Summary of Discrete Element Model (DEM) implementation in MFIX

D. S. Boyalakuntla

Oak Ridge National Laboratory (Currently at Widener University)

S. Pannala

Oak Ridge National Laboratory

December 2006

Table of Contents

Abstract

Discrete element model (DEM) also referred to as discrete particle method (DPM), discrete element method etc. is a technique to simulate granular flows by tracking the motion of individual particles. This is unlike the continuum theory approach where the discrete particles are treated as a solids phase continuum with averaged properties. In DEM the particles are treated individually by accounting for all the forces on each particle and the resulting momentum changes when they move and collide. In the fluidized bed scenarios, which are of main interest in the current work, the forces on each particle are mainly due to inter particle collisions and due to the drag imposed by the surrounding gas. Two DEM models that explicitly consider the particulate nature of granular materials and the particulate interactions are the *hard sphere model* and the *soft sphere model*. In the present work the soft-sphere model has been used.

1. Hard Sphere Model

In dilute gas-solids flows where the particulate phase is quite dispersed, the inter-particle collisions are analogous to molecular collisions in rarefied gases. Though the particle collisions are inelastic, energy is re-supplied to the system through particle acceleration, gravity, or through collision with a moving boundary. Most published literature has considered only binary collisions for rapid granular flow regimes. Particles move in well-defined trajectories until collision with another particle occurs. These interactions are modeled as instantaneous collisions and the post-collision state of the particles is determined from classical dynamics (Campbell, 1982). The hard sphere model considers the Newtonian equations of motion in the integral form and is based on the impulsive force, which is defined by the integral of the forces acting on a particle versus time. The equations and detailed description of hard sphere model can be found in Campbell (1982) and Crowe et al (1998). This method is not implemented in MFIX primarily because of the need to simulate enduring contacts between particles, for which the hard sphere model is not suitable.

2. Soft Sphere Model

Most of the commonly observed gas-solids flows are dense particle flows and exhibit multiple-particle long-duration contacts. The soft sphere model, proposed by Cundall and Strack (1979) is better suited than hard sphere model in such regimes. Though the soft sphere model is computational expensive than the hard sphere model, some information like the inter-particle forces obtained in this approach can not be obtained in the hard sphere model. Also the hard sphere model breaks down in systems with long inter-particle contact durations (Campbell, 1982; Crowe *et al.*, 1998). Presently MFIX has the soft sphere implementation of DEM.

2.1 Particle-Particle Interaction

The soft sphere model starts with a set of differential equations and the variations in momentum and displacement are obtained for arbitrary times as solutions to the differential equations. Once the forces acting on each particle in the system are calculated, using Newton's second law, the acceleration of each particle is calculated. Then, by integrating the acceleration in time, new particle states are obtained in terms of the velocity and position. When two elastic particles collide, in reality, they deform. The soft sphere model assumes an overlap displacement δ as shown in Figure 1, as a representation of the particle deformation during contact.

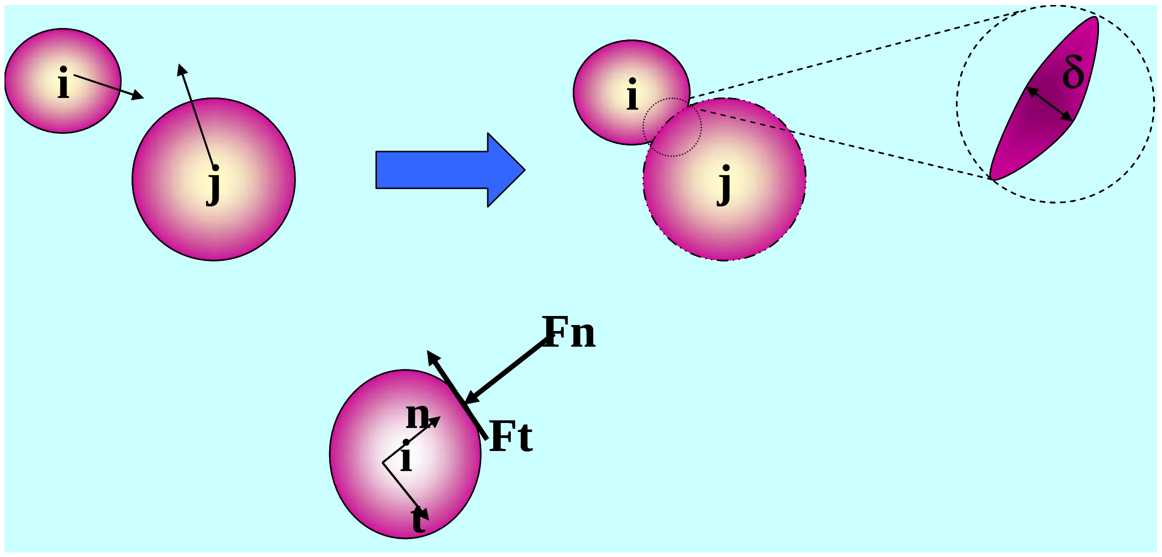

Figure 1. Particle-particle collision in a soft sphere model

The inter-particle contact forces, namely, the normal, damping and sliding forces, act on the two particles and are modeled in the soft sphere model using the mechanical elements springs, dash-pots, and sliders respectively as shown in Figure 2. The normal contact between two particles is modeled as a linear spring in parallel with a dashpot element. The spring provides an elastic restoration force while the dashpot dissipates energy during contact. As a result the effective coefficient of restitution can be less than one. The tangential contact model is slightly more complex. In the tangential model the spring is in series with a Coulombic friction sliding element. The spring allows the particle to respond elastically, while the sliding friction element allows particles to slide against each other. The magnitude of tangential force is limited by the sliding element.

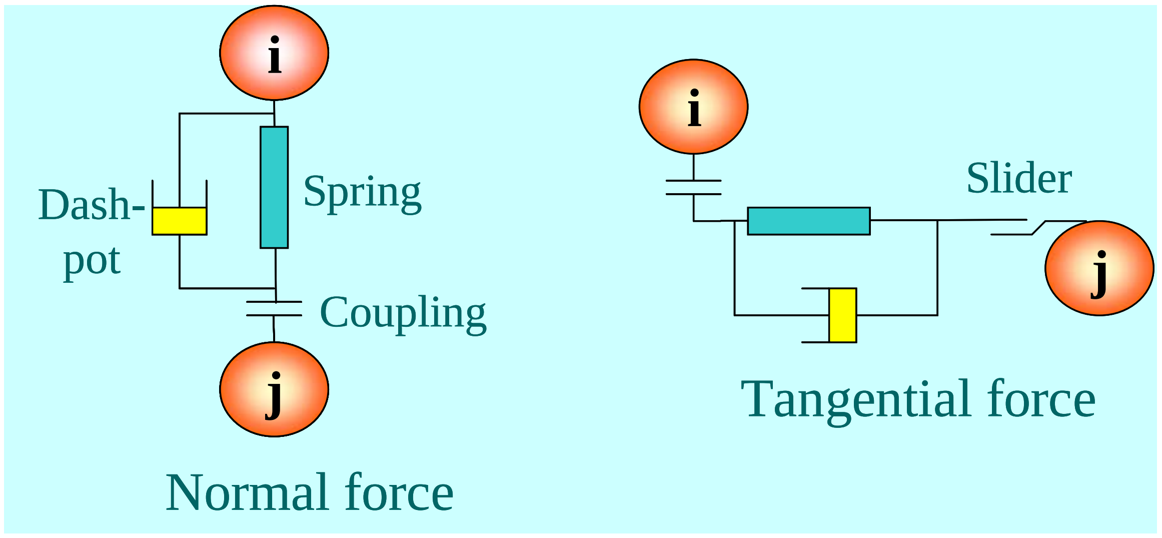

Figure 2. Spring-damper system to model contact forces

The effects of these mechanical elements on particle motion appear in mathematical equations through the spring stiffness constant k, the damping coefficient η, and the friction coefficient μ_f. Both translational and rotational motions are considered. The translational motion is caused by contact force, interaction forces with the fluid and gravitational force. Only contact forces are considered to cause rotational motion. From Newton's second law, the translational and rotational accelerations are given by

$$\ddot{\vec{r}} = \frac{\vec{F}}{m} + \vec{g} \quad \text{...(1)}$$

$$\dot{\vec{\omega}} = \frac{\vec{T}_C}{I} \quad \text{...(2)}$$

where $\vec{r}$ is the position vector of the particle center of gravity, m is the particle mass, $\vec{F}$ is the total force acting on the particle, $\vec{g}$ is the gravity vector, $\vec{\omega}$ is the rotational velocity, $\vec{T}_C$ is the total torque caused by contact forces, I is the moment of inertia ($=\frac{2}{5}ma^2$ for a sphere of radius a). ($\dot{}$) denotes the time derivative. The above equations are implemented in the routine mfix**/model/des/cfnewvalues.f**.

The contact force can be written as:

$$\vec{F}_C = \vec{f}_C + \vec{f}_D \quad \text{...(3)}$$

where $\vec{f}_C$ is the total force on the particle due to inter-particle contact and $\vec{f}_D$ is the force on the particle due to fluid drag and pressure gradient.

