CFD – DEM Modeling and Parallel Implementation of Three Dimensional Non- Spherical Particulate Systems

Vivek Srinivasan

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Mechanical Engineering

Danesh K. Tafti, Chair
Rui Qiao
Mehrdad Shahnam

07.08.2019
Blacksburg, Virginia

Keywords: Computational Fluid Dynamics, Discrete Element Modeling, Non – Spherical Particles, Impulse Collision Response, Parallel Computing, Collision Detection, Event-driven Hard Sphere, Time-driven Soft Sphere

CFD – DEM Modeling and Parallel Implementation of Three Dimensional Non- Spherical Particulate Systems

Vivek Srinivasan

Abstract

Particulate systems in practical applications such as biomass combustion, blood cellular systems and granular particles in fluidized beds, have often been computationally represented using spherical surfaces, even though the majority of particles in archetypal fluid-solid systems are non-spherical. While spherical particles are more cost-effective to simulate, notable deficiencies of these implementations are their substantial inaccuracies in predicting the dynamics of particle mixtures. Alternatively, modeling dense fluid-particulate systems using non-spherical particles involves increased complexity, with computational cost manifesting as the biggest bottleneck. However, with recent advancements in computer hardware, simulations of three-dimensional particulate systems using irregular shaped particles have garnered significant interest.

In this research, a novel Discrete Element Method (DEM) model that incorporates geometry definition, collision detection, and post-collision kinematics has been developed to accurately simulate non-spherical particulate systems. Superellipsoids, which account for 80% of particles commonly found in nature, are used to represent non-spherical shapes. Collisions between these particles are processed using a distance function computation carried out with respect to their surfaces. An event - driven model and a time-driven model have been employed in the current framework to resolve collisions. The collision model's influence on non–spherical particle dynamics is verified by observing the conservation of momentum and total kinetic energy. Furthermore, the non-spherical DEM model is coupled with an in-house fluid flow solver

(GenIDLEST). The combined CFD-DEM model's results are validated by comparing to experimental measurements in a fluidized bed. The parallel scalability of the non-spherical DEM model is evaluated in terms of its efficiency and speedup. Major factors affecting wall clock time of simulations are analyzed and an estimate of the model's dependency on these factors is documented. The developed framework allows for a wide range of particle geometries to be simulated in GenIDLEST.

CFD – DEM Modeling and Parallel Implementation of Three Dimensional Non- Spherical Particulate Systems

Vivek Srinivasan

General Audience Abstract

CFD – DEM (Discrete Element Method) is a technique of coupling fluid flow solvers with granular solid particles. CFD – DEM simulations are beneficial in recreating pragmatic applications such as blood cellular flows, fluidized beds and pharmaceutics. Up until recently, particles in these flows have been modeled as spheres as the generation of particle geometry and collision detection algorithms are straightforward. However, in real – life occurrences, most particles are irregular in shape, and approximating them as spheres in computational works leads to a substantial loss of accuracy. On the other hand, non – spherical particles are more complex to generate. When these particles are in motion, they collide and exhibit complex trajectories. Majority of the wall clock time is spent in resolving collisions between these non – spherical particles. Hence, generic algorithms to detect and resolve collisions have to be incorporated.

This primary focus of this research work is to develop collision detection and resolution algorithms for non – spherical particles. Collisions are detected using inherent geometrical properties of the class of particles used. Two popular models (event-driven and time-driven) are implemented and utilized to update the trajectories of particles. These models are coupled with an in – house fluid solver (GenIDLEST) and the functioning of the DEM model is validated with experimental results from previous research works. Also, since the computational effort required is higher in the case of non – spherical particulate simulations, an estimate of the scalability of the problem and factors influencing time to simulations are presented.

Acknowledgments

I would like to thank my advisor, Dr. Danesh Tafti, for the support and guidance offered by him for successful completion of this research work. His expertise in this research topic was invaluable and his vision, in steering the research in the right direction when I most needed it, helped in making swift progress. I would also like to express my profound gratitude to my committee members, Dr. Rui Qiao and Dr. Mehrdad Shahnam, for their willingness to be on my committee and their interest in my research work.

I have been fortunate to be a part of a group of extremely talented individuals at the High Performance Computational Fluid Dynamics laboratory. I express my sincerest appreciations to Susheel, Husam, Peter, Hamid, Aevelina, Mohammad, Maryam, Ze, Xiaozhou, Steven, Tae and Vicky for helping me with my research. A special thanks to Susheel for his countless advises and suggestions on High Performance Computing and DEM formulations. I would also like to thank Husam and Peter for being instrumental in putting the mathematical framework together, and, Aevelina, Tae, Vicky and Ze for helping me out with challenges faced when working with GenIDLEST.

I am grateful to my friends at Virginia Tech who had made my stay as a graduate student memorable. I would also like to thank my friends and family members in India who have constantly offered me support and motivation.

I am most grateful to my parents and my brother. I am indebted for a lifetime for the love and support they have offered me. I extend my heart-felt gratitude to my mother who has always been there for me regardless of my concerns. Her selfless nature and sacrifices have played a

prodigious role in my success. In spite of the distance between us, she motivated me to be the best version of myself, every single day, and I cannot thank her enough.

