}} \right) - \frac{\eta_{\mathrm{n}}}{m_{\mathrm{p}}} \dot{y} \tag{54}$$

For convenience the terms $\beta = \eta_{\mathrm{n}} / 2 \sqrt{k_{\mathrm{n}} m_{\mathrm{p}}}$ and $\omega_{\mathrm{o}} = \sqrt{k_n / m_{\mathrm{p}}}$ are introduced and equation 54 can be rewritten and rearranged as

$$\ddot{y} + 2 \beta \omega_{\mathrm{o}} \dot{y} + \omega_{\mathrm{o}}^2 y = \omega_{\mathrm{o}}^2 r_{\mathrm{p}} - g \tag{55}$$

The solution to this equation depends on the value of $\beta$. For $\beta < 1$ (under damped system) the expression describing particle motion during contact is

$$y = \left[ \begin{array}{l} \frac{g}{\omega_{\mathrm{o}}^2} \cos\left( \sqrt{1 - \beta^2} \omega_{\mathrm{o}} t \right) + \\ \frac{-\sqrt{2g(h_{\mathrm{o}} - r_{\mathrm{p}})} + \frac{\beta g}{\omega_{\mathrm{o}}}}{\omega_{\mathrm{o}} \sqrt{1 - \beta^2}} \sin\left( \sqrt{1 - \beta^2} \omega_{\mathrm{o}} t \right) \end{array} \right] \exp\left( -\beta \omega_{\mathrm{o}} t \right) + \left( r_{\mathrm{p}} - \frac{g}{\omega_{\mathrm{o}}^2} \right) \tag{56}$$

with the initial conditions $\dot{y}(t=0) = v_c$ and $y(t=0) = r_\mathrm{p}$. The particle velocity is

$$\dot{y} = \left[ \begin{array}{l} -\sqrt{2g\left(h_\mathrm{o} - r_\mathrm{p}\right)}\cos\left(\sqrt{1-\beta^2}\omega_\mathrm{o} t\right) + \\ \frac{\beta\omega_\mathrm{o}\sqrt{2g(h_\mathrm{o}-r_\mathrm{p})}-g}{\omega_\mathrm{o}\sqrt{1-\beta^2}}\sin\left(\sqrt{1-\beta^2}\omega_\mathrm{o} t\right) \end{array} \right] \exp\left(-\beta\omega_\mathrm{o} t\right). \tag{57}$$

#### 4.1.3 Stage III: Rebound

The time at which the particle is no longer in contact with the fixed wall ($t_\mathrm{r}$) signifies the end of the contact stage and the beginning of the rebound stage. This time corresponds to the particle center position equal to the particle radius (i.e., $y(t = t_\mathrm{r}) = r_\mathrm{p}$) and its value can be found via equation 56. Using the value of $t_\mathrm{r}$ and equation 57 the velocity at the end of the contact stage ($v_\mathrm{r}$) can be found. For particle motion during rebound the same starting equation is used as for free fall, but the initial conditions differ:

$$\dot{y} = \int \ddot{y} dt = -gt + v_\mathrm{r} \tag{58}$$

$$y = \int \dot{y} dt = r_\mathrm{p} + v_\mathrm{r} t - \frac{1}{2} g t^2 \tag{59}$$

with initial conditions $\dot{y}\left(t=0\right) = v_\mathrm{r}$ and $y(t=0) = r_\mathrm{p}$.

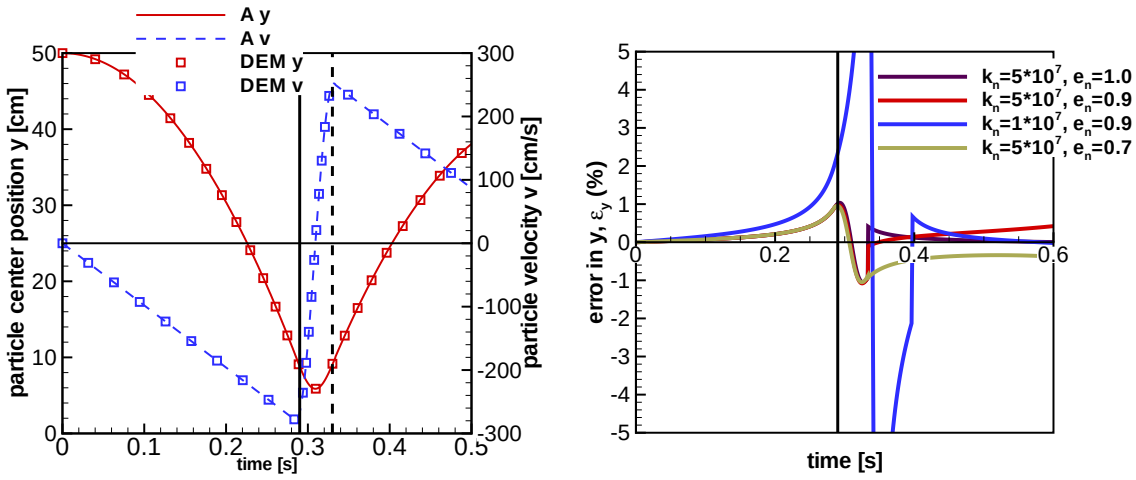
#### 4.1.4 Results

Equations 51, 56 and 59 are used to solve for particle position versus time and the values of $t_\mathrm{c}$ and $t_\mathrm{r}$ are used to stitch the three stages (free fall, contact, rebound) together. The analytic solution for particle position and velocity, labeled with (A), is compared to the results obtained from DEM simulation, labeled with (DEM), in Figure 7(a) for a system with a particle-wall spring coefficient $k_\mathrm{n} = 5 \times 10^7$ dyne/cm and a particle-wall restitution coefficient $e_\mathrm{n} = 0.9$. Note that $\eta_\mathrm{n}$ can be found using equation 30 and knowing $k_\mathrm{n}$, $e_\mathrm{n}$, and the effective mass $m_\mathrm{eff}$, which for particle-wall contact is simply taken as the particle mass $m_\mathrm{p}$. In all the verification studies of the free fall system the following values are used: $r_\mathrm{p} = 10$ cm, particle material density $\rho_\mathrm{p} = 2.6$ g/cm$^3$, $h_\mathrm{o} = 50$ cm and $g = 980.0$ cm/s$^2$. Differences between the DEM results and analytic solution are difficult to discern in Figure 7(a). Accordingly, the relative percent error in the prediction of particle position ($\epsilon_y$) is presented in Figure 7(b) for two values of the spring coefficient ($5 \times 10^7$ and $1 \times 10^7$ dyne/cm) and three different values of the coefficient of restitution (1.0, 0.9, 0.7). For any quantity $Q$, the relative percent error $\epsilon_Q$ between the values predicted by DEM simulation (denoted by $\{Q\}$) and analytically expected values (denoted by $Q_A$) can be defined as

$$\epsilon_Q = 100 \times \left| \frac{Q_A - \{Q\}}{Q_A} \right|. \tag{60}$$

For the four cases shown, the magnitude of the percent error is generally less than 1% during all three stages. The case characterized by $k_\mathrm{n} = 1 \times 10^7$ dyne/cm and $e_\mathrm{n} = 0.9$ is the exception. In this particular case, the particle center nearly touches the fixed boundary ($y \to 0$); as a result, the percent error in particle center position is relatively large during the contact stage as the absolute values of $y$ approach zero. This case also has both the smallest spring constant (softest particle) and the largest time step for the DEM simulations.

A few reasons for some of the discrepancies observed between the DEM and analytic solution are discussed. In these DEM simulations, position is updated using a first order scheme (i.e., as a




(a) (b)

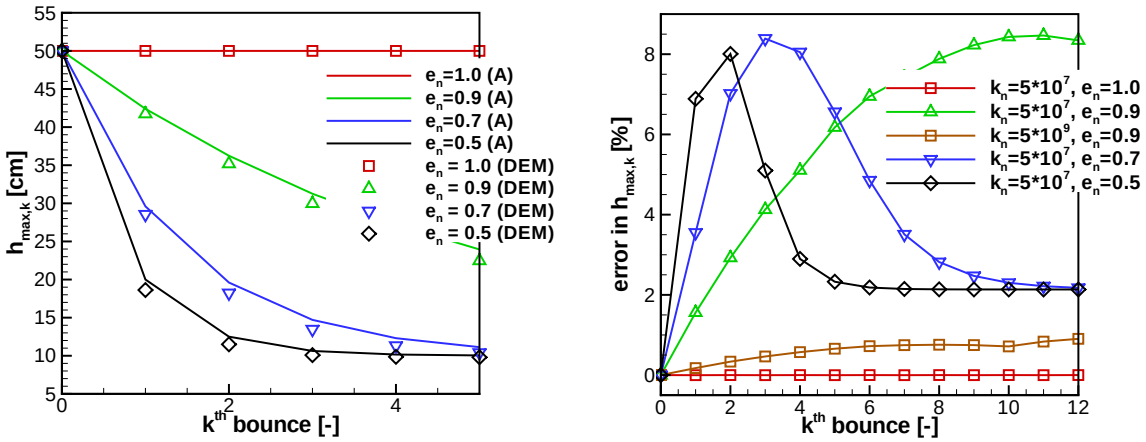
Figure 7: (a) Comparison between analytic solution and DEM results for the single particle free fall case. (a) particle position and velocity for system with $k_{\mathrm{n}} = 5 \times 10^7$ dyne/cm and $e_{\mathrm{n}} = 0.9$. (b) relative percent error between analytic and DEM results for four different systems. In (a) and (b) the solid vertical line, labeled $t_{\mathrm{c}}$, refers to the time of collision, while in (a) the dotted vertical line, labeled $t_{\mathrm{r}}$, refers to the time of rebound.

result, the error in position grows during each stage with each successive time step. Other, higher order schemes could be used to update position with potentially more accurate results. Besides errors from the specific time-stepping method, errors are also introduced at the start and end of the collision. In the DEM simulation the particle position will be advanced such that its edge will overlap with the wall before the contact (collision) is detected, that is, the particle motion is still considered as freely falling even though it is in contact with the wall. In addition, the particle is advanced a finite distance beyond the wall while still being considered in contact with the wall. Either of these errors may be mitigated by using smaller time steps, which in the current DEM code is achieved by using larger spring constants or restitution coefficients closer to 1.