From the spring-damper model, when particle *i* is in contact with particle *j*, the normal component of the contact force, $\vec{f}_{Cnij}$, acting on the particle *i* is given by the sum of forces due to the spring and dash-pot.

$$\vec{f}_{Cnij} = -k\vec{\delta}_{nij} - \eta\vec{v}_{nij} \quad \text{...(4)}$$

where δ_{nij} is the particle displacement caused by the normal force (Figure 1.), $\vec{v}_{nij}$ is the normal component of the relative velocity and is given by $\vec{v}_{nij} = \vec{v}_{rij} \cdot \vec{n}_{ij}$, where $\vec{v}_{rij}$ is the velocity vector of particle *i* with respect to particle *j* and $\vec{n}_{ij}$ is the normal vector between them. The normal force equation is implemented in the routine mfix**/model/des/cffn.f**. The normal vector can be determined from the following equation and is calculated in **cfnormal.f**:

$$n_{ij} = \frac{\vec{x}_j - \vec{x}_i}{\left|\vec{x}_j - \vec{x}_i\right|} \quad \text{...(5)}$$

δ_{nij} is the normal overlap and is the distance traveled by the particle pair with the normal component of the relative velocity in the given time step. The total overlap at a given time is the accumulation of the overlaps over each solids time step since the particles came into contact. This accounts for the change in the particle normal direction during contact, thus accurately capturing the total overlap. The incremental (calculated in mfix**/model/des/cfincrementaloverlaps.f**) in each solid time step is given by:

$$\vec{\delta}_{nij} = \vec{v}_{nij}\Delta t \quad \text{...(6)}$$

It may also be computed directly as the difference between the sum of the radii and the center to center distance $\delta_{nij} = (r_i + r_j) - \left|\vec{x}_i - \vec{x}_j\right|$. The total overlap is calculated in mfix**/model/des/cftotaloverlaps.f**.

The tangential component (calculated in mfix/**model/des/cfft.f**) of the contact force $\vec{f}_{Ctij}$, is given by

$$\vec{f}_{Ctij} = -k\vec{\delta}_{tij} - \eta\vec{v}_{sij} \quad \text{..(7)}$$

where δ_{tij} is the displacement in tangential direction. The subscripts *n* and *t* imply normal and tangential directions, respectively. The slip velocity $\vec{v}_{sij}$ (computed in **mfix/model/des/cfslipvel.f**) is given by

$$\vec{v}_{sij} = \vec{v}_{rij} - \vec{v}_{nij} + r\left(\vec{\omega}_i + \vec{\omega}_j\right) \times \vec{n}_{ij} \quad \text{..(8)}$$

where *r* is the radius of the particle. Also the tangential force should satisfy Coulomb's friction law.

$$\left|\vec{f}_{Ctij}\right| \le \mu_f \left|\vec{f}_{Cnij}\right| \text{...(9)}$$

Else, the value is set (in mfix/**model/des/cfslide.f**) such that $\vec{f}_{Ctij} = -\mu_f \left|\vec{f}_{Cnij}\right| \vec{t}_{ij}$

where the tangent, $\vec{t}_{ij} = \dfrac{\vec{v}_{sij}}{\left|\vec{v}_{sij}\right|}$, is computed in mfix/**model/des/cftangent.f**

Finally the total contact forces (computed in mfix/**model/des/cffctow.f**) on a particle *i* due to its neighbors *j* is the summation of forces due to individual neighbor pairs.

$$\vec{f}_{Ci} = \sum_j \left(\vec{f}_{Cnij} + \vec{f}_{Ctij}\right) \text{...(10)}$$

$$\vec{T}_{Ci} = \sum_j \left(r\vec{n}_{ij} \times \vec{f}_{Ctij}\right) \text{...(11)}$$

2.1.1 Newton's 3rd Law for Inter-Particle Contact Force Calculation

A particle pair in contact has the same overlap and hence the force seen by the neighbors would be equal and opposite. This fact is used to eliminate duplicate force calculations. When particle *i* is tracked and its neighbors *j* are listed, the interactions between *i* and *j* are calculated and the inter-particle collision force on *i* due to *j*, F_{ij} is obtained as described in section 2. An equal and opposite inter-particle force is immediately assigned to the neighbor *j*. This eliminates a repeated calculation of F_{ji}.

2.2 Particle-Wall Interaction

The particle-wall pair is dealt just as a pair of particles, but with a few modifications. All the equations mentioned earlier for two particle interaction hold when the neighbor particle j in the above equations is replaced by wall. The wall particle may have a given velocity if the wall is a moving boundary, but the wall particle is always assigned zero angular velocity. If the wall is stationary,

$$\left|\vec{v}_j\right| = \left|\vec{\omega}_j\right| = 0 \qquad (12)$$

Though in terms of equations a particle-wall pair is treated just as a particle-particle pair, the complexity increases in the way one treats the wall itself. One idea is to treat the wall as a reflecting body. So when a particle comes in contact with the wall it is reflected but the velocity is altered by taking into account the coefficient of restitution.

Another way (which is implemented in MFIX-DEM) is to replace the neighbor *j* with the wall particle. In this case the wall is treated as a particle with infinite radius and infinite mass. Thus the normal between any particle and the wall would always be the normal to the wall itself. A simple way of ensuring that the normal of the particle-wall pair is normal to the wall is to dynamically declare a new particle of the same size as that of particle *i* whenever it comes in contact with the wall. The new particle has the same translational velocity as that of the wall and its center is located at a particle radius away from the wall, outside the domain (see Figure 3). It is also ensured that the normal between the particle and the wall particle is same as the normal to the wall. Since its position and velocity are fixed, the wall particle effectively acts as a boundary condition.

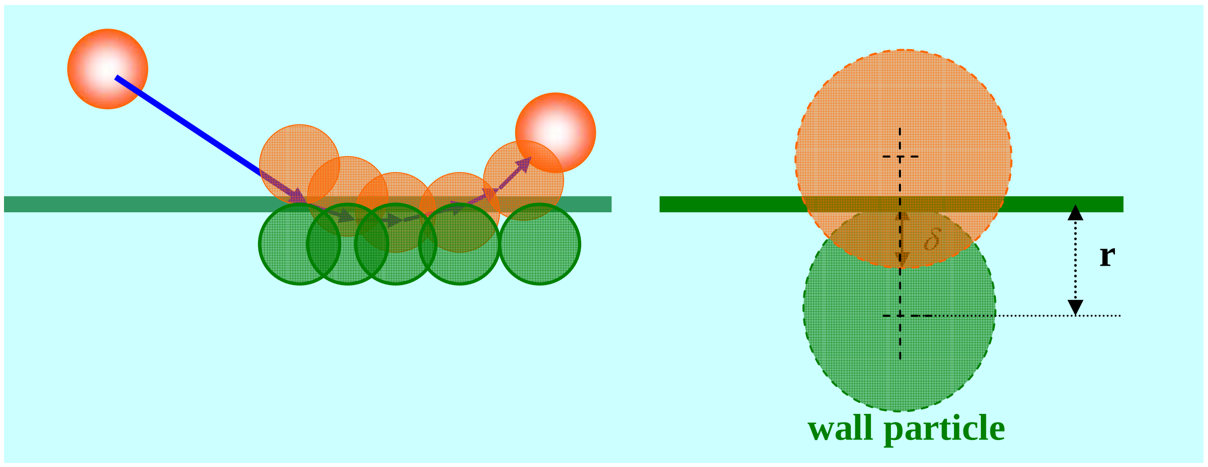

Figure 3. Particle-wall interaction; treating the wall as a particle of finite radius

Another situation of interest is periodic boundaries. Periodic boundaries are necessary when simulating large particle systems. Instead of simulating the whole system, only a repeating module in the system is identified and simulated. Periodicity is imposed by allowing the particles crossing over the periodic boundary to be introduced into the system at the opposite boundary. In this case the particles close to the periodic boundary are identified at each time step and particles with the same velocity and properties are declared on the opposite periodic boundary and outside the particle bed domain. This procedure enables the particles at a periodic boundary and inside the particle bed domain to see the particles at the other periodic boundary, thus behaving as they would in a large system.

Let a one-dimensional periodic domain of length L be considered. If a particle is close to $x=L$, then in a periodic case it should feel the influence of neighbor particles close to x=0. Hence at every time step the particles close to $x=0$ (i.e., say within one or two diameters distance from $x=0$) are identified and positioned at $x=L$ such that a particle of position x_i at distance i and within a diameter distance from $x=0$ is reproduced as x_{i+L} such that it is at a distance i from $x=L$. That way all particles between $x=0$ and $x=L$, which are close to $x=L$ will see their complete neighbor list for force calculations. If a particle $x_{L+\delta}$ crosses $x=L$ then it is introduced into the domain at $x=0$ by changing its position to x_{δ}. A similar process is performed for particles near the $x=0$ region.

2.3 Determination of Key Parameters

An important issue in the soft sphere model is the determination of key parameters such as the spring stiffness constant *k* and the damping coefficient η. Numerical parameters such as the time step Δt are determined from these values.