I would like to thank Advanced Research Computing at Virginia Tech for providing the computational resources required for successful completion of this research work.

Table of Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

1.1. Motivation

In nature, it is common to encounter granular assemblies composed of arbitrary shaped and textured particles. The varying geometries of particles define the general characteristics of the assemblies and affects their dynamics. Practical applications such as biomass combustion, packing of systems, blood-cellular flow and fluidized beds are quintessential examples of granular assemblies. Experimental studies on three dimensional assemblies of particles are difficult to carry out owing to the complexity involved in capturing the motion of the minuscule particles using real-time imaging systems. Hence computational techniques such as Discrete Element Method (DEM), initially formulated by Cundall and Strack [1] and Particle Resolved Method (PRM) are used to model particulate systems. Initial representations of particle assemblies assumed that the particles were circular (in 2D) or spherical (in 3D). However, as has been discovered later, the surface definition of a particle contributes the most to the dynamic behavior of a cluster of particles[2], [3]. To replicate the dynamics of most prevalent fluid-solid particulate systems found in reality, the particles have to be computationally modeled as non-spherical in shape. The associated computational modeling techniques are time and memory intensive when compared to spherical particulate systems. However, with the advancement in computing power, simulating non-spherical particulate systems has been researched upon with increased interest; efficient parallel implementations of 3D multi-particulate non-spherical systems have been carried out and their coupling with CFD solvers has been investigated[4].

1.2. Overview of the Research Problem

Numerical simulations of 3D non-spherical particulate systems usually involves generation of particles and collision kinematics. Detecting the collisions and resolving them by computing the contact points, contact normal and interpenetration distance are integral in calculating the collision forces acting on particles. Subsequently, numerical techniques to update the trajectories of particles post collisions - in accordance with conservation of momentum and energy - have to be incorporated. Typical modeling techniques employ impulse based methods - illustrative of the change in momentum of non-deforming particles - or time-driven methods - illustrative of spring-dashpot models of deforming particles - to resolve collisions, the implementation of which is non-trivial for non-spherical particles[5], [6].

When dealing with non-spherical particles, superquadrics is a commonly used classification of particles. It is believed that 80% of the varied geometries exhibited by solid particles can be represented by superquadrics or sub-categories derived from them[7]. Superellipsoids are one among the group of convex shaped superquadrics which constitute a wide range of geometries - including spheres. Initially proposed by Barr[8], this group of particles has been used to model non-spherical particulate systems as they allow for a wide range of geometrical specifications more commonly found in nature.

A generic method aiding in the collision detection is the generation of neighborhood list for each particle and checking for possible contact. Data structures such as linked lists, binary tress and octrees are computationally efficient ways to check for collision between particles in a system [9] . Approximating the surface with bounding boxes (or cubes) and discrete orientation polytopes are beneficial in developing a model as the contact and collision mechanics are relatively

straightforward[10], [11]. However, there is a considerable loss of accuracy by not accounting for the particles' actual geometries and in CFD applications, these methods are therefore not viable. Recently, techniques to detect collisions between non spherical particles, namely, the Gilbert Johnson Keerthi (GJK) algorithm and the Orientation Discretization Database solutions have been used and the efficiency of these algorithms in detecting collisions have been documented [12], [13].

Detecting collisions between particles is most efficient when appropriate collision detection techniques are employed. Most DEM models have a geometry-specific collision detection technique implemented as it is critical in depicting the mechanics of real-life systems. However, a shortcoming of the model is that collision detection technique incorporated might limit the applicability of the model to a limited range of particulate assemblies. Specifically, in the area of rock-mechanics, contact detection algorithms pertaining to polyhedral particle collision detections are employed. In such cases, the collision detection algorithm has to be modified when diverse geometries of particles have to be modeled[14], [15]. Since superellipsoids encompass a major fraction of the particle textures found in nature, researches have come up with, and have documented, credible collision mechanics for superellipsoids. Inherently, certain geometrical features of superellipsoids are beneficial in detecting and resolving collisions which make them a desirable sub-category of non-spherical particles while modeling a system[16].

Superellipsoids utilize explicit and implicit functions to represent their surfaces. The manner in which the surface is defined determines the collision detection algorithm[17]. Two popular approaches utilized in generating a superellipsoid have been the Continuous Function Representation (CFR) and the Discrete Function Representation (DFR). In the case of particles with surfaces defined using a CFR, the contact detection can be done by numerically solving –

either directly or iteratively - for the intersection of the surface functions of two superellipsoidal particles. However, the non-linearity of the surface equations makes it computationally expensive and unrealistic to implement in certain cases to solve for possible contacts[2], [18]. Depending on the contact kinematics, the resulting contact normal from the collision detection algorithm can be imprecise [16]. Methods such as the common normal and the geometric potential methods are effective in detecting contacts between superellipsoids using CFR. However, these methods also have non-linear solution methods associated with them [19].