To this point the discussion has focused on comparing the simulation results to the analytical solution from a soft-sphere collision model. Since MFIX also employs the soft-sphere model to resolve collisions, this comparison serves to verify the implementation of the model in the code and reflects the accuracy of the integration method. As evident by Fig. 7(b), the first-order time-stepping method appears to be sufficient for this case (errors less than 1%). In addition to the soft-sphere comparison performed above, the simulation results may also be compared to the analytic solution from a hard-sphere collision model. The hard-sphere model does not involve a contact stage (collisions are assumed instantaneous). If the particle is dropped from an initial height $h_{\mathrm{o}}$, (recall $h_{\mathrm{o}}$ = particle center position) then the maximum height it reaches after its first collision with the wall is $e_{\mathrm{n}}^2\,(h_{\mathrm{o}} - r_{\mathrm{p}})$. A general expression for the maximum height of the particle center attained after $k$ collisions is

$$h_{\mathrm{max},k} = (h_{\mathrm{o}} - r_{\mathrm{p}})\, e_{\mathrm{n}}^{2k} + r_{\mathrm{p}}. \tag{61}$$

Figure 8(a) shows the evolution of $h_{\mathrm{max},k}$ obtained from DEM simulation (denoted DEM) compared with the above analytical expression (denoted A) for different values of $e_{\mathrm{n}}$. The problem setup is the same was used in Fig. 7 with $r_{\mathrm{p}} = 10$ cm, $\rho_{\mathrm{p}} = 2.6$ g/cm$^3$, $h_{\mathrm{o}} = 50$ cm and $g = 980.0$ cm/s$^2$. The relative percent error in $h_{\mathrm{max},k}$ ($\epsilon_{h_{\mathrm{max},k}}$) is shown in Fig. 8(b). This "error" is really a reflection




(a) (b)

Figure 8: Comparison between the analytic solution from a hard-sphere model and the DEM results for a freely falling particle under gravity. Evolution of (a) $h_{\max,k}$ (the maximum height attained after $k$ collisions with a wall) and (b) relative percent error (Eq. 60) between analytic and DEM results for different values of normal coefficient of restitution $e_\mathrm{n}$ (in (a), a constant value of $k_\mathrm{n} = 5 \times 10^7$ is used for all cases) and normal spring stiffness coefficient $k_\mathrm{n}$.

of the difference between the hard-sphere and soft-sphere collision models (note errors associated with time-stepping were already demonstrated in Fig. 7(b) to be minimal in this case). In the limit of the hard-sphere model (increasing the spring constant) the difference between the two models will decrease as is demonstrated in Fig. 8(b) for $e_\mathrm{n} = 0.9$. However, to accurately capture collisions in such a limit requires increasingly small time steps, and in turn, increased computational time. The error is minimal for the purely elastic case. In contrast, for inelastic collisions the error may exhibit a local maximum before the particle comes to a rest at which point the error remains constant.

## 4.2 Two Stacked Particles Compressed between Two Boundaries

**Directory: mfix/tests/dem-tests/stacked-particles**

This case study is based on the work of (Chen et al., 2007) and consists of a system of two stacked particles placed between two fixed walls, as shown in Figure 8, so that they are compressed. The particles are of equal radius but may have differing densities. The lower wall is placed at $y = 0$, the upper wall at $y = 3.6r_\mathrm{p} = y_\mathrm{w}$, and the initial position of the two particles is $y_{1\mathrm{o}} = y(t = 0) = 0.25y_\mathrm{w}$ and $y_{2\mathrm{o}} = y(t = 0) = 0.75y_\mathrm{w}$. In this setup the particles and the walls will remain in contact at all times so that the contact spring force will always be in compression. The differential equation of motion for this system is examined and a numerical solution obtained which is then compared with the results from the DEM simulation.

### 4.2.1 Motion of Particle 1: Lower Particle

A general expression for the acceleration of particle 1 ($\ddot{y}_1$) is as follows

$$\ddot{y}_1 = F_{1\mathrm{b}} + F_{1\mathrm{kw}} + F_{12\mathrm{k}} + F_{1\mathrm{dw}} + F_{12\mathrm{d}} \tag{62}$$

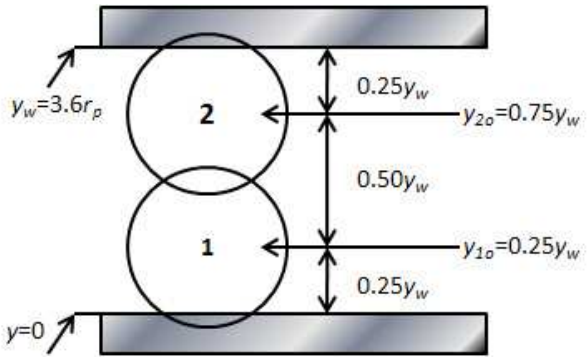



Figure 9: Schematic for the two stacked particle system verification case: two smooth spherical particles stacked between two fixed walls so that the system is always under compression. The various forces acting on particle 1 and on particle 2 are also indicated.

where the terms on the right-hand-side represent the various forces acting on particle 1, specifically, $F_{1b}$ = gravity force, $F_{1kw}$ = particle 1-wall spring force, $F_{1dw}$ = particle 1-wall damping force, $F_{12k}$= particle 1-particle 2 spring force, $F_{12d}$= particle 1-particle 2 damping force. The expressions for each of these forces are shown below:

$$F_{1b} = -g, \quad F_{1kw} = -\frac{k_{nw}}{m_1}\left(y_1 - r_p\right), \quad F_{1dw} = -\frac{\eta_{n1w}}{m_1}\dot{y}_1,$$
$$F_{12k} = -\frac{k_{nw}}{m_1}\left(2r_p - (y_2 - y_1)\right) \quad \text{and} \quad F_{12d} = -\frac{\eta_{n12}}{m_1}\left(\dot{y}_1 - \dot{y}_2\right), \tag{63}$$

where $g$ is the acceleration due to gravity, $k_{nw}$ is the particle-wall spring coefficient, $F_{12k}$ is the particle-particle spring coefficient, $\eta_{n1w}$ is the particle-wall damping coefficient for particle 1, $\eta_{n12}$ is the particle-particle damping coefficient between particles 1 and 2, $m_1$ is the mass of particle 1, $r_p$ is the particle radius, $y_1$ is the $y$ position of the center of particle 1 with respect to the lower wall, $\dot{y}_1$ is the velocity of particle 1 and similarly, $y_2$ is the $y$ position of the center of particle 2 and $\dot{y}_2$ is the velocity of particle 2.

### 4.2.2 Motion of Particle 2: Upper Particle

A general expression for the acceleration of particle 2 ($\ddot{y}_2$) is as follows

$$\ddot{y}_2 = F_{2b} + F_{2kw} + F_{21k} + F_{2dw} + F_{21d}, \tag{64}$$

where the terms on the right-hand-side are the various forces acting on particle 2, specifically, $F_{2b}$ = gravity force, $F_{2kw}$ = particle 2-wall spring force, $F_{2dw}$= particle 2-wall damping force, $F_{21k}$ = particle 1-particle 2 spring force, $F_{21d}$ = particle 1-particle 2 damping force. The expressions for each of these forces are shown below:

$$F_{2b} = -g, \quad F_{2kw} = -\frac{k_{nw}}{m_2}\left(r_p - (y_w - y_2)\right), \quad F_{2dw} = -\frac{\eta_{n2w}}{m_2}\dot{y}_2,$$
$$F_{21k} = -\frac{m_1}{m_2}F_{12k} \quad \text{and} \quad F_{21d} = -\frac{m_1}{m_2}F_{12d}, \tag{65}$$

where $\eta_{n2w}$ is the particle-wall damping coefficient for particle 2, $m_2$ is the mass of particle 2, and the other quantities are as defined earlier.