The spring stiffness *k* is deduced from Hertzian contact theory. According to the Hertzian contact theory (Tsuji *et al.*, 1992), the relation between the normal force F_n and normal displacement δ_n is given by

$$F_n = \delta_n^{3/2} \quad \text{...(13)}$$

In the case of two spheres of the same radius *r*,

$$k_n = \frac{\sqrt{2r}E_s}{3\left(1-\sigma_s^2\right)} \quad \text{...(14)}$$

where E_s is Young's modulus for the solid and σ_s is the Poisson's ratio. In the case of the particle-wall contact the normal stiffness constant is given by:

$$k_n = \frac{\frac{4\sqrt{r}}{3}}{\frac{1-\sigma_s^2}{E_s}+\frac{1-\sigma_w^2}{E_w}} \quad \text{...(15)}$$

where E_w is Young's modulus and σ_w the Poisson's ratio for the wall. The tangential force and the tangential displacement are related as

$$F_t = \frac{2\sqrt{2r}G_s}{2-\sigma_s}\delta_n^{1/2}\delta_t \quad \text{...(16)}$$

where G_s is the shear modulus and is related to E_s by

$$G_s = \frac{E_s}{2(1+\sigma_s)} \quad \text{...(17)}$$

Thus, for contact between two particles, k_t is given by

$$k_t = \frac{2\sqrt{2r}G_s}{2-\sigma_s}\delta_{nij}^{1/2} \quad \text{...(18)}$$

and for contact between a particle and wall

$$k_t = \frac{8\sqrt{r}G_s}{2-\sigma_s}\delta_{nij}^{1/2} \quad \text{...(19)}$$

When physical properties such as Young's modulus and Poisson's ratio are known, Cundall and Strack (1979) proposed that the damping coefficient η be calculated using $\eta = 2\sqrt{mk}$. Tsuji *et al.* (1992) showed that the damping coefficient for a nonlinear spring is related to the coefficient of restitution as $\eta = \alpha\sqrt{mk}\delta^{1/4}$ where α is a constant related to the coefficient of restitution *e*, which can be derived as follows. The coefficient of restitution *e* is defined as the ratio of particle velocities before and after the collision. For a system of particles *e* is a fixed constant

$$e = -\frac{v}{v_0} \quad \text{..(20)}$$

The equation of motion for a spring dashpot model is $m\ddot{x} + \eta\dot{x} + kx = 0$

Under given initial conditions of $x = 0; \dot{x} = v_i$ at *t*=0 is given by

$$x = \frac{v_i}{q}\sin(qt)\exp(-\gamma\omega_o t) \quad \text{..(21)}$$

$$\dot{x} = \frac{v_i}{q}\exp(-\gamma\omega_o t)\left\{q\cos(qt) - \gamma\omega_o\sin(qt)\right\} \quad \text{.....................................(22)}$$

where

$$\omega_o = \sqrt{k/m} \quad \text{...(23)}$$

$$\gamma = \frac{\eta}{2\sqrt{mk}}$$

$$q = \omega_o \sqrt{1-\gamma^2} \quad \text{...(24)}$$

The oscillation period is $2\pi/q$. A particle colliding with another particle at time t=0 detaches itself at time t=π/q. The velocity when it detaches is

$$v_o = -v_i \exp(\gamma\omega_o \pi/q) \quad \text{..(25)}$$

Therefore

$$e = -\frac{v_o}{v_i} = \exp(-\gamma\omega_o \pi/q) \quad \text{...(26)}$$

Hence for a fixed value of coefficient of restitution (Tsuji et al, 1993)

$$\eta = 2\gamma\sqrt{mk}$$

$$\gamma = \frac{\alpha}{1+\alpha^2} \quad \text{..(27)}$$

$$\alpha = \frac{1}{\pi}\ln\left(\frac{1}{e}\right)$$

These parameters are important in determining the time step *Δt* for numerical integration. Because the contact forces can only be determined after determining which particles are in contact, explicit time stepping methods are required. Explicit schemes impose a limit on the time step due to stability considerations. Smaller time steps ensure greater stability but one should choose a suitable time step so that the computational time is minimized. Cundall and Strack (1979) suggested that a time step be chosen such that

$$\Delta t < 2\sqrt{\frac{m}{k}} \quad \text{..(28)}$$

However Tsuji *et al.* (1992) report that the above criterion is not sufficient to ensure stability and proposed that the time step be chosen depending on the system frequency. They chose a spring mass equal to the mass of the particle and adjusted spring stiffness constant. The value of the time step is kept smaller than the frequency of the system. Another way is to choose the time step such that the highest frequency of the system could be integrated accurately (Wassgren, 1996). The smaller of the translational and rotational periods of oscillation is chosen. For a 2D problem with mono-sized particles, the maximum number of non-overlapping neighbors any particle can have is six. The translational period of a particle with six neighbors is

$$\tau_{trans} = 2\pi\sqrt{\frac{m}{6k}} \quad \text{...(29)}$$

and the rotational period is given by

$$\tau_{rot} = 2\pi\sqrt{\frac{I}{6kr^2}} \quad \text{...(30)}$$

It can be shown that $\tau_{rot} < \tau_{trans}$ and as a general rule the time step is chosen to be one-tenth of τ_{rot}. These time step criteria currently implemented in MFIX-DEM are very limited as collisional velocities or other constraints which play into the translation *etc.* are not considered. Additional time-step limitations need to be considered so that one can resolve the different processes as needed and one possibility is to use the time-step based on collisional velocity (Wada *et al.*, 2006).

An explicit (first order Euler scheme) scheme is used for solving the model equations given above and this is usually sufficient as the time steps are small enough to provide stable and accurate solutions. However, higher order or implicit time-stepping schemes are not studied systematically for their efficacy and accuracy. The new values of translational and rotational velocities and the new positions of particles are calculated from the old values explicitly.

$$\begin{aligned}
\vec{v}_{new} &= \vec{v}_{old} + \ddot{\vec{r}}_{old}\,\Delta t \\
\vec{r}_{new} &= \vec{r}_{old} + \vec{v}_{new}\Delta t \qquad \text{...(31)} \\
\vec{\omega}_{new} &= \vec{\omega}_{old} + \dot{\vec{\omega}}_{old}\,\Delta t
\end{aligned}$$

It is important to note that at every time step the neighbor list for each particle *i* needs to be obtained in order to compute the contact forces. This is the most time consuming and hence computationally expensive step in DEM. Faster search algorithms are key to reducing the overall computational time. Currently MFIX-DEM has two search algorithms, which are described in the next section.

3. Search Algorithms

3.1 N^2 Search:

This is the most easy search algorithm to implement. In this algorithm, the neighbors are found by calculating the distance between the particle of interest and every other particle in the system. Such a search has an operation count of $O(N^2)$, where N is the number of particles, and is extremely expensive for large-scale simulations. Some useful alternate search techniques have been explored, of which the most significant are presented here.

3.2 Quadtree and Octree Search

The concepts of quadtrees (2D) and their 3D counterparts, octrees, is not new in the field of geometric modeling (Mortensen, 1985). The concept of quadtrees (octrees) is a hierarchical data structure that recursively subdivides a square area (cubic volume) into four (eight) smaller squares (cubes) called quadrants (octants) until a stopping criterion is met. This decomposition process is often presented as a tree of out-degree four (eight). The idea is to divide the plane more efficiently into regions of a maximum desired resolution. In short, quadtrees and octrees are hierarchical variants of efficient spatial occupancy enumeration.

Typically, to store data, the domain is meshed into a number of boxes of equal size and the data in each such box is stored. As opposed to such standard methods of dividing the plane and distributing the data using the grid, which results in unnecessary storage of redundant data, quadtrees are a more efficient method for storing as well as processing data. Figure 4 is a pictorial comparison of a standard mesh and a quadtree. Quadtrees have been used extensively for two and three dimensional grid generators, though their role there is only to define the objects to be meshed. Presently quadtrees are being used not only for defining the objects to be meshed, but also to provide an $O(N\log N)$ search algorithm for arbitrary point distributions.