On the other hand, surfaces defined using a DFR are a more attractive technique as they are easier to implement. The use of a DFR to generate the particles' surfaces allows the development of a collision detection technique using an advantageous feature of the surface geometry. The surface function of superellipsoids is related to the distribution of the vertices on the surface of the superellipsoids. For a binary collision, the surface vertices on one superellipsoid are converted to the local coordinate space (body-fixed coordinate space) of the other particle and evaluated with respect to its surface function. Subsequently, depending on the resulting magnitude, contact between two superellipsoids can be accurately determined[20]. While this model involves coordinate transformations for particles, it is comparatively easier to implement than the common normal and geometric potential methods utilized in CFR of particles.

Following the determination of possible contacts, a resolution algorithm needs to compute information about the contact points, contact normal and the inter-penetration (overlap) distance. Generally, for collision resolution between particles, event-driven and time-driven models are used. An event-driven collision resolution model, devised using the impulse acting on particles during collision, and a time-driven based model represented by a spring-dashpot system, have been employed for collision resolution. Concretely, these models adhere to the laws of motion and have

been validated to be energy conservative[21]. While an event-driven method resolves collisions instantaneously (hard sphere), a time-driven method is representative of a deformable particle (soft sphere) resolution model. For the presently developed framework, both the hard sphere model and the soft sphere model have been utilized to resolve collisions. Theoretically the event-driven model does not need information about the inter-penetration distance, although depending on particles' kinematics, it might be required. Moreover, frictional contacts and tangential mechanics in real-life kinematics between particles can be incorporated in both these models. In relatively sparse assemblies, event-driven models function better compared to time-driven models and are more efficient, as the numerical time step could be set to a larger value. Other advantages such as the simplicity of the model and accurate representation of real-life physics make the event-driven model favorable. However, in dense systems, the model could perform poorly due to multiple contacts between particles and the possibility of losing particles due to the relocation adjustment (adjustment of the particles' locations based on the interpenetration distance). A time-driven model is beneficial in such cases and helps in circumventing these shortcomings[4].

To effectively simulate the particles' trajectories, both the event-driven and time-driven models need information about the contact point and the contact normal along which the collision forces act. Several approximations of the contact normal and contact point have been made in previous researches. If the common normal concept (or) the geometric potential algorithms are used to detect collisions, the contact point and the contact normal are corollaries resulting from the detection algorithm. It should be noted that the common normal and geometric potential algorithms differ in the computation of the above mentioned contact information[16], [22]. Alternatively, Dong et al [13] , in their research, have suggested a means of computing the contact point and the contact normal. Since information about the vertices in contact is available (if the implicit surface

function of either particle is used to detect contact), the contact point is designated to be the geometric center of the overlapping area (in 2D) or the overlapping volume (in 3D). Consequently, the contact normal can be determined by averaging the surface normal from each of the surface vertices in contact for both particles. These methods to determine the contact point and the contact normal are beneficial as they are applicable to both event-driven and time-driven models.

The inter-penetration distance can also be computed from the surface vertices of particles in contact with each other. Inter-penetration distance is of utmost importance to relocate the particle centroids post collision (in case of an event-driven model) and to calculate the effective collision forces acting on the particle (in case of a time-driven model). When consolidated as a whole, the information about the contact point, contact normal and the interpenetration distance serve as requisites to calculate the post collision trajectories of the particles.

1.3. Objectives and Contributions of Research

In this research work, a model incorporating the geometry definition, contact detection and the collision resolution methods for non-spherical particulate systems, modeled using superellipsoids has been developed. The kinematics of particulate assemblies and individual particle motion, carried out using event-driven and time-driven models, are numerically validated with canonical tests. The current model is devised in such a way that the collision kinematics and the surface definition of the particles are not restricted to specific geometries. Collisions between cylinders, ellipsoids and cuboidal shaped particles are processed in periodic and wall-bounded computational domains. Particle-particle collisions are incorporated using surface function characteristics of superellipsoids and particle-wall collisions are detected based on a novel method stemming from computational geometry. Collisions are subsequently resolved using an impulse

based method in the event-driven model and a linear spring-dashpot model in the time-driven model.

Practical applications such as fluidized beds, conveyors, densely packed particulate systems, blood flow and granular assemblies in rotating cylinders, wherein the computational domains differ significantly from each other, require a generic CFD-DEM model. Developing the mathematical framework to detect and process collisions and, interlacing the model with other subroutines, functioning as part of GenIDLEST - a general CFD solver -manifest as the significant outcomes of this research work. Furthermore, the entire model is structured in such a way that the computations can be parallelized. The model is implemented in a framework involving MPI communication in a multi – processor architecture. An evaluation of its performance is carried out and compared to that of the existing spherical DEM model in GenIDLEST. The computational efficiency in terms of the number of surface vertices, number of collision instances between particles, and the number of processors utilized is presented accordingly. The relative speed-up of the simulation wall-clock time with increasing number of processors is studied and documented. Also, a comparison of a fluidized bed simulation carried out with the developed model and previous studies is illustrated. The ensuing coupled CFD-DEM model will allow for more accurate predictions of drag and heat transfer coefficients of arbitrarily shaped dense fluid-solid systems.