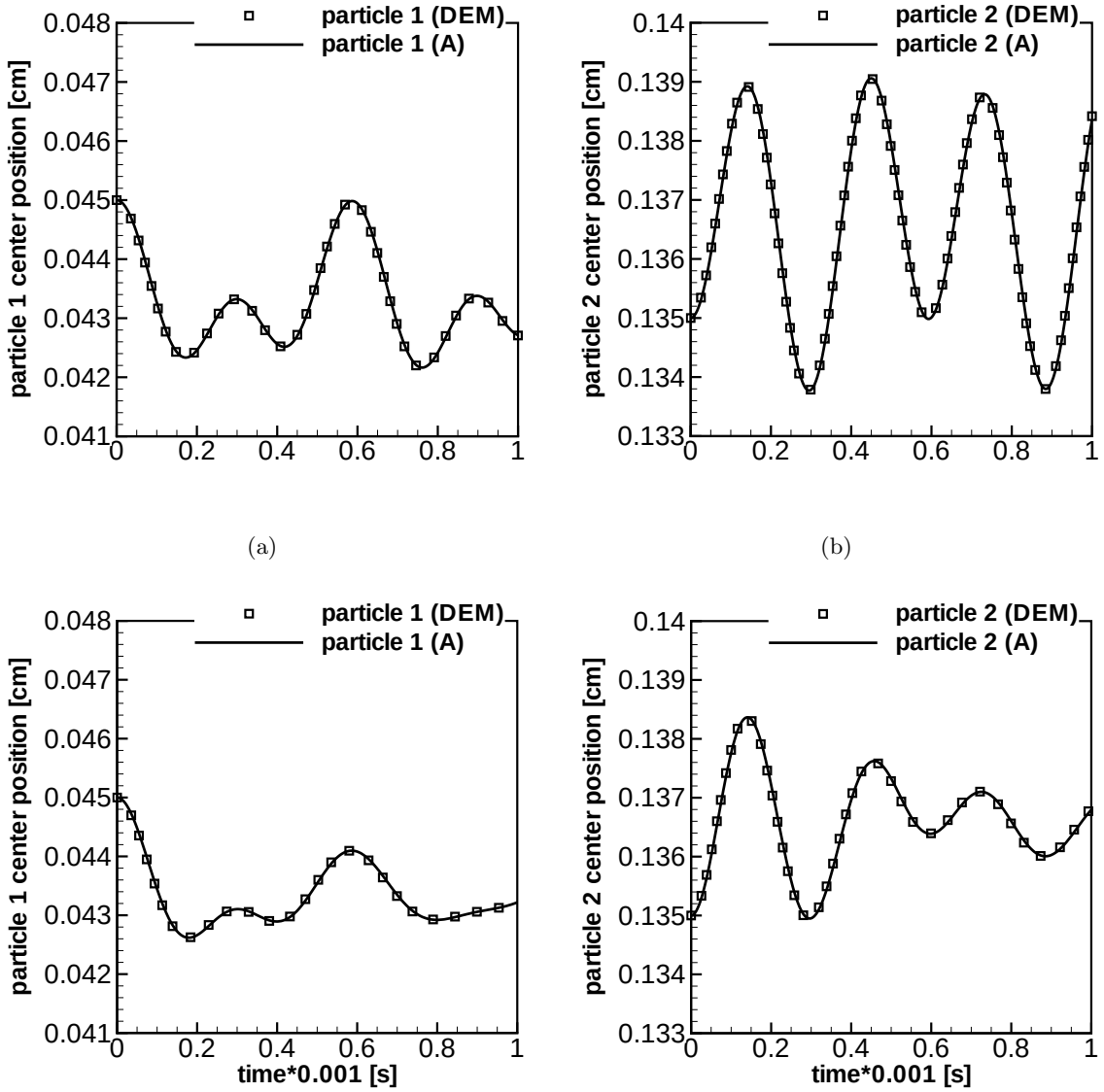
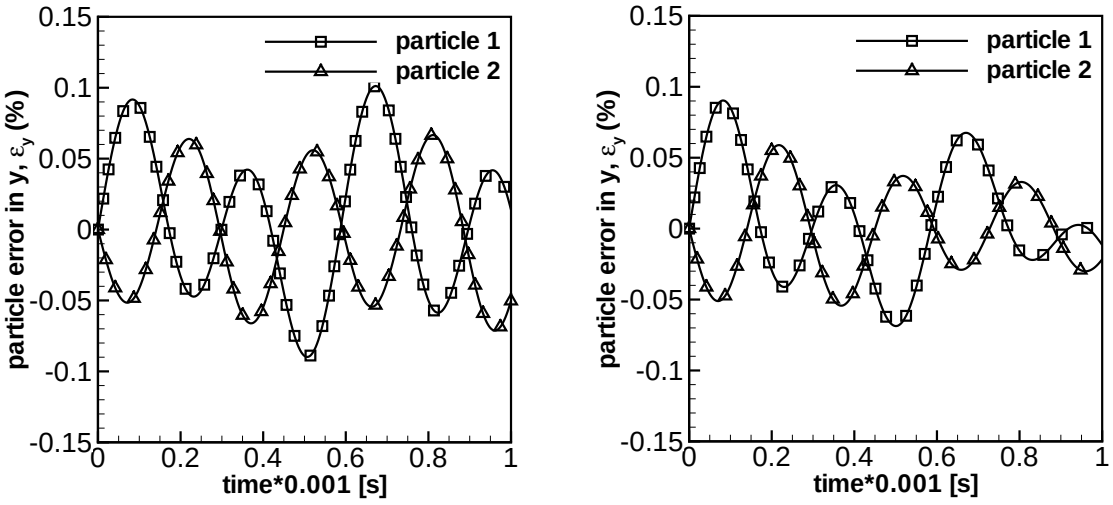



Figure 10: Comparison between analytic solution and DEM results for two stacked particle system. Panels (a) & (b) correspond to system with $e_{\mathrm{n}} = 1.0$. Panels (c) & (d) correspond to system $e_{\mathrm{n}} = 0.8$. The $y-$ position for the center of particle 1 is given in (a) & (c) while the $y-$ position for the center of particle 2 is given in (b) and (d).




(a) (b)

Figure 11: Relative percent error in particle position ($\epsilon_y$) between the numeric solution and DEM results corresponding to the systems presented in Figure 10. Panels (a) and (b) correspond to systems with $e_{\rm n} = 1.0$ and $e_{\rm n} = 0.8$, respectively.

### 4.2.3 Results

An analytic expression for the motion of each particle can readily be obtained in the case of perfectly elastic ($F_{\rm 1dw} = F_{\rm 2dw} = F_{\rm 12d} = 0$) equal mass ($m_1 = m_2$) particles. For particles of unequal mass ($m_1 \neq m_2$) and inelasticity, the problem becomes more complicated. Accordingly, a numerical solution is found using numerical methods, specifically, using the Lsode function with the default options as implemented in GNU Octave Hindmarsh (1983). This function is designed to solve a set of differential equations with the form $\frac{dy}{dt} = f(y,t)$ with $y(t_o) = y_o$. Therefore, the two second order differential equations describing the system (62 and 64) are re-written as a set four first order differential equations. The four coupled first order differential equations are then solved using Lsode with the initial conditions $\dot{y}_1(t=0) = 0$, $\dot{y}_2(t=0) = 0$, $y_1(t=0) = 0.25y_{\rm w}$ and $y_2(t=0) = 0.75y_{\rm w}$.

In all the verification studies of the two stacked particle system the following values are used: $r_{\rm p} = 0.05$ cm, $\rho_{\rm p1} = 20$ g/cm$^3$, $\rho_{\rm p2} = 10$ g/cm$^3$, $k_{\rm n} = k_{\rm nw} = 1 \times 10^6$ dyne/cm, and $g = 980.665$ cm/s$^2$. Thus, the lower particle is twice as dense as the upper particle ($m_1 = 2m_2$). Two coefficients of restitution are tested, a perfectly elastic case ($e_{\rm n} = 1$) and a slightly inelastic case ($e_{\rm n} = 0.8$). (Recall that the damping coefficients are determined using Eq. 30 and knowing the spring constants, the restitution coefficients, and the effective masses).

The numerical solution for each particles position, labeled (A), is compared to the results obtained from DEM simulation, labeled (DEM), in Figures 10(a)- 10(d) for both restitution coefficients examined. Differences between the DEM results and numerical solution are difficult to discern. Accordingly, the relative percent error in the prediction particle position ($\epsilon_y$) is presented in Figure 11. For the two cases shown, the magnitude of the percent error is generally less than 0.2% indicating that the DEM results agree well with the numerical solution.

### 4.3 Ball Slipping on a Rough Surface

**Directory: mfix/tests/dem-tests/rolling-ball**

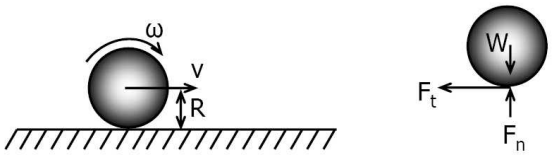



Figure 12: Schematic of the second verification problem. A spherical ball with finite translational velocity and zero angular velocity is placed on a rough surface. Forces acting on the ball is shown by the free body diagram on the right.

In this second verification problem, a spherical ball with finite translational velocity but zero angular velocity is left on a rough surface, also shown by the schematic in Fig. 12. As a result of finite slip at the point of contact between the ball and the rough surface, rolling friction will act in the direction shown in Fig. 12. This rolling friction will reduce the translational velocity and, at the same time, generate an angular velocity until there is zero slip at the point of contact, i.e. $v = \omega R$. After the zero slip condition is reached, rolling friction will cease to act and the solid ball keep on moving with fixed translational and angular velocities.

From the force balance shown in the free body diagram, the normal contact force $F_{\rm n} = W = mg$, where $W$ and $m$ are, respectively, the weight and mass of the spherical ball, and $g$ is the acceleration due to gravity. The tangential contact force $F_t$, which is the force due to rolling friction, is equal to $\mu mg$. Therefore, the evolution equations for translational and angular velocities become

$$\frac{dv_{\rm x}}{dt} = -\mu g, \tag{66}$$

and

$$\frac{d\omega}{dt} = \frac{\mu mgR}{I}, \tag{67}$$

where $I = 2/5mR^2$ is the moment of inertia of the spherical ball. The above equations can be integrated with the initial conditions $\{v_{\rm x}, \omega\}_{t=0} = \{v_0, 0\}$, where $v_0$ is the initial translational velocity of the ball. Since the evolution equations for $v_{\rm x}$ and $\omega$ are known, the time $t_{\rm s}$ at which slipping ends (i.e. $v_{\rm x} = \omega R$), or rolling friction ceases to act, can be calculated analytically. This time $t_{\rm s}$ is

$$t_{\rm s} = \frac{2v_0}{7\mu g}. \tag{68}$$

The non–dimensional translational and angular velocities at $t_{\rm s}$ are

$$\left\{v'_{\rm x}, \omega'\right\}_{t=t_{\rm s}} = \left\{\frac{v_{\rm x}}{v_0}, \frac{\omega R}{v_0}\right\}_{t=t_{\rm s}} = \left\{\frac{5}{7}, \frac{5}{7}\right\} \tag{69}$$

Fig. 13 shows the comparison of $t' = \mu g t_{\rm s}/v_0$ (left axis), and $\{v'_{\rm x}, \omega'\}_{t=t_{\rm s}}$ (right axis) obtained from DEM simulation with the analytic values for different values of coefficient of friction. The relative error, not shown, is always less than 0.1%.