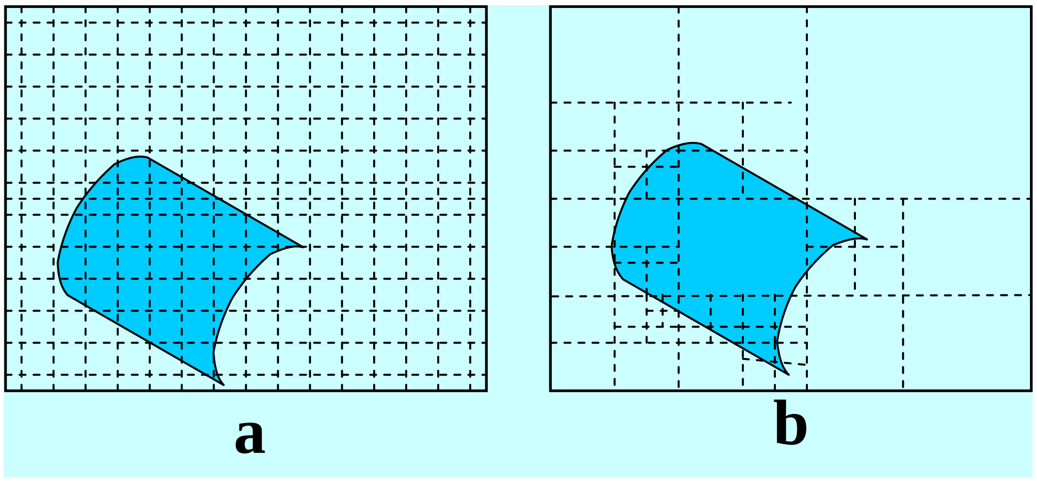

Figure 4. (a) Standard mesh versus (b) Quadtree

A good description of the octree search procedures is given in Lohner (1988). The following description is for a quadtree; the description of octree is very similar. In this type of search, particles are first arranged in quads, which are then searched for neighbors. In the routine mfix/model/des/quadtree(octree).f an array LQUAD(1:7, MQUAD) is defined to store the particle label based on the spatial location. Here MQUAD denotes the maximum number of quads allowed (4). For each quad with the label IQ, we store in LQUAD(1:7, IQ) the following information.

LQUAD(7,IQ): < 0 (e.g. -1) - the quad is full

= 0 - the quad is empty

= 1 to 4 – the number of particles in the quad

LQUAD(6,IQ): the label of the parent quad the present quad came from

LQUAD(5,IQ) : > the relative position (1-4) of the present quad in its parent quad

LQUAD(1:4, IQ) :

for LQUAD(7,IQ)>0 – the labels of the particles stored in the quad

for LQUAD(7,IQ)<0 – the labels of the quads into which the present quad is subdivided

In each quad a maximum of 4 particles are stored. The quad in which the particles lie is known as their parent quad. If a fifth particle falls into a quad, then the quad is divided into four sub-quads (known as the children quads) and the old particles are relocated into their respective new children quads. Then the fifth particle is introduced into the new quad it falls in. If the quad is full again the division of the parent quad into 4 child quads is repeated until a vacant storage space is found. The process is performed such that no quad has more than 4 particles. The next step after allotting all the particles into their respective quads is to search for the neighbors of a given particle. In order to find the neighbors for a given particle a search region is created around the particle such that the particles falling in the search region contain all the neighbors of the particle of interest. In order to find the particles that fall in the search region, the levels of the quadtree are traversed, eliminating at the highest possible level all quads that lie outside the search region. Once the quads that overlap with the search region are identified the particles in those quads are searched for the neighbors of the particle of interest (See Figure 5.). Lohner (1988) shows the schematic of the process. It is not difficult to see that a quadtree is an O(Nlog$_4$N) search algorithm.

Analogously an octree is an O(Nlog$_8$N) search algorithm [Mortensen, 1997], where N is the number of particles. For octree, the array structure would be

LQUAD(1:11, MQUAD) to store these particles, parent quad and other information

LQUAD(11, IQ): < 0 (e.g. -1): the octant is full

= 0: the octant is empty

= 1-8: the number of particles stored in the octant

LQUAD(10, IQ): the label of the octant the present octant came from

LQUAD(9, IQ): the relative position (1-8) in the present octant in its parent octant

LQUAD(1:8, IQ):

for LQUAD(11, IQ)> 0: the labels of the particles stored in this octant

for LQUAD(11,IQ)<0 – the label of the octants into which the present octant is subdivided

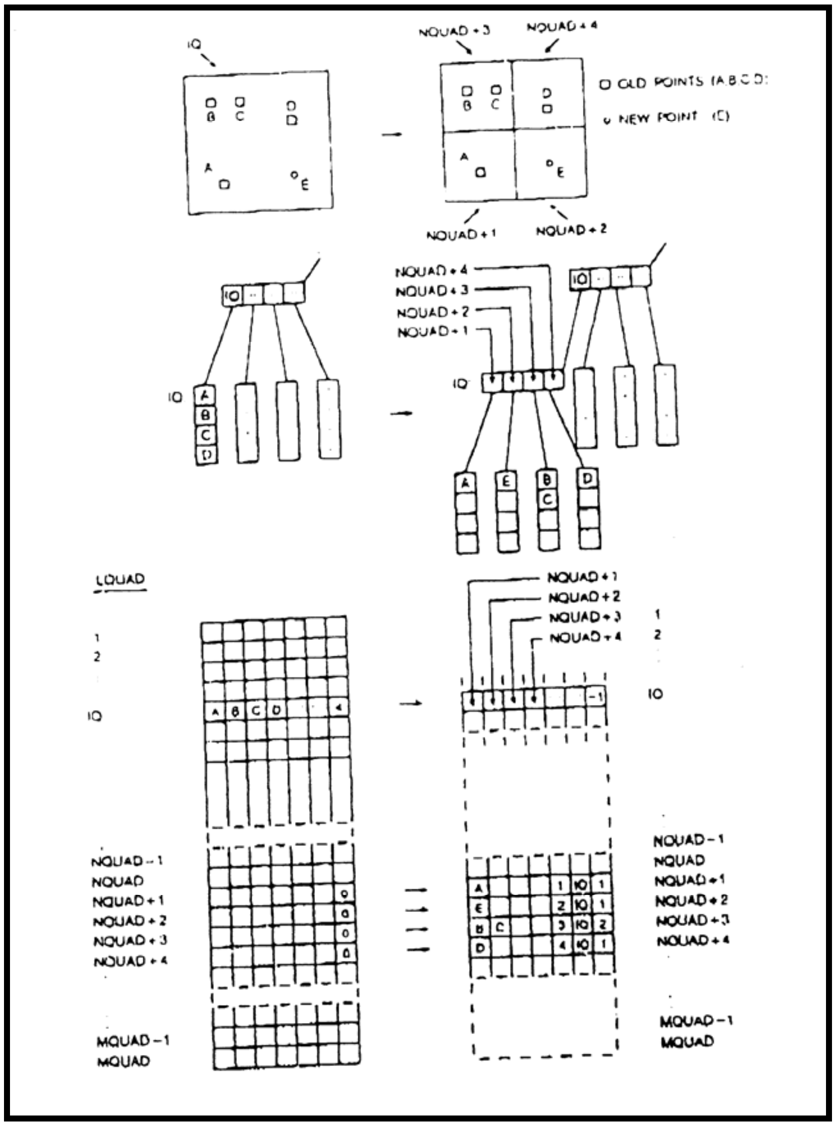

Figure 5. Schematic representation of the Quadtree algorithm (from Lohner, 1998)

3.3 Neighbor Search Options and Improvements in MFIX-DEM

In the previous sub-sections the 2 neighbor search mechanisms currently available *viz.* n-square search and the quadtree/octree search have been explained. Further options that have been provided in MFIX for the convenience the user as well as some implementation improvements are listed below:

- Choice to perform neighbor search at user-defined number (input through mfix.dat file) of solid time steps and not at every solid time step. This is particularly advantageous as it has been seen that neighbor search is one of the most expensive steps in DEM. Furthermore in dense beds the probability of neighbor list changing within a small number of solids time steps is small.

- Another user defined parameter (input through mfix.dat file), which can be used to perform neighbor search using a more physics based approach, is the distance traveled by a particle before a neighbor search is performed. The idea is to construct a user-defined virtual sphere of influence around each particle at the beginning of DEM. Then, the neighbor list is assumed to be unchanged as long as the particle lies within this sphere. Only when the particle traverses out of this sphere a new neighbor search is performed and the sphere center coordinates are shifted to the new location.

- In quadtree/octree search an important step is to create new quads and rearrange the tree. According to the algorithm provided in section 3, this is done each time the quadtree routine is called, which could prove expensive since it may not be necessary to re-define the quads for particles existing in the same quads as in the previous time the quadtree routine was called. In the current implementation this drawback is addressed. When the quadtree routine is called new quads are created only for those particles that have moved out of the quads they belong during the previous time. Using this reuse of particle information, the computational costs are drastically reduced. This involves moving the particles into new quads, which may already exist, or creating new quads only for the particles moved for which there are no quads to accommodate them. Also when a quad becomes empty, as a result of particles moving out, it is deleted and the parent and related quads are updated accordingly.

- In the above process while creating new quads for the particles which moved out of their current quads, it is not necessary to start at the outermost quad and traverse through the tree. Since the position of the particle at the previous instant is known (also the quad in which a particle existed is stored each time the quadtree search is performed), we traverse a few levels "up" the tree from the quad in which the particle existed and travel "down" to place it in a new quad. The number of levels to traverse "up" is supplied through a user-defined variable. If the algorithm cannot find a quad with the search performed then it starts from quad number 1 and places the particle in an appropriate quad.

- The procedure described above is also used while performing neighbor search. To perform neighbor search in a quad tree one typically starts at the outer most quad and travels "down" the tree to locate a particle in a quad and then search for its neighbors in the neighboring quads. The travel is eliminated as the quad in which the particle existed is stored. The neighbor search is started from only a few levels “up” (again the number of levels is user-defined). If neighbor search criteria are not met in these levels then the search is done starting from the outer most quad which is level 1 in the quadtree.