Chapter 2

2. Methodology

Modeling three dimensional particulate systems is heavily dependent on the choice of technique to generate the geometry. The description of particle shape and size influences the choice of collision detection and resolution algorithms. Furthermore, the particle shape descriptors govern the applicability, accuracy and flexibility of the model. For instance, a DEM model described only by spheres or circles (in 2D) will have a higher computational efficiency which may not work well for non-spherical particles. On the other hand, non-spherical particle shapes offer a wider range of flexibility in terms of the shapes that can be simulated, with computational efficiency turning out to be a bottleneck in most cases. The choice of the surface definition consequently affects the collision algorithms and the post-collision kinematics. In this work, superellipsoids are generated using their surface function as this class of particles allows for a wide range of shapes to be simulated as illustrated in Figure 2-1.

2.1. Geometry Definition

In general, particles in a DEM simulation can be generated by the following techniques:

- Polyhedrons / Spheres
- Computational Mesh
- Composite particle approach
- Continuous Function Representation
- Discrete Function Representation

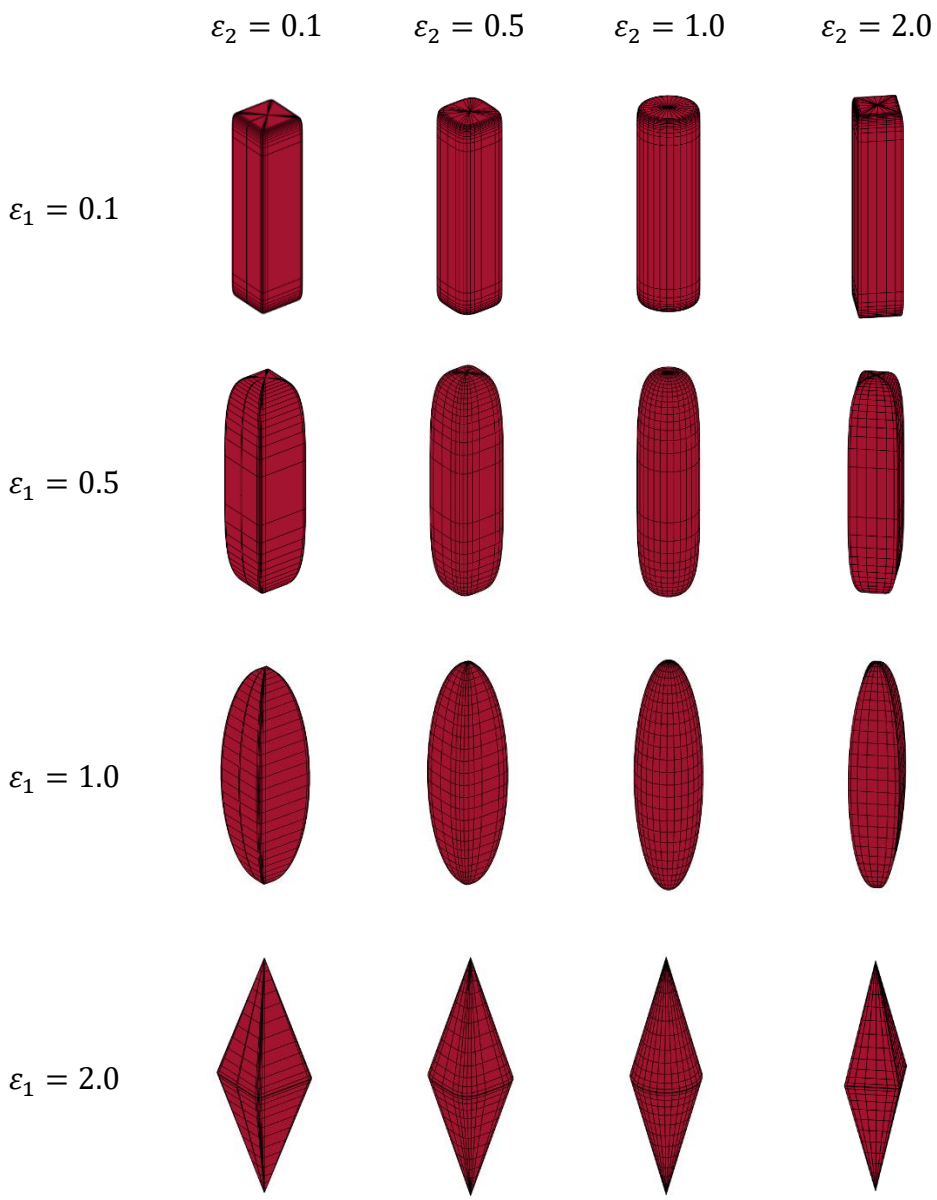

Figure 2-1 : Superellipsoids with varying squareness parameters and constant semi major axis lengths

The easiest method to describe particles' geometry is to use polyhedrons and spheres. In the case of polyhedrons, the surface can be defined by straight edges and vertices joining the edges. For spheres, the only information that needs to be tracked, as the particle evolves in space and time, is its centroid. As a result, the simulation becomes highly efficient compared to other

complex shapes. The collision detection and resolution methods are also well defined for polyhedrons and spheres. Collision between polyhedrons can be detected using one of the edge-edge, vertex-edge and vertex-vertex collision techniques depending on the type of collision encountered [23], [24]. An advantage of these techniques is that the supplemental information required to process the post collision kinematics are obtained with relative ease. Similarly, collisions between spheres are detected based on an inequality relation between the distance between the centroids and their respective radii. As is the case with polyhedrons, the collision detection algorithms for spheres yield sufficient information to resolve the collisions.