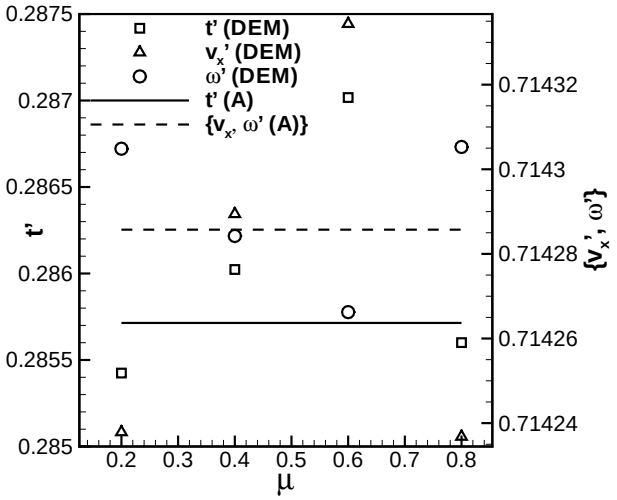



Figure 13: Comparison of $t' = \frac{\mu g t_s}{v_0}$ (left axis), and $\{v_x', \omega'\}_{t=t_s}$ (right axis) obtained from DEM simulation with the expected values for different values of coefficient of friction.

The above three test cases have been limited to pure DEM simulations with the objective of verifying each component of the spring-dashpot model independently. In the next few test cases the objective is now shifted to CDM simulations. Therefore, in regard to CDM simulations, we verify and validate the gas-solids coupling through simple verfication and qualitative analyses.

### 4.4 Particle Terminal Velocity

**Directory: mfix/tests/dem-tests/terminal-velocity**

In this verification test case, the implementation of gas-solids coupling (through interphase drag force) is examined. For a very small spherical particle falling under gravity in gas-phase which is flowing in vertically upwards direction, the velocity of particle evolves by

$$\frac{d\mathbf{v}_p}{dt} = \frac{\mathbf{g}(\rho_p - \rho_g)}{\rho_p} - \frac{3}{4}\frac{\rho_g \left|\mathbf{v}_p - \mathbf{v}_g\right|^2}{d_p \rho_p} C_d, \tag{70}$$

where $d_p$ is the particle diameter, $\mathbf{g}$ the gravitational acceleration, $\rho_p$ and $\rho_g$ are densities of particle and gas, respectively, and $C_d$ is the drag coefficient. The drag coefficient $C_d$ is estimated from the Schiller and Naumann (1933) drag correlation for single particle in an unbounded medium, which is

$$C_d = \frac{24}{\mathrm{Re}}(1 + 0.15\mathrm{Re}^{0.687}) \tag{71}$$

where Re is the Reynolds number based on slip velocity between particle and gas-phase, and is defined as $\mathrm{Re} = \dfrac{\rho_g \left|\mathbf{v}_p - \mathbf{v}_g\right| d_p}{\mu_g}$. It is worth noting that the pressure form drag force has been neglected in the above particle velocity evolution equation due to the assumption of very small particle (particle diameter equal to 100 $\mu$m in this test case).

The gas-phase is assumed to be flowing upwards at 0.4 m/s and the particle's intital velocity is equal to zero. Given the initial conditions and properties of the particle and gas flow, Eq. 70 can be solved numerically (using the above from for $C_d$) to obtain particle velocity at any time. When

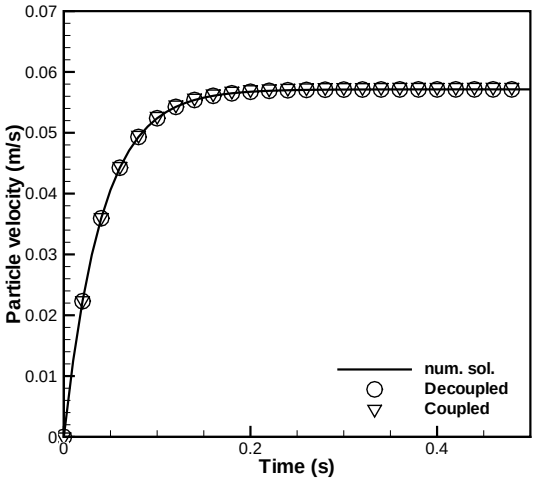



Figure 14: Comparison of the particle velocity evolution obtained from MFIX-DEM with the nmerical solution of Eq. 70. (gas: $\rho_g$=1.2 kg/m$^3$, $\mu_g$=1.8×10$^{-5}$ Pa.s, and $u_g$=0.4 m/s; particle: $\rho_p$=2000 kg/m$^3$, $d_p$=100 $\mu$m).

the weight of particle is exactly balanced by the upward buoyancy and drag forces, the terminal velocity is reached which is obtained from the numerical solution of Eq. 70.

Two different cases are considered in this test case. In the first case (referred to as decoupled case), the gas-phase velocity is fixed and does not evolve. In the second case (referred to as coupled case), the gas-phase velocity also evolves and is affected (through interphase drag force) by the presence of freely falling particle. The evolution of particle velocity obtained from MFIX-CDM for the two cases (decoupled and coupled) is compared with numerical solution of Eq. 70 in Fig.14. It can be seen from the figure that the particle velocity obtained from MFIX-CDM compares excellently with the numerical solution for both cases considered. This test verifies the gas-solids coupling. It should be mentioned that this a very simple, and also limited, test to verify the gas-solids coupling. This is because the very week interphase drag force does not cause any significant change to the gas-phase velocity field.

### 4.5 Advection of a circle and sphere in an oscillating vortex field

**Directories: mfix/tests/dem-tests/circle-advection and mfix/tests/dem-tests/sphere-advection**

In this case we subject the particles arranged in a circle or sphere to an off-centered oscillating vortex field. The particles get distorted from the initial arrangement but return to the original configuration after one cycle and this is good procedure to ascertain any errors introduced for drag calculations. In particular this case tests the gas velocity interpolation routines in arbitrary directions. This test case is typically used in testing advection algorithms (Rider and Kothe, 1998; Liovic et al., 2006; Leveque, 1996).

Deformable vortex in 2D: We use the single vortex velocity field typically used in testing algorithms for interface tracking (Rider and Kothe, 1998) with temporal deformation (Leveque, 1996). The particles are seeded in a circle of radius 0.15 off-centered at (0.5, 0.75) in a unit square box

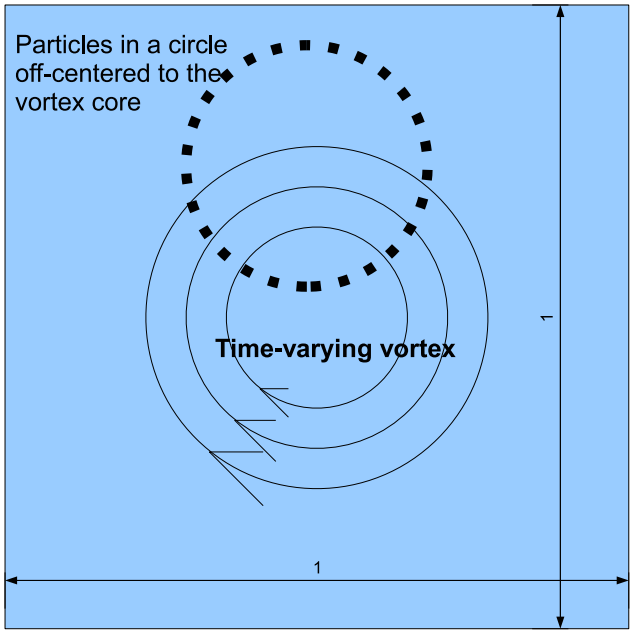



Figure 15: Schematic of the advection of the particles on a circle in a oscillating vortex field

(see the schematic in Fig. 15) with the following gas-phase velocity field

$$u = 2\sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \quad (72)$$
$$v = -\sin(2\pi x)\sin^2(\pi y)\cos(\pi t/T). \quad (73)$$