- Initializing the quad arrays, which is done when quadtree routine is called, consumes a lot of time and is an unnecessary expense. This issue is addressed by a user-defined (input through mfix.dat file) interval for initializing the quad. The reason the quadtree needs to be initialized is due to the fact that the tree can go out of balance as some branches can grow faster than the others when quads are added and removed to an existing tree when particles move.

- Also if the number of quads exceeds the maximum number of quads then the quadtree is rearranged by expunging the empty, deleted quads.

- Quadtree initialization is done at every fluid time step and thereafter at the user defined interval.

4. MFIX-DEM Coupling

Gas-particulate flows have a number of applications in industry including fluidized bed transport, pneumatic transport of pharmaceutical powders and pellets, as well as in pulverized coal transport. The study of the dynamics of such a flow aids in better design of gas-particulate systems. The analysis of gas-solids flows is complex because of the strong coupling between the solids and gas phases. The gas flows through the interstitial spaces or voids created by the particles, moving the particles and re-arranging the gas flow paths. The gas phase exerts a drag force on the solids; the solids in turn exert an equal and opposite force on the gas. Furthermore, the pressure gradients created in the gas flow give rise to pressure forces on the particulate phase. Density differences between the two phases cause buoyancy driven flows. Thus the two phases exchange mass, momentum and energy.

Coupling DES with computational fluid dynamics (CFD) offers a powerful way to understand gas-solids flow dynamics by accurately accounting for the interactions between the two phases. Each particle trajectory in the solids phase is computed using the discrete element model while the gas phase is treated as a continuum. The inter-particle collision forces are computed using DEM as explained in the previous sections. The present section deals with the calculation of drag and pressure forces on each particle and the implementation of DEM and gas-phase coupling. In the present formulation, we make no attempt to accurately resolve the flow around each particle because of the cost involved; the current formulation assumes that there are many particles in each fluid cell. Instead, empirical drag correlations available in the literature are used. These correlations compute the drag based on the void fraction and gas-solids relative velocities and other governing parameters. Though a number of techniques for coupling the two phases have been published in the literature (Tsuji 1993; Limtrakul *et al.*, 2003; Xu and Yu 1997; Li and Kuipers, 2003), not all ensure that interaction forces are equal and opposite between the phases. When mass and energy are exchanged, these must be reciprocal as well.

4.1 Governing Equations: Treatment of the Two Phases in MFIX

The gas phase calculations are done using a continuum approach. In the present work the software MFIX (Multiphase Flow with Interphase eXchanges) which is a general-purpose computer program developed at NETL by Syamlal *et al.* (Syamlal et al 1993; Syamlal, 1994; Syamlal, 1998) is used for the gas phase continuum calculations. MFIX uses the kinetic theory approach (Lun et al. 1984; Syamlal et al 1988; Ding et al 1990) to close the solids phase momentum equations. We first describe the governing equations in MFIX and then describe the changes due to the inclusion of DES. The governing continuity and momentum equations for the continuum phases and the procedures for their numerical solution are given in MFIX manuals. The current set of equations is given in Benyahia et al. (2006).

Gas Continuity Equation:

$$\frac{\partial}{\partial t}(\varepsilon_g \rho_g) + \nabla \cdot (\varepsilon_g \rho_g \vec{v}_g) = \sum_{n=1}^{N_g} R_{gn} \quad \text{..(32)}$$

Solids Continuity Equation (*m*th solids-phase):

$$\frac{\partial}{\partial t}(\varepsilon_{sm} \rho_{sm}) + \nabla \cdot (\varepsilon_{sm} \rho_{sm} \vec{v}_{sm}) = \sum_{n=1}^{N_{sm}} R_{smn} \quad \text{.......................................(33)}$$

The first term on the left in equation (32) and (33) accounts for the rate of mass accumulation per unit volume, and the second term is the net rate of convective mass flux. The term on the right accounts for interphase mass transfer because of chemical reactions or physical processes such as evaporation.

Gas Momentum Equation:

$$\frac{\partial}{\partial t}(\varepsilon_g \rho_g \vec{v}_g) + \nabla \cdot (\varepsilon_g \rho_g \vec{v}_g \vec{v}_g) = \nabla \cdot \overline{\overline{S}}_g + \varepsilon_g \rho_g \vec{g} - \sum_{m=1}^{M} \vec{I}_{gm} + \vec{f}_g \text{ ....(34)}$$

Here $\overline{\overline{S}}_g$ is the gas-phase stress tensor, $\vec{I}_{gm}$ is an interaction force representing the momentum transfer between the gas phase and mth solids phase, and $\vec{f}_g$ is the flow resistance offered by internal porous surface.

Solids Momentum Equation (mth solids-phase):

$$\frac{\partial}{\partial t}(\varepsilon_{sm} \rho_{sm} \vec{v}_{sm}) + \nabla \cdot (\varepsilon_{sm} \rho_{sm} \vec{v}_{sm} \vec{v}_{sm}) = \nabla \cdot \overline{\overline{S}}_{sm} + \varepsilon_{smg} \rho_{sm} \vec{g} + \vec{I}_{gm} + \sum_{\substack{l=0 \\ l \neq m}}^{M} \vec{I}_{ml} \text{ ....(35)}$$

Here $\overline{\overline{S}}_{sm}$ is the stress tensor for *mth* solids phase. The term $\vec{I}_{ml}$ is the interaction force between the *mth* and *lth* solids phases. The first term on the left represents the net rate of momentum increase and the second the usual convective term. The first term on the right represents forces due to the stress tensor, while the second term represents body forces (gravity in this case). The last two terms in equation (35) represent the momentum exchange between the fluid and solids phases and between the different solids phase, from left to right.

The interaction forces considered are the buoyancy and drag forces and momentum transfer due to mass transfer, since those are the most significant forces for fluidized bed applications. Thus, the fluid-solids interaction force is written as

$$\vec{I}_{gm} = -\varepsilon_g \nabla P_g - F_{gm}(\vec{v}_{sm} - \vec{v}_g) - R_{0m}[\xi_{0m} \vec{v}_{sm} + \bar{\xi}_{0m} \vec{v}_g] \text{ ....(36)}$$

where the first term on the right side describes the buoyancy force, the second term describes the drag force, and the third term describes the momentum transfer due to mass transfer. R_{0m} is the mass transfer from the gas phase to solids phase-m where

$$\xi_{0m} = \begin{cases} 1 & for\ R_{0m} < 0 \\ 0 & for\ R_{0m} \geq 0 \end{cases} \text{ ....(37)}$$

and $\bar{\xi}_{0m} = 1 - \xi_{0m}$

The solids-solids momentum transfer $\vec{I}_{ml}$, is represented as

$$\vec{I}_{ml} = -F_{sml}(\vec{v}_{sl} - \vec{v}_{sm}) + R_{ml}[\xi_{ml}\vec{v}_{sl} + \bar{\xi}_{ml}\vec{v}_{sm}] \quad \text{......................................(38)}$$

where R_{ml} is the mass transfer from the solids phase-m to solids phase-l,

$$\xi_{ml} = \begin{cases} 1 & for \;\; R_{ml} < 0 \\ 0 & for \;\; R_{ml} \geq 0 \end{cases} \quad \text{..(39)}$$

and $\bar{\xi}_{ml} = 1 - \xi_{ml}$

The gas-solids drag correlation for F_{sml} is derived from correlations relating the terminal velocity in fluidized or settling beds with void fraction and Reynolds number. Details may be found in (Syamlal *et al.* 1993).

The solids stress tensor is required to close the above equation set. The kinetic theory relations used in MFIX for computing the solids stress, including expressions for the solids pressure and the granular temperature, may be found in (Syamlal *et al.* 1993). The solids stress tensor calculations are not required in the present work as the kinetic theory for solid phase calculations is replaced by DEM. In the present work Eqns. (33) and (35) are also eliminated in favor of a discrete element formulation. Thus, the solids volume fraction and solids velocity are directly computed by using DEM to find particle velocities and positions, and locating the particles in the computational cells for the gas. The procedures for doing this are described below.

4.1.1 Interphase Drag Force Calculation

Though a variety of interaction forces couple the gas and solids (Basset force, Magnus lift force, drag force etc.) the focus of the present work is on the simulation of gas-particle flows for fluidized bed applications. Two primary interaction forces that are dominant for this application, interphase drag, and pressure force/buoyancy, are considered. The calculation domain is discretized into cells suitable for the fluid calculation. Typically, a fluid cell must contain many particles so as to be consistent with the volume averaging concept used in the gas phase. Since the gas-phase mesh is much larger than the individual particle, it is not possible to resolve the drag numerically. The interphase drag is computed from experimental correlations for fluidized beds.