However, the use of polyhedrons and spheres severely limits the applicability of the DEM model. In the majority of granular assembly studies, the particles used are irregularly shaped and carrying out computational studies to recreate the tests, using polyhedrons or spheres, would ensue in significant loss of accuracy. In CFD-DEM studies, accuracy is of utmost importance as in many cases the necessary equipment to carry out an experimental study might not be available. Since the results would solely be based on numerical simulations, efforts have to be taken to ensure the fidelity of the results. In that regard, complex shape descriptors aid in reproducing the exact kinematics of particulate assemblies in practical applications.

Alternatively, an independent mesh could be generated around the particle's surface which would take significantly longer owing to the absence of any functional form defining the surface. Storing the surface descriptors using vertices, triangular or quadrilateral elements and the associated connectivity lists consume a significant amount of memory as well. However, a meshed representation could be widely applied to any particle geometry irrespective of its shape and texture.

Table 2-1 : Comparison of the different techniques to generate particle geometries

Geometry description method	Advantages	Disadvantages
Polyhedrons/Spheres	• Easy to implement • High computational efficiency • Highly reliable collision detection algorithms	• Significantly low accuracy when compared to experiments • Does not generalize to a lot of shapes and textures • Limited applicability to practical applications
CFR	• Wider applicability to practical applications • Generalizes well to varying shapes and textures	• Computationally very expensive • Not reliable collision detection algorithms • Relatively difficult to implement in global space
DFR	• Wider applicability to practical applications • Generalizes well to varying shapes and textures • Relatively easier to implement in global • Reliable collision detection algorithms	• Computationally expensive • Accuracy depends on the discretization technique used
Computational Mesh	• Wider applicability to practical applications • Generalizes well to almost all shapes	• Computationally very expensive • Significant usage of memory resources • Collision detection and resolution algorithms changes pertaining to the geometry
Composite particle approach	• Reliable collision detection algorithms • Can apply to certain non-spherical shapes	• Tough to implement • Loss of accuracy with increasing number of component spheres • Does not generalize well to shapes with sharp corners

Additionally, if storage and simulation time take precedence over accuracy, the surface definition of the particles could also be done with a composite particle approach, wherein the non-

spherical particle geometry is approximated by a cluster (or) collection of spheres, as shown in Figure 2-2. The multi-sphere approach – introduced by Favier et al (1999) - is easier to implement in terms of the collision detection and resolution algorithms as they employ spheres in their framework[26]. Also, since the composite structure of spheres do not change with time, the initially quantified physical properties could be used without much modification. However, an unfavorable characteristic of this model is its lack of applicability to a wide range of shapes. Since spheres can only be used to approximate other non-spherical shapes like ellipsoids and cylinders, the model is of little relevance to shapes with sharp corners. Furthermore, deciding upon the number of spheres that would be required to approximate a non-spherical particle is a laborious task. With increasing number of spherical particles in a composite structure, the non-spherical particle could be approximated better, although, due to the varying shapes and sizes, calculation of the moment of inertia turns out to be less accurate, in addition to the loss of computational efficiency[27].

When dealing with non-spherical particles, the most commonly used classes of particles are superellipsoids. The surface of a superellipsoid can be defined by both continuous and discrete function representations. A Continuous Function Representation (CFR) defines the surface by a mathematical relation, often in the particle's local coordinate system. For instance, a superellipsoid's surface can be described by the following explicit formulation:

$$\left(\left(\frac{x}{a_1}\right)^{2/\varepsilon_2} + \left(\frac{y}{a_2}\right)^{2/\varepsilon_2}\right)^{\varepsilon_2/\varepsilon_1} + \left(\frac{z}{a_3}\right)^{2/\varepsilon_1} = 1 \qquad (2.1)$$

where, 'a_1',' a_2' and 'a_3' are the semi major axes' lengths along x, y and z coordinates. The parameter 'ε_1' and 'ε_2' control the 'squareness' of the particle along the x-y plane and the z axis

respectively. This surface function is sometimes referred to as the ‘inside-outside’ function as any point evaluating to a value less than one lies inside the surface of the superellipsoid, whereas any point outside the surface of the superellipsoids evaluates to a value greater than one.