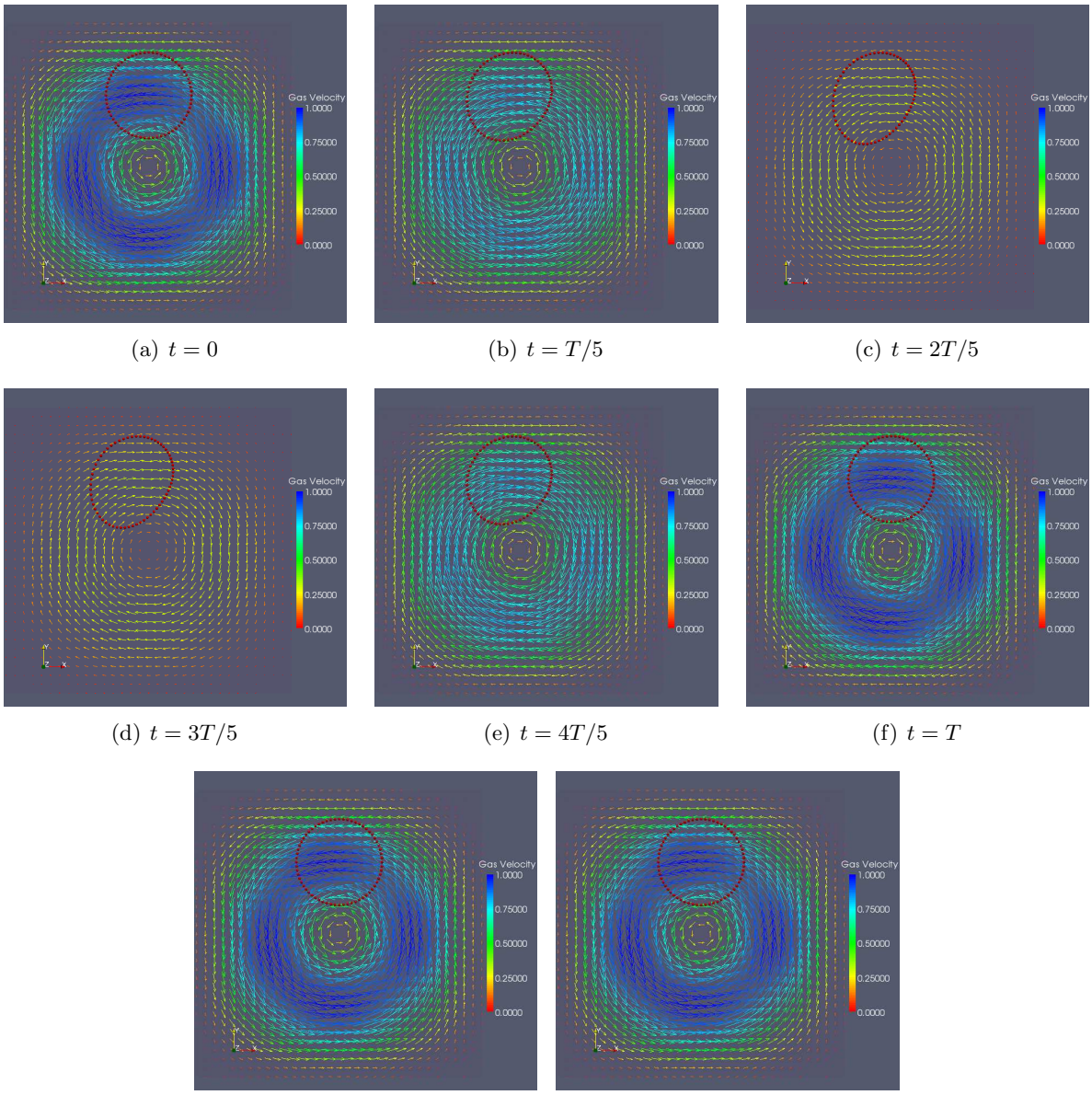
The particles in the circle in the vortex are sheared in arbitrary directions (because the center of the particles is off-centered to that of the vortex) and the degree of deformation will depend on the value of $T$. In our case, we have chosen $T = 0.25$, to test the small deformations over several periods to track the error. Fig. 16 shows the particles along with gas velocity vectors at $t = 0, T/5, 2T/5, 3T/5, 4T/5, T, 2T$, and $16T$. As can be seen in this case, the particles are deformed at $t = T/5, 2T/5, 3T/5$, and $4T/5$ and restored to original location at $t = T, 2T$, and $16T$ as expected. We can repeat the case for 3D where the particles are on the sphere of radius 0.5 and centered at (0.35, 0.35, 0.35) using the following velocity field (Liovic et al., 2006):

$$u = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \quad (74)$$
$$v = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \quad (75)$$
$$w = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T). \quad (76)$$

We have plotted the $\mathbf{L}1$ error for the 2D case as a function of number of cycles in Fig. 17. It can be seen that the error is below $2.5 \times 10^{-5}$ after 15 cycles and this test case verifies the gas-particle coupling as well as associated interpolation schemes work well in 2D and a similar analysis can be performed for the 3D case.

### 4.6 Particle Motion in Vortex

**Directory: mfix/tests/dem-tests/particle-vortex**




(a) $t = 0$ (b) $t = T/5$ (c) $t = 2T/5$

(d) $t = 3T/5$ (e) $t = 4T/5$ (f) $t = T$

(g) $t = 2T$ (h) $t = 16T$

Figure 16: Figure showing the gas velocity vectors along with particle locations at $t = 0$, $T/5$, $2T/5$, $3T/5$, $4T/5$, $T$, $2T$, and $16T$).

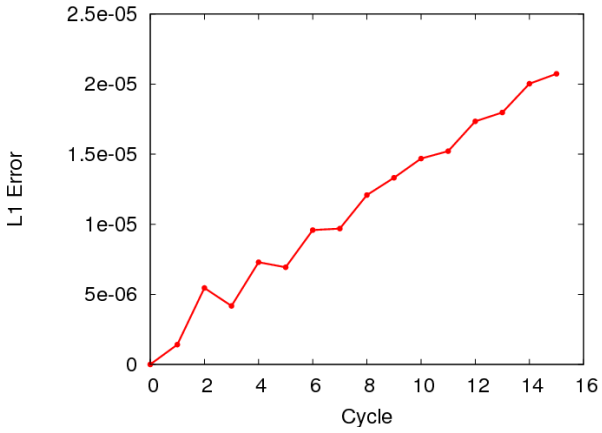



Figure 17: $L1$ error as a function of the number of the cycles of the imposed oscillating vortex field

This case studies the motion of particles in a two-dimensional Talylor-Green vortex flow. The gas velocity components in a 2D Taylor-Green vortex flow are

$$u_g = -\cos(k_x x)\sin(k_y y), \tag{77}$$

in the $x$- direction, and

$$v_g = \sin(k_x x)\cos(k_y y), \tag{78}$$

in the $y$- direction. In the above equations, $k_x$ and $k_y$ are the wavenumber of Taylor-Green vortices.

The extent of gas-particle interaction with the vortex flow depends on the relaxation time of the particle compared to the time available for particle-fluid interaction, characterized by Stokes number as

$$\mathrm{St} = \frac{\tau_p}{\tau_f}, \tag{79}$$

where $\tau_p$ is the particle relaxation time (also know as aerodynamic response time) defined as

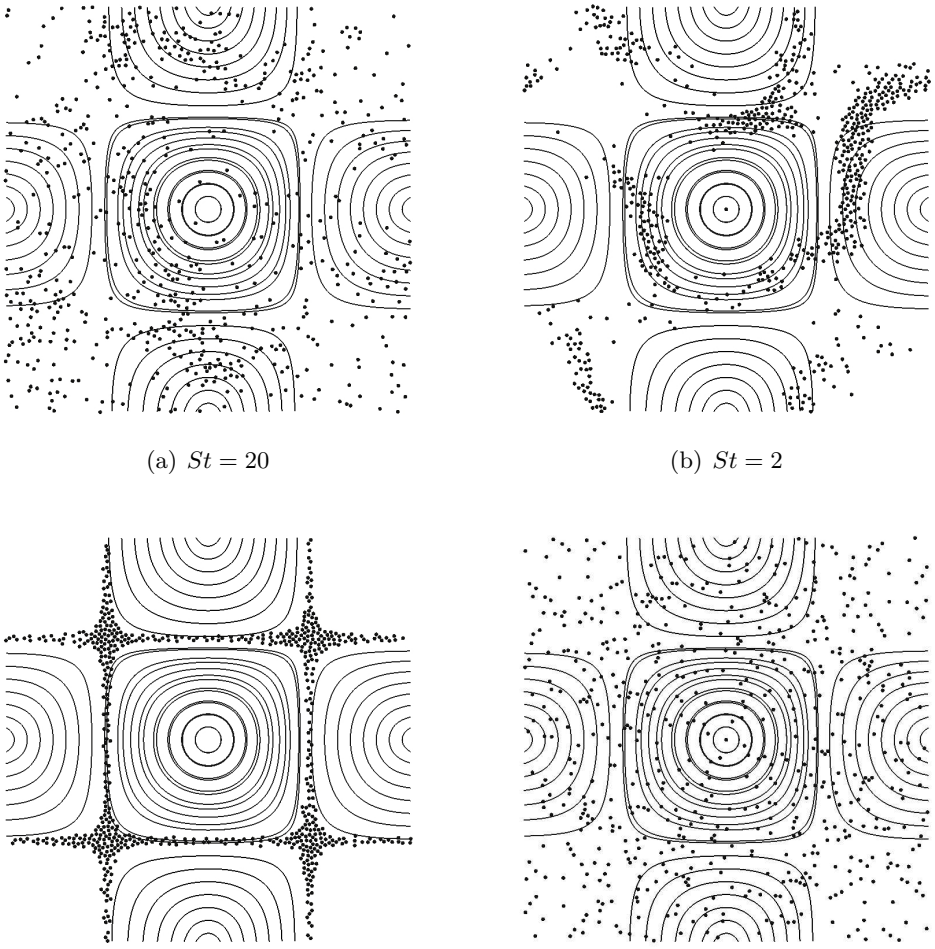
$$\tau_p = \frac{\rho_p {d_p}^2}{18\mu_g} \tag{80}$$

and $\tau_f$ is the response time for the flow, estimated by

$$\tau_f = \frac{L}{U} \tag{81}$$

where $L$ and $U$ are the characteristic length and characteristic velocity of the flow, respectively.