The drag force is expressed in the form:

$$F_D = F_{gs}(V_s - V_g) \quad \text{...(40)}$$

as a product of a gas-solids drag coefficient and the interphase velocity difference, where V_s is the solids velocity and V_g is the gas velocity. It is worth mentioning here that velocities are face centered values and the drag coefficient F_{gs}, which depends on the cell centered values of void fraction, is a cell centered value. While calculating the overall drag on the solids in a cell and distributing it onto individual particles we are interested in a cell centered value and hence we interpolate the velocities to obtain a cell centered solids velocity V_s to be used in drag calculation. The drag coefficient F_{gs} can be calculated using various correlations and the formulations available for the two-fluid MFIX model can be employed when DEM is invoked for particle phase.

Typically, several DEM time steps are traversed during a single fluid time step. After the completion of a DEM time step, the solid particles are located on the fluid background mesh. The void fraction and the averaged solids velocities in each fluid cell are computed. This information is used to calculate the gas-solids drag force in each cell using any of the above mentioned drag correlations. The net drag on the solids phase in each cell is equal to the net drag on the gas phase in the cell. The net solids drag is distributed to each particle in the cell based on the ratio of the particle volume to total solids volume in the cell. If V_{ijm} is the volume of a particle *i* of solid phase *m* in the cell *j*, and F_{Dj} is the total drag force on the solid mass of volume V_s in the cell, the drag F_{Dijm} on the particle is given by

$$F_{Dij} = F_{Dj}\left(\frac{\sum V_{pijm}}{V_s}\right)\left(\frac{V_{pijm}}{\sum V_{pijm}}\right) \quad \text{..(41)}$$

For a single solids phase and same sized particles,

$$F_{Dij} = F_{Dj}\left(\frac{1}{N}\right) \quad \text{..(42)}$$

where, N is the number of particles in the cell. Other algorithms can be used where the drag force can be based on the relative velocities between the particles and gas or other variations and combinations.

4.1.2 Void Fraction Calculation

To calculate the drag force on the solids mass in each cell, the value of the void fraction in the cell must be known. The domain is meshed into computational cells for the fluid continuum calculations. In the flow at every time step particles enter and leave a cell. Each particle from the DEM simulation must be located within the fluid control volume; the sum of all particle volumes in the cell yields the solids volume fraction. It is useful to find the cell values of solids volume fraction and velocity classified by particle size. This requires the computation of several solids volume fractions per cell.

If V_{pimj} is the volume of a particle of index *i* of a size representing phase *m* in a fluid cell *j*, then the total solids volume V_{smj} of the phase *m* in the cell *j* is

$$V_{smj} = \sum_i V_{pimj} \quad \text{...(43)}$$

The corresponding volume fraction is then given by

$$\varepsilon_{smj} = \frac{V_{smj}}{V_j} \quad \text{...(44)}$$

where, the cell volume $V_j = \Delta x \Delta y \Delta z$ and Δx is the length of the cell in x-direction, Δy is the length of the cell in y-direction and Δz is the depth of the cell in Z-direction. For 2-dimensional simulations, Δz is taken as one particle diameter in Cartesian coordinates and as one radian in the cylindrical coordinates.

The total solids volume fraction in the cell is given by the sum of the solids volume fractions of each phase in the cell

$$\varepsilon_{sj} = \sum_m \varepsilon_{smj} \quad \text{...(45)}$$

Since the sum of the various solids volume fractions and the fluid volume fraction must be equal to unity, the j^{th} cell void fraction ε_j can then be obtained from the solids volume fraction as:

$$\varepsilon_j = 1 - \varepsilon_{sj} \quad \text{...(46)}$$

Once the volume fraction is made available to MFIX, blockage effects are automatically taken into account.

4.1.2.1 Particle in Fluid Cells to Compute Void Fraction

To calculate the void fraction in each computational cell, it is necessary to identify all the particles in that cell to obtain the solids volume fraction in the cell and hence the void fraction. This is usually done by looping over all particles and comparing the particle center coordinates with all the fluid cell boundaries and locating each particle in a cell. This is an inefficient approach and has been modified in the present version. This looping over each particle is done only at the first time step. Thereafter for each particle the direction of motion is obtained by its velocity and only the few neighbor cells in that direction are checked for its new location. This eliminates the looping of each particle over all the computational cells. Once all the particles in a cell are located, the volume fraction of each solids phase (based on the particle radius or density) is obtained and fed to MFIX functions to calculate the cell void fraction ε_g and hence ρ_{pg} .

4.1.3 Solids Velocity Calculation

Another variable required in the drag force calculation is the solids velocity of each particle size group in each fluid cell. DEM gives the velocity of each individual particle at the end of every solids time step. The particle velocities must be averaged in each cell to obtain a representative cell solids velocity. In the present work, the volume- averaged solids velocity is computed and stored at the cell centers.

If U_{pimj} is the velocity of particle *i* of solid phase *m* in cell *j*, then the volume averaged solid velocity U_{smj} of phase *m* in the cell is given by

$$U_{smj} = \frac{\sum_i V_{pimj} U_{pimj}}{V_{smj}} \quad \text{...(58)}$$

This value is used in the calculation of the drag force. It should be noted that the value thus obtained is a cell-center value where as the numerical scheme follows the convention of velocities stored at the face center. To obtain the face center value the corresponding neighbor cell center values are weighted averaged across the face.

4.2 Pressure Force Calculation

The other important force that is exerted by the gas on the particles is the pressure force/buoyancy. At every fluid time step the gas side continuum equations are solved and the gas side quantities like gas velocity and pressure are obtained. Pressure is stored at the cell center. Thus the pressure exerted by the gas in each cell is obtained and stored at its center. Correspondingly, a pressure force is exerted by the gas on all the particles in that cell. The pressure force on the cell in a given direction is the difference of the pressure on the faces of the cell normal to the direction, times the average cross sectional area of the cell normal to the direction. Consider a cell in a one dimensional uniform mesh. The pressure force on the cell in one dimension (in y-direction) is analyzed. Let the cell dimensions be Δx , Δy , Δz in the x, y and z directions respectively. The pressure in the cell is P_p . The pressures in the north and south cells are P_n and P_s respectively. Then the pressure on the north face would be $P_1 = \frac{P_n + P_p}{2}$ and the pressure on the south face would be $P_2 = \frac{P_s + P_p}{2}$. The y-direction pressure force on the cell would be $P_c = \left(P_2 - P_1\right)\Delta x \Delta z$ (Δz for a Cartesian analysis is the diameter of the particle). It should be noted here that the pressure averaging across the cells assumes uniform Cartesian grid. However, for non-uniform meshes weighted averaging should be done based on the cell dimensions. Further more, this pressure force should be distributed to all the particles in the cell based on the location to mimic the physical picture. For a particle *j* of solid phase *m* in the cell, the pressure force P_{jm} is given by

$$P_{jm} = \varepsilon_m P_c \frac{V_j}{V_m} \quad \text{..(59)}$$

where $\varepsilon_m = \frac{V_m}{\Delta x \Delta y \Delta z}$ is the solids volume fraction of the phase, V_j is the volume of the particle and V_m is the total volume of the solids phase in the cell.

It can be seen that for a static fluid, if the north and south cells considered were at heights h_1 and h_2 , $h_2 - h_1 = \Delta y$, then P_{jm} would be

$$P_{jm} = \varepsilon_m P_c \frac{V_j}{V_m} = \frac{V_m}{\Delta x \Delta y \Delta z}\left(P_2 - P_1\right)\Delta x \Delta z \frac{V_j}{V_m}$$

$$P_{jm} = \frac{V_m}{\Delta x \Delta y \Delta z}\rho g\left(h_2 - h_1\right)\Delta x \Delta z \frac{V_j}{V_m} = \frac{V_m}{\Delta x \Delta y \Delta z}\rho g \Delta x \Delta y \Delta z \frac{V_j}{V_m} \qquad \text{.....................(60)}$$

$$P_{jm} = \rho g V_j$$

Here, ρ is the gas density. Combined with the $-\rho_s g$ term in the particle equation of motion, the buoyancy force on the particle is obtained.

4.3 Coupling Algorithm for Gas-Solid Flows

4.3.1 Time Step

The intent of the present work is to use a sequential and iterative procedure for coupling the two calculations. Due to the limitations in the soft sphere model the solids time steps typically are very small. This is because of the limitations imposed by the particle properties like high stiffness constants. We wish to use far larger time-steps for the gas than for the solids, since the gas has no explicit time-stepping limitations. At every gas-phase time-step, several DES sub-steps are taken. The volume fraction and averaged solids velocity are computed at the end of the sub-steps. The gas flow is assumed stationary at the old values during this sub-stepping. At the end of these sub-steps the void fraction and the averaged solids velocities are fed to the gas side and then the gas flow is iterated to convergence within the next time step. The procedure is repeated until the desired time interval is covered.

4.3.2 Drag Coupling to the gas side

The interphase drag coupling and the drag calculation on the solids due to the gas was explained in the previous sections (4.1.1, 4.1.2 and 4.1.3). The gas phase is also exposed to an equal and opposite amount of drag. This is done by adding the drag terms to the A_m and B_m matrices. From the expression for drag

$$F_D = F_{gs}(V_s - V_g) \qquad \text{..(61)}$$

the $F_{gs}V_s$ would enter the source matrix B_m and $F_{gs}V_g$ is added to the diagonal elements of A_m matrix, while correctly accounting for the sign conventions. It should be noted that velocities are face centered values and the drag coefficient F_{gs}, which depends on the cell centered values of void fraction, is a cell centered value. Hence the face averaged drag coefficients need to be used for consistent treatment of the face centered and cell-centered values.