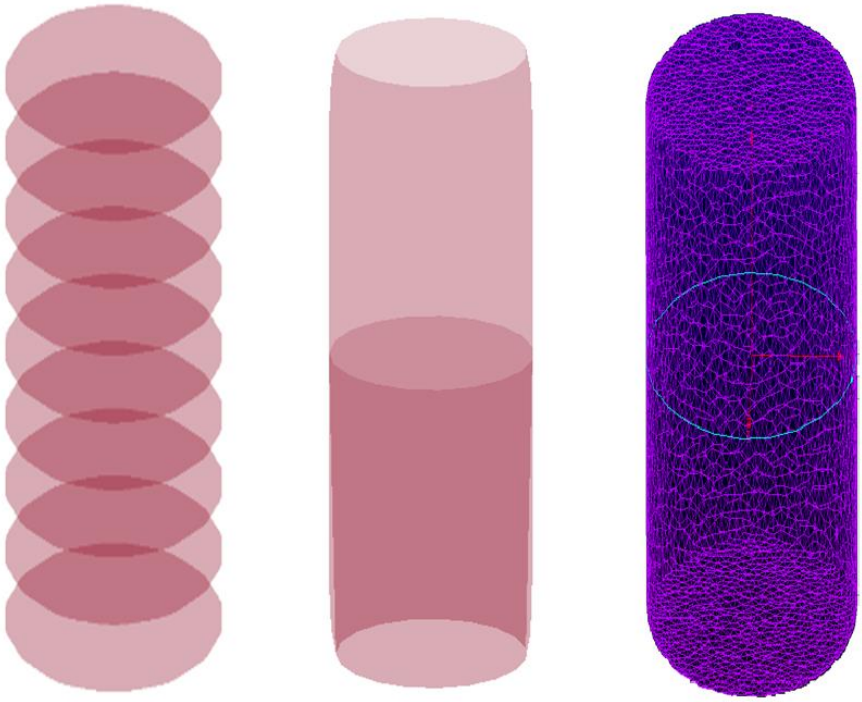

Figure 2-2 : Illustration of multi-sphere (left), CFR (center) and Meshed techniques (right) to generate a superellipsoid geometry

The biggest disadvantage of the continuous function representation is that the collision detection and resolution algorithms adjunct to the geometry definition are computationally intensive as they involve solving for non-linear equations, using Newton Raphson method, in global coordinate systems. Another limitation of the technique is that the solution can diverge when the initial estimates are incongruous with the final solution[19].

As a consequence, it is easier to model a superellipsoid with the implicit formulation given by,

$$x = a_1 * sgn(\cos\varphi_1) * |\cos\varphi_1|^{\varepsilon_2} * |\cos\varphi_2|^{\varepsilon_1}$$

$$y = a_2 * sgn(\sin\varphi_1) * |\sin\varphi_1|^{\varepsilon_2} * |\cos\varphi_2|^{\varepsilon_1} \quad (2.2)$$

$$z = a_3 * sgn(\sin\varphi_2) * |\sin\varphi_2|^{\varepsilon_1}$$

where the implicit parameter 'φ_1' varies from $-\pi$ to π and 'φ_2' varies from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. The implicit function is convenient for discretely approximating the surface. A Discrete Function Representation (DFR) refers to the method of discretizing a surface with nodes or vertices in such a way that the approximation replicates the continuous surface to a reasonable extent. Making use of this formulation and with a linearly spaced interval for φ_1 and φ_2, an equal distribution of vertices on the surface of the superellipsoid can be obtained for the majority of cases. However, for certain geometries such as a cylinder, it results in a paucity of vertices between the enclosing end-surfaces. In other words, there is an uneven distribution of vertices between the surfaces, which could pose an issue in contact detection and in the accuracy of the resolution algorithm. In order to circumvent this possible problem, the interval for φ_2 can be modified to be exponentially spaced as opposed to the conventional linear spacing as shown in Figure 2-3.

In practice, DFR method is implemented frequently when working with superellipsoids. The technique offers advantages in terms of accuracy and reliability over spherical surface description methods. The computational cost associated with DFR is higher than the spherical descriptors, however with the ability to parallelize the DEM model, the increased cost could be overcome to a manageable extent. Also, describing a surface using DFR would enable the use of simpler methods – in comparison with CFR - to detect and resolve the collisions. Specifically, the detection and collision resolution methods for superellipsoids have been well defined in the literature for geometries defined using DFR [2], [25]. In most cases, DFR is preferred to CFR as

the solutions when detecting collisions are guaranteed and unique. Moreover, DFR is preferred to meshed techniques as the simulation time and the storage required are less in comparison. A brief summary of the methods and their advantages and disadvantages are shown in Table 2-1.

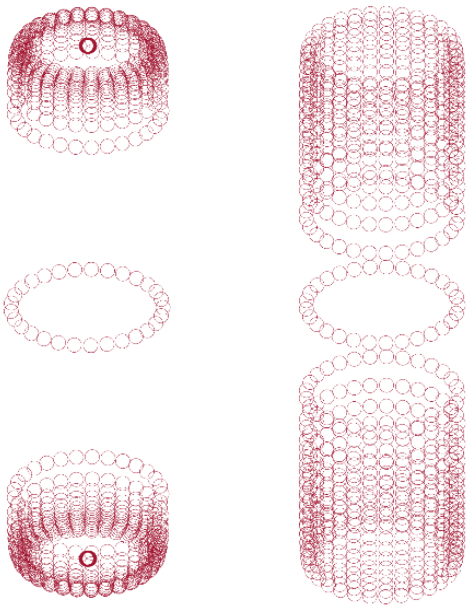

Figure 2-3. Representation of linearly (left) and exponentially (right) spaced surface vertices on a cylinder