Figure 18 shows the flow patterns of particles in Taylor vortex at different Stokes numbers. For very small particles with $\mathrm{St} \ll 1$ (see Figure18(d)), they tend to be in dynamic equilibrium with the carrier fluid and follow the streamlines of the flow closely, hence particles movements are strongly controlled by the vortex structure. For large particles with $\mathrm{St} \gg 1$, they are barely affected by the flow field due to their large inertia and are persistent in maintaining their own movement as shown in Figure 18(a). However, for $\mathrm{St} \sim 1$ (Figures 18(b) and 18(c)), the particles tend to be centrifuged from the vortex cores and accumulate at the edge of the vortices.



(a) $St = 20$ (b) $St = 2$

(c) $St = 0.2$ (d) $St = 0.002$

Figure 18: Snapshot of solid particles in Taylor-Green vortex for different Stokes numbers. Solid lines represent the gas-flow streamlines and dots represent the solid particles.

# 5 Summary

Multiphase flows are prevalent in different natural phenomenon and various industrial processes. While these flows are practically important, they are also extremely complex and this is largely due to their multiscale nature: flows may span multiple time and length scales. As discussed in the introduction many Computational Fluid Dynamic (CFD) codes have been developed that attempt to predict the hydrodynamics and related characteristics of multiphase flows in order to provide insights into the systems. These codes may differ in their mathematical modeling approach and/or solution technique. The focus of this document has been on the continuum discrete method capability of the open source code MFIX, that is, MFIX–DEM. The underlying theory (e.g., governing equations & physical models) was presented first, followed by the numerical implementation and a series of verification tests.

An important step prior to the application of any model is verification and validation of that model. Verification refers to the process of evaluating the numerical accuracy of a model (Grace and Taghipour, 2004) where the accuracy of the solution algorithm can be assessed by applying the model to problems for which the solution is already known (e.g., via an analytical solution of a limiting case). This approach should reveal whether the code contains errors but does not guarantee that it is completely correct. That is, the code may show agreement with the solution for one test problem, but disagreement with that of another untried test problem which may invoke different components of the code. Another obstacle in conducting verification is that relatively few problems are available in multiphase flows in which an exact solution is available (Grace and Taghipour, 2004). With this in mind, the current effort included a series of test cases, of varying complexity, which were selected for their ability to test different aspects of the code.

Cases 1 and 2 (freely falling particle and two stacked particles) targeted the implementation of the normal collision model and the time stepping algorithm. Case 3 (ball slipping) targeted implementation of the tangential force model. Case 4 (terminal velocity) was slightly more complex than the first three cases and served as a relatively simple test of the drag force. For this case, the code was invoked both with and without coupling to the fluid phase. The final two test cases were again more complex and were designed to target the interpolation routines, which are used when the particles and fluid are coupled. All of these cases demonstrate fairly good agreement with the corresponding analytic solution (when available) or yielded the anticipated behavior for the problem.

Practically speaking, full verification is not possible (one cannot prove that numerical formulation and corresponding code is free of bugs), however, an acceptable level of confidence in the CDM model was pursued. Additional test cases may be developed in the future and applied to test other aspects of the code. Moreover, as the code may continue to evolve and as new features are added, aptly designed test cases will be needed to verify the new code.

While verification has been the primary focus, validation is another important step prior to using the model for physical insights. Validation refers to the process of assessing the ability of a (verified) model to accurately predict the physical phenomena observed experimentally (Grace and Taghipour, 2004). Good validation involves testing the model against data for a wide range of conditions. Context, however, is also important in the validation process as different applications may require different degrees of validation. For example, a model may be validated on the basis of showing correct trends but not for purposes of engineering design. Like verification, full validation is not practically possible as some future experiment may show deficiencies in the model which had previously gone undetected (Grace and Taghipour, 2004). The current effort did not include any validation studies and this is left for future work and to those who desire to use the code for simulating real systems.

# 6 Future Work

As mentioned MFIX–DEM is an evolving code. While a well-verified basic foundation has been created for performing CDM simulations, areas in need of additional development and/or investigation exist. Some possible topics/areas are listed below.

Coarse graining capabilities

Electrostatics

Improved neighbor search algorithms

Heat and mass transfer

Chemical reactions

Lee-Edwards boundary conditions

Efficient parallel implementation

This is not an all inclusive list and user input is welcome (contact the MFIX–DEM email list: dem@mfix.netl.doe.gov). The intention of this list is to direct those who may be interested to areas in need of investigation and/or development.

## References

Allen, M. P., Tildesley, D. J., 1989. Computer Simulation of Liquids. Oxford University Press, Oxford, United Kingdom.

Amsden, A. A., O'Rourke, P. J., Butler, T. D., May 1989. KIVA–II: A Computer Program for Chemically Reactive Flows with Sprays. Tech. Rep. LA–11560–MS, Los Alamos National Laboratory.

Anderson, T. B., Jackson, R., 1967. A fluid mechanical description of fluidized beds. Ind. Eng. Chem. Fundam. 6, 527–539.

Bird, G. A., 1994. Molecular gas dynamics and the direct simulation of gas flows. No. 42 in Oxford engineering science series. Clarendon Press, Oxford.

Boyalakuntla, D. S., Pannala, S., 2006. Summary of discrete element model (dem) implementation in mfix. Tech. rep., Oak Ridge National Laboratory, From URL http://www.mfix.org/documents/MFIXDEM2006-4-1.pdf.

Boylakunta, D. J., 2003. Simulation of granular and gas-solid flows using discrete element method. Ph.D. thesis, Carnegie Mellon University.

Chen, F., 2009. Coupled Flow Discrete Element Method Application in Granular Porous Media using Open Source Codes. Ph.D. thesis, The University of Tennessee, Knoxville.

Chen, F., Drumm, E. C., Guiochon, G., 2007. Prediction/Verification of Particle Motion in One Dimension with the Discrete-Element Method. International journal of geomechanics 7, 344–352.

Cundall, P. A., Strack, O. D. L., 1978. The Distinct Element Method as a Tool for Research in Granular Media. Tech. Rep. NSF Grant ENG76-20711, National Science Foundation.

Drew, D. A., 1971. Average field equations for two–phase media. Stud. Appl. Math. 50, 133–166.

Drew, D. A., 1983. Mathematical modeling of two–phase flow. Annu. Rev. Fluid Mech. 15, 261–291.

Drew, D. A., Passman, S. L., 1998. Theory of Multicomponent Fluids. Applied Mathematical Sciences. Springer, New York.

Dziugys, A., Peters, B., 2001. An approach to simulate the motion of spherical and non-spherical fuel particles in combustion chambers. Journal of Granular Matter 3 (4), 231–266.

Frankl, F. I., 1953. On the theory of motion of suspended sediments. Dokl. Akad. Nauk. SSSr, 92–247.

Galizzi, O., Kozicki, J., 2005. YADE-Yet another dynamic engine. Tech. rep., Available at:http:yade.berlios.de.

Garzo, V., Dufty, J. W., Hrenya, C. M., 2007. Enskog theory for polydisperse granular mixtures. i. navier-stokes order transport. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 76 (3), 031303.

Grace, J. R., Taghipour, F., 2004. Verification and validation of CFD models and dynamic similarity for fluidized bed. Powder Technology 139, 99–110.

Hindmarsh, A. C., 1983. Odepack, a systematized collection of ode solvers , r. s. stepleman et al. (eds.), north-holland, amsterdam, (vol. 1 of ), pp. 55-64. IMACS Transactions on Scientific Computation 1, 55–64.

Itasca, I., Accesssed January 2010. Fixed coarse-grid fluid scheme in PFC2D, The PFC2D user's manual. Available at http:www.itascacg.comhome.php.

Kashiwa, B., Rauenzahn, R. M., 1994. A Multimaterial Formalism. Tech. Rep. LA-UR-94-771, Los Alamos National Lab.

Kataoka, I., Serizawa, A., 1989. Basic equations of turbulence in gas–liquid two–phase flow. Intl. J. Multiphase Flow 15 (5), 843–855.

Khakimov, Z. M., 2002. New integrator for molecular dynamics simulations. Computer Physics Communications 147 (1-2), 733 – 736.
URL http://www.sciencedirect.com/science/article/B6TJ5-45NGNS5-C/2/8b1ebb4f5da27853fa5d33d92

Leveque, R. J., 1996. High-resolution conservative algorithms for advection in incompressible flow. SIAM Journal on Numerical Analysis 33 (2), 627–665.

Liovic, P., Rudman, M., Liow, J. L., Lakehal, D., Kothe, D., 2006. A 3d unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction. Computers and Fluids 35 (10), 1011–1032.

Omelyan, I. P., Mryglod, I. M., Folk, R., May 2002. Optimized verlet-like algorithms for molecular dynamics simulations. Phys. Rev. E 65 (5), 056706.

O'Rourke, P., Amsden, A. A., 1987. The TAB Method for Numerical Calculation of Spray Droplet Breakup. SAE Paper 872089.

Rider, W. J., Kothe, D. B., 1998. Reconstructing volume tracking. Journal of Computational Physics 141 (2), 112–152.

Rougier, E., Munjiza, A., John, N. W. M., 2004. Numerical comparison of some explicit time integration schemes used in dem, fem/dem and molecular dynamics. International Journal for Numerical Methods in Engineering 61 (6), 856–879.

Savage, S. B., Jeffrey, D. J., 1981. The stress tensor in a granular flow at high shear rates. J. Fluid Mech. 110, 255–272.

Schiller, L., Naumann, A. Z., 1933. A Drag Coefficient Correlation. Z. Ver. Deutsch Ing., 318–320.