5. Miscellaneous Improvements/Modifications in the Present MFIX-DEM Version

- More efficient looping is implemented by making sure that the longest loop is the innermost. One place further gains can be obtained is to modify calc_force_des.f and the subroutines called by this routine to perform looping at subroutine level.

- Many variables are renamed for easy understanding and consistency.

- Duplication of variables is eliminated. For example in the current implementation the gravity vector and domain boundaries are read from the MFIX inputs in mfix.dat, unlike in the previous version where DEM domain and gravity had to be separately specified.

- All arrays are dynamic memory allocated based on the input value of the number of particles.

- 2D and 3D capability

5.1 DEM Output

In the current version DEM output has been made Paraview compatible. The output files are dumped at the same frequency as the MFIX SPX files (currently hardwired to link to void fraction frequency, SPX1). The paraview with MFIX reader can be used to view the MFIX Continuum field and the DEM particle information simultaneously. When MFIX and DEM are uncoupled the output files are generated at a user-defined frequency (input along with other DEM inputs in the mfix.dat file). For more information about this capability, please follow this link: http://www.mfix.org/members/develop/paraview/des/des.php.

5.2 Restart Capability

A hitherto non-existing and very useful capability in MFIX-DEM is the capability to restart a DEM run. All the necessary information like the particle positions, particle velocities, particle neighbor list, inter-particle overlaps and inter-particle forces are stored in arrays at the same frequency MFIX restart files are written. These arrays are over written each time and hence the restart storage does not grow. When restart is performed the information is read from these arrays and the run is continued.

5.3 DEM particle field generation utility

A utility has been added to the mfix/tools/DES_Particle_Generator to generate particle input deck for the DEM cases. This directory has a FORTRAN source file (DESParticleGen.f) and a sample input file for this utility (Pgen.in). One needs to compile the source file (e.g. ifort -FR -o DESParticleGen DESParticleGen.f) and run the resulting executable. The Pgen.in is commented well and one can modify that to achieve the desired input.

5.4 DEM User-Input

VARIABLE	TYPE	DESCRIPTION
DISCRETE_ELEMENT [F]	L	Variable to decide to do DEM for solids or not. Must be TRUE to do DEM.
DES_CONTINUUM_COUPLED [F]	L	Decided whether gas and solids are coupled together in flow.
WALLDTSPLIT [F]	L	Treats wall interaction also as a two-particle interaction but accounting for the wall properties. Must be TRUE for DEM.
TSUJI_DRAG [F]	L	To use Tsuji's drag correlation
PARTICLES [UNDEFINED_I]	I	Number of particles
DTSOLID_FACTOR [UNDEFINED]	DP	DT_FLUID/DTSOLID; number of solid marches per fluid time step.
Boundary Conditions (Not needed for MFIX-DEM coupled simulations)		
DES_PERIODIC_WALLS [F]	L	Periodic wall boundary condition is imposed on any pair of walls.
DES_PERIODIC_WALLS_X [F]	L	Direction of periodicity: X
DES_PERIODIC_WALLS_Y [F]	L	Direction of periodicity: Y
DES_PERIODIC_WALLS_Z [F]	L	Direction of periodicity: Z
INLET_OUTLET [F]	L	If inlet-outlet boundary condition is imposed.
INLET_OUTLET_X [F]	L	Direction of Inlet-Outlet: X
INLET_OUTLET_Y [F]	L	Direction of Inlet-Outlet: Y
INLET_OUTLET_Z [F]	L	Direction of Inlet-Outlet: Z
Neighbor Search Parameters		

DES_NEIGHBOR_SEARCH [1]	I	1= N-Square search; 2 = Quadtree/Octree search.
NEIGHBOR_SEARCH_N [1]	I	Number of time solid steps when neighbor search is performed
MN [10]	I	Maximum number of neighbors
QLM [1]	I	Number of levels to traverse “up” to move a particle to its new quad
QLN [1]	I	Number of levels to traverse “up” to perform particle neighbor search
INIT_QUAD_COUNT [UNDEFINED_I]	I	Count to initialize quadtree
NEIGHBOR_SEARCH_RAD_RATIO [1000]	DP	Ratio of the particle radius and the distance (imaginary sphere radius) to be traveled by a particle before a neighbor search is performed
MQUAD_FACTOR [1.1]	DP	Factor to create quadtree arrays based on the number of particles
Particle-particle and Particle-wall contact parameters		
KN [UNDEFINED]	DP	Normal spring constant for inter-particle collision
KT [UNDEFINED]	DP	Tangential spring constant for inter-particle collision
KN_W [UNDEFINED]	DP	Normal spring constant for particle-wall collision
KT_W [UNDEFINED]	DP	Tangential spring constant for particle-wall collision
ETA_DES_N [UNDEFINED]	DP	Normal damping coefficient for inter-particle collision
ETA_DES_T [UNDEFINED]	DP	Tangential damping coefficient for inter-particle collision
ETA_N_W [UNDEFINED]	DP	Normal damping coefficient for particle-wall collision
ETA_T_W [UNDEFINED]	DP	Tangential damping coefficient for particle-wall collision
MEW [UNDEFINED]	DP	Particle friction coefficient
MEW_W [UNDEFINED]	DP	Wall friction coefficient
DES_F [UNDEFINED]	DP	Frequency of bottom wall oscillation
DES_GAMMA [UNDEFINED]	DP	Acceleration amplitude of the bottom wall
Output and Restart Control		
PRINT_DES_DATA [F]	L	Option to print DEM output
P_TIME [UNDEFINED_I]	DP	Time interval to print DEM output when doing only granular flow simulation

6. References:

1. Benyahia S. , M. Syamlal, T. J. O’Brien, 2006, “Summary of MFIX Equations 2005-4”, From URL http://www.mfix.org/documents/MfixEquations2005-4.pdf
2. Campbell C.S., 1982, Shear flows in granular material, Ph.D Thesis, California Institute of technology
3. Crowe C., Sommerfeld M., and Tsuji Y., 1998, Multiphase flows with droplets and particles, CRC Press, New York
4. Cundall P.A., and Strack O.D.L., 1979, “A discrete numerical model for granular assemblies” Geotechnique, Vol. 29, No. 1, 47-65
5. Ding, J. and D. Gidaspow, 1990, “A Bubbling Fluidization Model Using Kinetic Theory of Granular Flow,” AIChE J., Vol. 36:523-538
6. Limtrakul, S.; Chalermwattanatai, A.; Unggurawirote, K.; Tsuji, Y.; Kawaguchi, T.; Tanthapanichakoon, W., 2003, “Discrete particle simulation of solids motion in a gas-solid fluidized bed,” Chemical Engineering Science, Vol. 58, 915-921.
7. Li J. and Kuipers J. A. M., 2003, “Gas-particle interactions in dense gas-fluidized beds,” Chemical Engineering Science, Vol. 58, 711-718
8. Lohner R., 1988, “Some useful data structures for the generation of unstructures grids”, Communications in Applied Numerical Methods, Vol. 4, 123-135
9. Lun, C.K.K., Savage, S. B., Jeffery, D.J., and Chepuriny, N., 1984, “Kinetic theories for granular flow: inelastic particles in Couette flow and slightly inelastic particles in a general flowfield”, J. Fluid Mech., Vol. 140, 223-256
10. Mortensen M. E., 1985, “Geometric Modeling,” John Wiley & Sons, Inc., New York, NY
11. Syamlal, M., Roger, W., and O’Brien, T.J., 1993, “MFIX documentation: theory guide, Technical Note, DOE/METC-94/1004,” NTIS/DE94000087, National Technical Information Service, Springfield, VA. {Available for download at http://www.mfix.org}
12. Syamlal, M., 1994, “MFIX Documentation, User's Manual,” Technical Note, DOE/METC-95/1013, NTIS/DE95000031, National Technical Information Service, Springfield, VA. {Available for download at http://www.mfix.org}
13. Syamlal, M., MFIX documentation: numerical technique, technical note, DOE/MC31346-582495, NTIS/DE98002029, National Technical Information Service, Springfield, VA. 1998. {Available for download at http://www.mfix.org}
14. Tsuji Y., Tanaka T., and Ishida T., 1992, “Lagrangian simulation of plug flows of cohesionless particles in a horizontal pipe”, Powder Tech., Vol. 71, 239-250
15. Tsuji Y., Kawaguchi T., and Tanaka T., 1993, “Discrete particle simulation of two dimensional fluidized bed”, Powder Tech., Vol. 77, 79-87
16. Wassgren C.R., Brennen C.E., and Hunt M.L., 1996, “Vertical vibration of a deep bed of granular material in a container”, J. Applied Mechanics, Vol. 63, 712-719
17. Wada, K., H. Senshu, and T. Matsui, 2006, “Numerical simulation of impact cratering on granular material,” Icarus, Vol. 180, 528–545
18. Xu B.H., and Yu A.B., 1997, “Numerical simulation of the gas-solid in a fluidized bed by combining discrete particle method with computational fluid dynamics”, Chemical Engineering Science, Vol. 52, 2785-2809