2.2. Physical Attributes Definition

The physical quantities such as mass, volume, density and inertia are essential parameters that govern the resolution of collisions. The mass and density of a superellipsoid have a scalar magnitude associated with them. The volume and inertia – in local coordinates – can be expressed in terms of the dimensional and squareness parameters of a superellipsoid. It is also possible to mathematically calculate the inertia and volume based on integrals but for the sake of simplicity, the ensuing expressions of the integration operations are used. For a detailed derivation of these expressions a perusal of Jaklič (2000) is recommended [28].

i. Volume of a superellipsoid :

$$Volume \;=\; 2 * a_1 * a_2 * a_3 * \varepsilon_1 \; * \varepsilon_2 \; * \; \beta\left(\frac{\varepsilon_1}{2}+1, \varepsilon_1\right) * \; \beta\left(\frac{\varepsilon_2}{2}, \frac{\varepsilon_2}{2}\right) \qquad (2.3)$$

ii. Local inertia of a superellipsoid :

$$I_{xx} = \frac{1}{\rho} * \left(0.5 * a_1 * a_2 * a_3 * \varepsilon_1 \; * \varepsilon_2 * \left[\left({a_2}^2 * \beta\left(\frac{3\varepsilon_2}{2}, \frac{\varepsilon_2}{2}\right) * \; \beta\left(\frac{\varepsilon_1}{2}, 2\varepsilon_1 + 1\right)\right) + \left(4 * {a_3}^2 * \beta\left(\frac{\varepsilon_2}{2}, \frac{\varepsilon_2}{2}+1\right) * \beta\left(\frac{3\varepsilon_1}{2}, \varepsilon_1 + 1\right)\right)\right]\right) \qquad (2.4a)$$

$$I_{yy} = \frac{1}{\rho} * \left(0.5 * a_1 * a_2 * a_3 * \varepsilon_1 \; * \varepsilon_2 * \left[\left({a_1}^2 * \beta\left(\frac{3\varepsilon_2}{2}, \frac{\varepsilon_2}{2}\right) * \; \beta\left(\frac{\varepsilon_1}{2}, 2\varepsilon_1 + 1\right)\right) + \left(4 * {a_3}^2 * \beta\left(\frac{\varepsilon_2}{2}, \frac{\varepsilon_2}{2}+1\right) * \beta\left(\frac{3\varepsilon_1}{2}, \varepsilon_1 + 1\right)\right)\right]\right) \qquad (2.4b)$$

$$I_{zz} = \frac{1}{\rho} * \left(0.5 * a_1 * a_2 * a_3 * \varepsilon_1 \; * \varepsilon_2 * ({a_1}^2 + {a_2}^2) * \beta\left(\frac{3\varepsilon_2}{2}, \frac{\varepsilon_2}{2}\right) * \; \beta\left(\frac{\varepsilon_1}{2}, 2\varepsilon_1 + 1\right)\right) \qquad (2.4c)$$

The local inertia tensor is subsequently expressed as

$$I_{local} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \qquad (2.5)$$

Additionally, in the expressions for volume and density, the term $\beta(i, j)$ refers to the Beta function which is given by,

$$\beta(i,j) = \frac{\Gamma(i) * \Gamma(j)}{\Gamma(i+j)} \tag{2.6}$$

where, $\Gamma(i)$ is the gamma function - an extension of the factorial function for complex and real numbers.

2.3. Kinematics

The particles once generated are displaced by an offset magnitude in the global coordinate system. The offset magnitudes are chosen such that the particles do not overlap initially, irrespective of the number of particles. Spatial and temporal advancement of the particles are done by displacing the particles by the designated linear and rotational velocities at each time step.

At each time step, if $\vec{X}$ is a vector containing the coordinates of a surface point in R^3 space, and $\vec{X}_{centroid}$ is the vector of the centroid of the particle, in global space, then the vector of the surface coordinate is converted into the local coordinate equivalent by,

$$\vec{X}_{local} = R * \left(\vec{X} - \vec{X}_{centroid}\right) \tag{2.7}$$

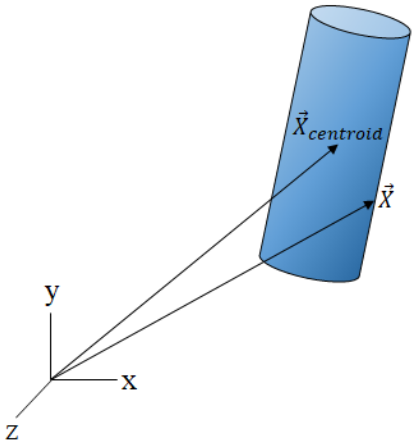

Figure 2-4 : Depicting the terminologies used for the global position vectors

where ‘R’ is the rotation matrix accounting for the orientation of the local coordinate space with respect to the global coordinate space. The sequence of converting a vector from global to local space is shown in Figure 2-5.