Silbert, L., Ertas, D., Grest, G., Halsey, T., Levine, D., Plimpton, S., 2001. Granular flow down an inclined plane: Bagnold scaling and rheology. Physical Review E 64.

Subramaniam, S., 2000. Statistical representation of a spray as a point process. Phys. Fluids 12 (10), 2413–2431.

Syamlal, M., 1998. Mfix documentation: Numerical guide. Tech. Rep. DOE/MC31346-5824, NTIS/DE98002029, National Energy Technology Laboratory, Department of Energy, see also URL http://www.mfix.org.

Syamlal, M., Rogers, W., O'Brien, T. J., 1993. Mfix documentation: Theory guide. Tech. Rep. DOE/METC-95/1013, NTIS/DE95000031, National Energy Technology Laboratory, Department of Energy, see also URL http://www.mfix.org.

Teletov, S. G., 1958. Problems of the hydrodynamics of two–phase mixtures. I. Vestn. Mosk. Gos. Univ., Ser. Mat. Mekh. Astron. Fiz. Khim. 2, 15–27.

van der Hoef, M. A., van Sint Annaland, M., Deen, N. G., Kuipers, J. A. M., 2008. Numerical simulation of dense gas-solid fluidized beds: A multiscale modeling strategy. Annu. Rev. Fluid Mech. 40, 47–70.

Weber, M., 2004. Simulation of cohesive particle flows in granular and gas-solid systems. Ph.D. thesis, University of Colarado.

Williams, F. A., 1958. Spray combustion and atomization. Phys. Fluids 1 (6), 541–545.

# A Gas-phase pressure correction for MFIX-DEM

***Contributed by Jin Sun.*** **(email: jsunia@gmail.com)**

The discretization equation for fluid pressure is derived for the special case where solid particle dynamics is solved by DEM method. The derivation procedure follows the corresponding part in the MFIX numerics documentation (Syamlal, 1998).

The discretized x-momentum equations for fluid phases, for example, is

$$a_{0p}(u_0)_p = \sum_{nb} a_{0nb}(u_0)_{nb} + b_0 - A_p(\varepsilon)_p \left[(P_g)_E - (P_g)_W\right] + F_{10}\left[(u_1)_p - (u_0)_p\right]\Delta V. \tag{82}$$

Use the pressure field $P_g^*$ and void fraction field $\varepsilon_0^*$ from the previous iteration to calculate tentative values of the velocity fields:

$$a_{0p}(u_0^*)_p = \sum_{nb} a_{0nb}(u_0^*)_{nb} + b_0 - A_p(\varepsilon_0^*)_p \left[(P_g^*)_E - (P_g^*)_W\right] + F_{10}\left[(u_1^*)_p - (u_0^*)_p\right]\Delta V. \tag{83}$$

The solid velocity $(u_1^*)_p$ is simply treated as the value from the previous time step and the value does not change during the iteration process, i.e., $(u_1^*)_p = (u_1)_p$, since the solid momentum equation is not solved during this process. Equation 83 is thus re-written as

$$(a_{0p} + F_{10}\Delta V)(u_0^*)_p = \sum_{nb} a_{0nb}(u_0^*)_{nb} + b_0 - A_p(\varepsilon_0^*)_p \left[(P_g^*)_E - (P_g^*)_W\right] + F_{10}(u_1)_p\Delta V. \tag{84}$$

Let the actual values differ from the starred values by the following corrections

$$\begin{aligned}
(P_g)_E &= (P_g^*)_E + (P_g')_E,\\
(P_g)_W &= (P_g^*)_W + (P_g')_W,\\
(u_0)_p &= (u_0^*)_p + (u_0')_p,\\
(u_0)_{nb} &= (u_0^*)_{nb} + (u_0')_{nb}.
\end{aligned} \tag{85}$$

Substituting the corrections in Eq. 85 into Eq. 82 and subtracting Eq. 84 from the resulting equation results in

$$(a_{0p} + F_{10}\Delta V)(u_0')_p = \sum_{nb} a_{0nb}(u_0')_{nb} - A_p(\varepsilon_0^*)_p \left[(P_g')_E - (P_g')_W\right]. \tag{86}$$

Neglecting the convection term in the above equation results in

$$(a_{0p} + F_{10}\Delta V)(u_0')_p = -A_p(\varepsilon_0^*)_p \left[(P_g')_E - (P_g')_W\right]. \tag{87}$$

Therefore, $u_0'$ becomes

$$(u_0')_p = -\frac{A_p(\varepsilon_0^*)_p}{(a_{0p} + F_{10}\Delta V)} \left[(P_g')_E - (P_g')_W\right], \tag{88}$$

which can be rewritten as

$$(u_0')_p = -d_{0p}\left[(P_g')_E - (P_g')_W\right], \tag{89}$$

where

$$d_{0p} = \frac{A_p(\varepsilon_0^*)_p}{(a_{0p} + F_{10}\Delta V)}. \tag{90}$$

Therefore, the velocity correction is given by

$$(u_0)_p = (u_0^*)_p - d_{0p}\left[(P_g')_E - (P_g')_W\right]. \tag{91}$$

The rest of derivation is the same as that in the MFIX numerics documentation (Syamlal, 1998). The resulting pressure correction equation is also in the same form except that $d_{0p}$ is given by Eq. 90

# B MFIX–DEM file list with purpose

Below is the List of all the files in /model/des and their purpose
NOTE: The equations solved in a subroutine are mentioned in the header of each file and they are also added to the Doxygen document appended to this document

| | |
|---|---|
| **calc_force_des.f** | Call all the subroutines needed to compute the inter-particle collision force on each particle (Eq. 26) due to its neighbors. |
| **cfassign.f** | Assign the necessary values to each particle (such as, particle volume, mass, moment of inertia) for DEM computation. Boundary conditions are assigned based on the input. Assigning DEM gravity vector from MFIX input. Calculating damping coefficients (such as $\eta_{nm\ell}$ from Eq. 30) and collision time $t^{col}_{n,m\ell}$ (Eq. 29) for particle–particle and particle–wall collisions. Finally calculate DTSOLID, which is equal to one–fifty of the minimum collision time. |
| **cffctowall.f** | Calculate the total contact force and Torque on a particle in a particle-wall collision. |
| **cffctow.f** | Calculate the total contact force and torque on a particle in a particle-particle collision. |
| **cfnewvalues.f** | Calculate the new values of position and velocity for the current time step from the values at previous time step. This is done explicitly. |
| **cfnocontact.f** | The subroutine sets all forces on a particle to zero if the particle is found to have no neighbors (either particles or walls). |
| **cfrelvel.f** | Calculate the relative velocity $\mathbf{V}_{ij}$ (Eq. 11), and its normal $\mathbf{V}_{nij}$ (Eq. 14) and tangential $\mathbf{V}_{tij}$ (Eq. 15) components. Also calculate the normal $\boldsymbol{\eta}_{ij}$ (Eq. 10) and tangential $\boldsymbol{t}_{ij}$ (Eq. 16) vectors in the plane of contact. |
| **cfslide.f** | Check for Coulomb's friction law (Eq. 24) and limit the maximum value of the tangential force on a particle in contact with another particle. |
| **cfslidewall.f** | Check for Coulomb's friction law and limit the maximum value of the tangential force on a particle in contact with a wall. |
| **cfupdateold.f** | Update the old values of particle position and velocity with the new values computed. |
| **cfwallcontact.f** | Check to see if a particle is in contact with any of the walls |
| **cfwallposvel.f** | Assign the wall particle a position and velocity. |
| **des_allocate_arrays.f** | Dynamic memory allocation for DEM arrays |
| **des_functions.f** | DEM dot product function and cross product sub routine |

| | |
|---|---|
| **des_granular_temperature.f** | Calculate the granular temperature for the DEM particles |
| **des_init_arrays.f** | Initialize DEM arrays. |
| **des_init_namelist.f** | Initialize DEM variable name list. |
| **des_time_march.f** | Time marching for solids treated using DEM. |
| **discretelement_mod.f** | Module containing the variable declaration. |
| **drag_fgs.f** | Calculate the drag force and pressure force on each particle due to the surrounding fluid. |
| **gas_drag.f** | Enter the drag terms into Am and Bm matrices for gas phase calculations. |
| **generate_particle_config.f** | Generates an initial lattice distribution. |
| **grid_based_neighbor_search.f** | Cell linked–list based particle neighbor search algorithm for both periodic and non–periodic boundary conditions. |
| **make_arrays_des.f** | Read the initial particle position, velocity, radius and density details. |
| **neighbour.f** | Perform neighbor search (n-square or quadtree/octree). |
| **nsquare.f** | Perform n-square neighbor search. |
| **octree.f** | Perform octree search (3D only) |
| **particles_in_cell.f** | Locate the fluid cell in which each particle lies in order to compute the solids volume fraction in that cell and hence the cell void fraction. |
| **quadtree.f** | Perform quadtree search (2D only) |
| **write_des_data.f** | Write DEM output files in Paraview compatible format. |
| **write_des_restart.f** | Write DEM restart file |
| **read_des_restart.f** | Read DEM restart file |
| **randomno_mod.f** | Generate uniform and normally distributed random variates |

# C MFIX–DEM user input variables

The DEM User-Input variables are listed below.