Appendix A: List of files in /model/des and their purpose

NOTE: The equations solved in a subroutine is mentioned in the header of each file along with the reference paper

calc_force_des.f	Call all the subroutines needed to compute the inter-paticle collision force on each particle due to its neighbors.
cfassign.f	Assign the necessary values for DEM computation. For example, assigning DEM boundaries from the values entered for MFIX input in mfix.dat. Assigning DEM gravity vector from MFIX input. Calculating RADIUS_EQ. Calculating DTSOLID based on rotational and translational constraints.
cffctowall.f	Calculate the total force Fc and Tow on a particle in a particle-wall collision.
cffctow.f	Calculate the total force Fc and Tow on a particle in a particle-particle collision.
cffn.f	Calculate the normal force on a particle in inter-particle collision
cffnwall.f	Calculate the normal force on a particle in particle-wall collision
cfft.f	Calculate the tangential force on a particle in inter-particle collision
cfftwall.f	Calculate the tangential force on a particle in particle-wall collision
cfincrementaloverlaps.f	Calculate the incremental overlap between a particle pair that occurred in a given solid time step. The incremental overlap would be the product of the velocity (normal or tangential) and solid time step.
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 is the particle is found to have no neighbors (either particles or walls).
cfnormal.f	Calculate the normal vector for a contacting particle pair.
cfnormalwall.f	Calculate the normal vector for a contacting particle-wall pair.
cfperiodicwallneighborx.f	Calculate the neighbors of particles near the x boundaries in a periodic boundary condition.
cfperiodicwallneighbory.f	Calculate the neighbors of particles near the y boundaries in a periodic boundary condition.
cfperiodicwallneighborz.f	Calculate the neighbors of particles near the z boundaries in a periodic boundary condition.
cfperiodicwallx.f	Calculate the list of particles near the x

	boundaries in a periodic boundary condition.
cfperiodicwally.f	Calculate the list of particles near the y boundaries in a periodic boundary condition.
cfperiodicwallz.f	Calculate the list of particles near the z boundaries in a periodic boundary condition.
cfrelvel.f	Calculate the translational relative velocity for a contacting particle pair.
cfslide.f	Check for Coulomb's friction law 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.
cfslipvel.f	Calculate the slip velocity of the contact point between a colliding particle pair.
cftangent.f	Calculate the tangent vector for a colliding particle pair.
cftotaloverlaps.f	Calculate the total overlap between a colliding particle pair by finding the center to center distance and compare with the sum of their radii.
cftotaloverlapswall.f	Calculate the total overlap between a colliding particle and wall particle by finding the center to center distance and compare with the sum of their radii.
cfupdateold.f	Update the old values of particle position and velocity with the new values computed.
cfvrn.f	Calculate the normal component of the relative velocity to compute the normal overlap.
cfvrt.f	Calculate the tangential component of the relative velocity to compute the tangential overlap
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_calc_d.f	Calculate coefficients linking velocity correction to pressure correction when coupled
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_inlet_outlet.f	Implement the inlet-outlet boundary condition and also calculate the inter-particle collision force by calling all related subroutines.
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.
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.
periodic_wall_calc_force_des.f	Implement the periodic wall boundary condition by calling allthe periodic wall related subroutines and also the inter-particle collision force calculation routines.
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

APPENDIX 1

Fluid Pressure Correction for MFIX-DEM

Jin Sun
Iowa State University
Ames, IA

August, 2006

1 Formulation

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 documentation (1).

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

$$a_{0p}(u_0)_p = \sum_{nb} a_{0nb}(u_0)_{nb} + b_0 - A_p(\varepsilon)_p[(P_g)_E - (P_g)_W] + F_{10}[(u_1)_p - (u_0)_p]\Delta V. \tag{1}$$

Use the pressure field P_g^* and void fraction field ε_0^* from the previous iteration to calculate tentative values of the velocity fields:

$$a_{0p}(u_0^*)_p = \sum_{nb} a_{0nb}(u_0^*)_{nb} + b_0 - A_p(\varepsilon_0^*)_p[(P_g^*)_E - (P_g^*)_W] + F_{10}[(u_1^*)_p - (u_0^*)_p]\Delta V. \tag{2}$$

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[1]. The Equation 2 is thus re-written as

$$(a_{0p} + F_{10}\Delta V)(u_0^*)_p = \sum_{nb} a_{0nb}(u_0^*)_{nb} + b_0 - A_p(\varepsilon_0^*)_p[(P_g^*)_E - (P_g^*)_W] + F_{10}(u_1)_p\Delta V. \tag{3}$$

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

$$\begin{aligned}(P_g)_E &= (P_g^*)_E + (P_g')_E, \\ (P_g)_W &= (P_g^*)_W + (P_g')_W, \\ (u_0)_p &= (u_0^*)_p + (u_0')_p, \\ (u_0)_{nb} &= (u_0^*)_{nb} + (u_0')_{nb}.\end{aligned} \tag{4}$$

[1]The solid velocity could be treated in a fully implicit fashion by incorporating the particle velocity update. However, it will also invoke the multi-time-scale issue due to the two fluid and DEM coupling and may need special treatment. Thus this approach is not used here for simplicity

Substitute the corrections in Equation 4 into Equation 1 and subtract Equation 3 from the resulting equation to get

$$(a_{0p} + F_{10}\Delta V)(u_0')_p = \sum_{nb} a_{0nb}(u_0')_{nb} - A_p(\varepsilon_0^*)_p[(P_g')_E - (P_g')_W]. \tag{5}$$

Drop the momentum convection to get

$$(a_{0p} + F_{10}\Delta V)(u_0')_p = -A_p(\varepsilon_0^*)_p[(P_g')_E - (P_g')_W]. \tag{6}$$

Solve for u_0' to get

$$(u_0')_p = -\frac{A_p(\varepsilon_0^*)_p}{a_{0p} + F_{10}\Delta V}[(P_g')_E - (P_g')_W], \tag{7}$$

which can be written as

$$(u_0')_p = -d_{op}[(P_g')_E - (P_g')_W], \tag{8}$$

where

$$d_{op} = \frac{A_p(\varepsilon_0^*)_p}{a_{0p} + F_{10}\Delta V}. \tag{9}$$

The velocity correction is given by

$$(u_0)_p = (u_0^*)_p - d_{0p}[(P_g')_E - (P_g')_W]. \tag{10}$$

The rest of derivation is the same as that in the MFIX documentation (1). The resulted pressure correction equation is also in the same form except that d_{0p} is given by Equation 9.

2 Performance

The central-jet fluidization tutorial case was tested using the new pressure correction formulation and compared to the results using the original formulation. The case was run for 20 second simulation time on an Opteron processor with MFIX compiled with the Intel Fortran compiler version 9.

The visual examination of the particle dynamics showed very similar dynamic behavior between the two formulations[2]. The pressure drop at 20 cm above the distributor as a function of time was shown in Figure 1, which indicates close agreement too.

The comparisons for convergence and computational time are shown in Table 1.

References

[1] M. Syamlal. MFIX documentation: Numerical technique. Technical Note DOE/MC31346-5824, NTIS/DE98002029, National Energy Technology Laboratory, Department of Energy, 1998. See also URL `http://www.mfix.org`.

[2] Please use the following link to view the animations: http://www.public.iastate.edu/ jinsun/results.htm

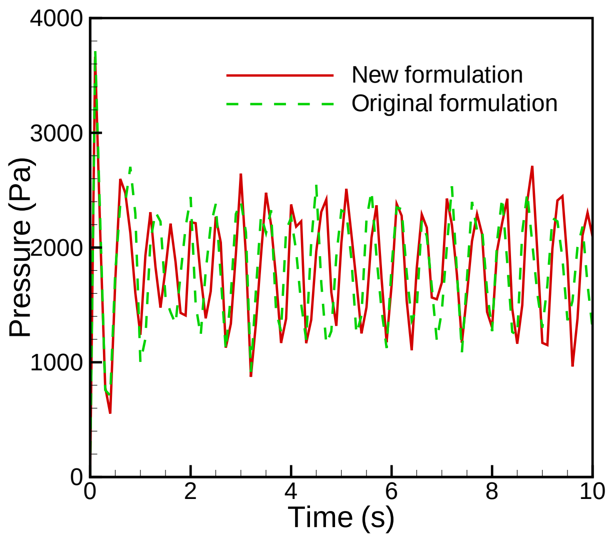

Figure 1: Pressure drop at 20 cm above the distributor as a function of time showing the comparison between new and original formulations

Table 1: Iteration per time step and total computational time using different formulations

Computational time	Iteration per time-step	ISIS sweep direction	RSRS sweep direction
Original formulation	≈ 70	15887 s	Divergence
New formulation	≈ 15	8185 s	6779 s