| VARIABLE | TYPE | DESCRIPTION |
|---|---|---|
| DISCRETE_ELEMENT [F] | L | Use discrete particle model for solids. Must be TRUE to do DEM. |
| DES_CONTINUUM_COUPLED [F] | L | Couple gas and solids flow together. |
| DES_INTERP_ON [F] | L | Use an interpolation suite to calculate the drag force on each particle based on particle location (Eq. 35) rather than cell averages |

| | | |
|---|---|---|
| DES_INTG_METHOD<br>[EULER] | C | Time stepping scheme<br>EULER - First order Euler scheme<br>ADAMS_BASHFORTH - Second order Adams_Bashforth scheme |
| DIMN<br>[UNDEFINED_I] | I | Specify the dimension of the simulation: 2 or 3. If NO_K = '.TRUE.', then DIMN will automatically be set to 2. |
| GENER_PART_CONFIG<br>[F] | L | Automatically generate an initial particle configuration (position) otherwise use particle_input.dat. Also requires setting VOL_FRAC(M), D_P0(M), and DES_EPS_XSTART, DES_EPS_YSTART, and DES_EPS_ZSTART. Once defined this feature will determine the total number of particles in the system and their initial placement. particle_input.dat file is ignored. |
| VOL_FRAC(m)<br>[UNDEFINED] | DP | Only relevant when GENER_PART_CONFIG is T. Volume fraction of the solid phase M for generating particles in the specified domain |
| DES_EPS_XSTART<br>[UNDEFINED] | DP | Only needed if GENER_PART_CONFIG. Length of the domain in the x direction wherein particles may be initially placed. |
| DES_EPS_YSTART<br>[UNDEFINED] | DP | Only needed if GENER_PART_CONFIG. Length of the domain in the y direction wherein particles may be initially placed. |
| DES_EPS_ZSTART<br>[UNDEFINED] | DP | Only needed if GENER_PART_CONFIG. Length of the domain in the z direction wherein particles may be initially placed. |
| NFACTOR<br>[10] | I | Only needed if DES_CONTINUUM_COUPLED. Number of times a pure DEM simulation is run before the coupled DEM simulation is started (allows settling). |
| TSUJI_DRAG<br>[F] | L | Use Tsuji's drag correlation |
| PARTICLES<br>[UNDEFINED_I] | I | Total number of particles |
| PVEL_MEAN<br>[0] | DP | Assign initial particle velocities from a Guassian distribution with the specified mean. Only relevant if PVEL_STDEV is not zero. If used, the assigned velocities will override any other initial settings. |

| | | |
|---|---|---|
| PVEL_STDEV<br>[0] | DP | If not zero, then assign initial particle velocities from a Guassian distribution with the specified standard deviation. Used with PVEL_MEAN. If used, the assigned velocities will override any other initial settings. |
| PARTICLES_FACTOR<br>[1.2] | DP | Expand the size of the particle arrays by an arbitrary factor (multiple of the number of particles). |
| FACTOR_RLM<br>[1.2] | DP | Effectively increase the radius of a particle (multiple of the sum of particle radii) for detecting neighbor contacts when using grid based neighbor search or n-square search methods (see Sec. 3.2 for details). |
| **Boundary Conditions** | | |
| WALLDTSPLIT<br>[F] | L | Treat wall interaction as a two-particle interaction but accounting for the wall properties. Must be TRUE for DEM. |
| DES_PERIODIC_WALLS<br>[F] | L | Periodic wall boundary condition is imposed on any pair of walls. |
| DES_PERIODIC_WALLS_X<br>[F] | L | Direction of periodicity: X |
| DES_PERIODIC_WALLS_Y<br>[F] | L | Direction of periodicity: Y |
| DES_PERIODIC_WALLS_Z<br>[F] | L | Direction of periodicity: Z |
| **Neighbor Search Parameters** | | |
| DES_NEIGHBOR_SEARCH<br>[1] | I | Neighbor search algorithm. 1=N-square; 2=quadtree (for 2D only); 3=octree (for 3D only); 4=grid based. Options 2 and 3 have not been recently tested. Use them at your own risk. |
| NEIGHBOR_SEARCH_N<br>[1] | I | Maximum number of steps through a DEM loop before a neighbor search will be performed. (Search may be called earlier). |
| MN<br>[10] | I | Maximum number of neighbors per particle |
| QLM<br>[1] | I | Number of levels to traverse “up” to move a particle to its new quad. Only needed when using octree or quadtree based neighbor search. |
| QLN<br>[1] | I | Number of levels to traverse “up” to perform particle neighbor search. Only needed when using octree or quadtree based neighbor search. |

| | | |
|---|---|---|
| INIT_QUAD_COUNT [UNDEFINED_I] | I | Count to initialize quadtree or octree. Only needed when using octree or quadtree based neighbor search. |
| MQUAD_FACTOR [1.1] | DP | Factor to create quadtree or octree arrays based on the number of particles. Only needed when using octree or quadtree based neighbor search. |
| NEIGHBOR_SEARCH_RAD_RATIO [1] | DP | Ratio of the distance (imaginary sphere radius) to particle radius that is allowed before a neighbor search is performed |
| **Particle-particle and Particle-wall contact parameters** | | |
| DES_COLL_MODEL [UNDEFINED_C] | C | Collision model for the soft-sphere approach. By default, the linear spring-dashpot (LSD) model is used (i.e., leave DES_COLL_MODEL undefined for LSD model).<br>Other models include: HERTZIAN.<br>All models require specifying the following parameters: DES_EN_INPUT, DES_EN_WALL_INPUT, MEW, and MEW_W.<br>The default (LSD) model requires: KN, KN_W, KT_FAC, KT_W_FAC, DES_ETAT_FAC, & DES_ETAT_W_FAC.<br>The HERTZIAN model requires: DES_ET_INPUT, DES_ET_WALL_INPUT, E_YOUNG, EW_YOUNG, V_POISSON, & VW_POISSON. |
| DES_EN_INPUT [UNDEFINED] | DP | The normal restitution coefficient for inter-particle collisions that is used to determine the inter-particle normal damping factor (see Sec. 2.2.2 above or routine cfassign.f for details). Values are stored as a one dimensional array (see Eq. 32). So if MMAX=3, then 6 values are needed, which are defined as follows: $e_{n11}$ $e_{n12}$ $e_{n13}$ $e_{n22}$ $e_{n23}$ $e_{n33}$. |
| DES_ET_INPUT [UNDEFINED] | DP | Tangential restitution coefficient for inter-particle collisions. Values are stored as a one dimensional array. Only needed when using the Hertzian collision model. |
| DES_EN_WALL_INPUT [UNDEFINED] | DP | Normal restitution coefficient for particle-wall collisions that is used to determine the particle-wall normal damping factor (see cfassign.f for details). Values are stored as a one dimensional array. So, if MMAX=3, then 3 values are needed, which are defined as follows: $e_{nw1}$ $e_{nw2}$ $e_{nw3}$. |

| | | |
|---|---|---|
| DES_ET_WALL_INPUT<br>[UNDEFINED] | DP | Tangential restitution coefficient for particle-wall collisions. Values are stored as a one dimensional array. Only needed when using the Hertzian collision model. |
| KN<br>[UNDEFINED] | DP | Normal spring constant for inter-particle collisions. Values are stored as a one dimensional array. Needed when using the default collision (LSD) model. |
| KT_FAC<br>[2/7] | DP | Ratio of the tangential spring constant to normal spring constant for inter-particle collisions. Use it to specify the tangential spring constant for particle-particle collisions as KT_FAC*KN. |
| DES_ETAT_FAC<br>[0.5] | DP | Ratio of the tangential damping factor to the normal dampign factor for inter-particle collisions. Needed when using the default collision model. |
| KN_W<br>[UNDEFINED] | DP | Normal spring constant for particle-wall collisions. Needed when using the default collision model. |
| KT_W_FAC<br>[2/7] | DP | Ratio of the tangential spring constant to normal spring constant for particle-wall collisions. Use it to specify the tangential spring constant for particle-wall collisions as KT_W_FAC*KN_W. |
| DES_ETAT_W_FAC<br>[0.5] | DP | Ratio of the tangential damping factor to the normal dampign factor for particle wall collisions. Needed when using the default collision model. |
| MEW<br>[UNDEFINED] | DP | Particle friction coefficient |
| MEW_W<br>[UNDEFINED] | DP | Particle-wall friction coefficient |
| E_YOUNG<br>[UNDEFINED] | DP | Young's modulus for the solid phase. Only needed when using the Hertzian collision model. |
| V_POISSON<br>[UNDEFINED] | DP | Poisson ratio for the solid phase. Only needed when using the Hertzian collision model. |
| EW_YOUNG<br>[UNDEFINED] | DP | Young's modulus for the wall. Only needed when using the Hertzian collision model. |
| VW_POISSON<br>[UNDEFINED] | DP | Poisson ratio for the wall. Only needed when using the Hertzian collision model. |
| **Output and Restart Control** | | |
| DEBUG_DES<br>[F] | L | Print out additional information from DEM model. |

| | | |
|---|---|---|
| FOCUS_PARTICLE<br>[0] | I | If DEBUG_DES, then additional information will be printed for the specified particle number. |
| PRINT_DES_DATA<br>[F] | L | Print DEM output |
| DES_RES_DT<br>[UNDEFINED] | DP | If PRINT_DES_DATA, this is the frequency at which DES.RES and .RES files will be written. This only applies to pure granular simulations, otherwise for coupled simulation the restart frequency is controlled by RES_DT. |
| DES_SPX_DT<br>[UNDEFINED] | DP | IF PRINT_DES_DATA, this is the frequency at which DEM data will be written. This only applies to pure granular simulations, otherwise for coupled simulation the output frequency is controlled by SPX_DT(1) |
| DEM_OUTPUT_DATA_TECPLOT<br>[F] | L | Write several .dat files (e.g. _DES_DATA.dat & _AVG_EPS.dat) rather than the standard .vtk files. See write_des_data.f for details. |
| | | |