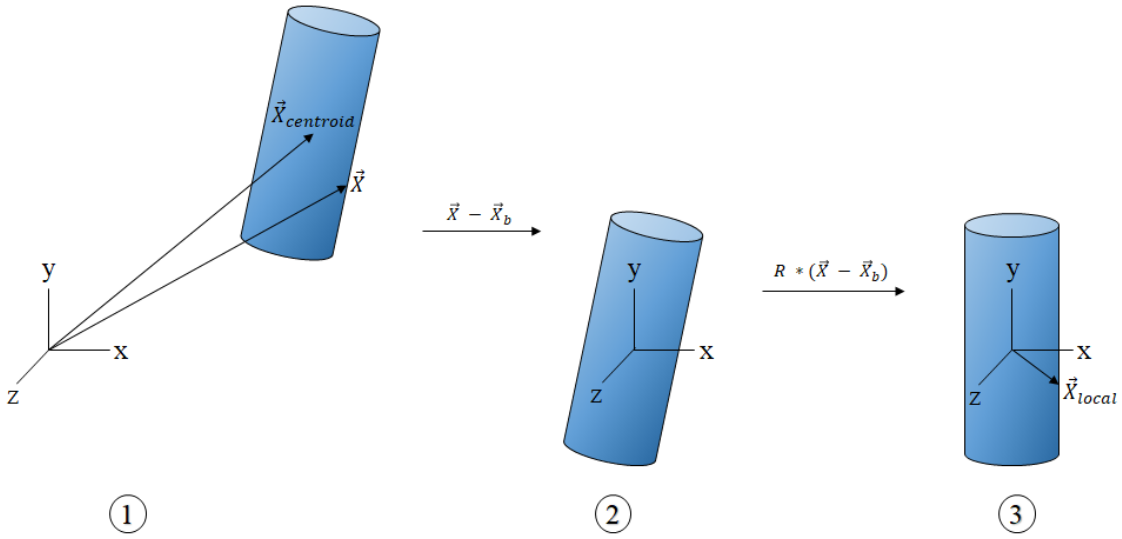

Figure 2-5 : Converting a position vector from global to local space

The rotation matrix can be computed using various methods. In this work, quaternions are used to compute the rotational matrix. This eliminates the problem of sequential (or) preferential rotation where the order of rotation matters. Using quaternions for computing the rotation matrix ensures that one single ‘resultant’ rotation is applied to the particle about a unit axis of rotation. At a given time step, an instantaneous unit quaternion is defined as:

$$\hat{Q}_{instant} = cos\frac{\theta}{2} + \left(\frac{\omega_x}{|\vec{\omega}|}\,\hat{\imath} + \frac{\omega_y}{|\vec{\omega}|}\,\hat{\jmath} + \frac{\omega_z}{|\vec{\omega}|}\,\hat{k}\right) * sin\frac{\theta}{2} \qquad (2.8)$$

In eq.2.8, $\vec{\omega}$ is given by,

$$\vec{\omega} = \langle\, \omega_x\,,\, \omega_y\,,\, \omega_z \,\rangle \qquad (2.9)$$

The individual components of $\vec{\omega}$ are the respective angular velocities along each dimension. The unit axis of rotation for the particles is given by the unit vector of angular velocity as expressed below.

$$\hat{\omega} = \langle \frac{\omega_x}{|\vec{\omega}|}\,\hat{\imath}, \frac{\omega_y}{|\vec{\omega}|}\,\hat{\jmath}, \frac{\omega_z}{|\vec{\omega}|}\,\hat{k} \rangle \tag{2.10}$$

This unit axis of rotation is the axis about which one resultant magnitude of rotation is applied, which is subsequently given by,

$$\theta = |\vec{\omega}| * \Delta t \tag{2.11}$$

where, the magnitude of the angular velocity vector ($|\vec{\omega}|$) is given by,

$$|\vec{\omega}| = \sqrt{{\omega_x}^2 + {\omega_y}^2 + {\omega_z}^2} \tag{2.12}$$

The magnitude of $\hat{Q}$ always equals one. To facilitate the computation of the rotation matrix, the unit quaternion vector is broken up into individual components as follows,

$$Q_1 = cos\left(\frac{\theta}{2}\right) \tag{2.13a}$$

$$Q_2 = \frac{\omega_x}{|\vec{\omega}|} * sin\left(\frac{\theta}{2}\right) \tag{2.13b}$$

$$Q_3 = \frac{\omega_y}{|\vec{\omega}|} * sin\left(\frac{\theta}{2}\right) \tag{2.13c}$$

$$Q_4 = \frac{\omega_z}{|\vec{\omega}|} * sin\left(\frac{\theta}{2}\right) \tag{2.13d}$$

Subsequently, the rotation matrix is assembled as illustrated below,

$$R = \begin{bmatrix} {Q_1}^2 + {Q_2}^2 - {Q_3}^2 - {Q_4}^2 & 2 * (Q_2Q_3 - Q_1Q_4) & 2 * (Q_2Q_4 - Q_1Q_3) \\ 2 * (Q_2Q_3 - Q_1Q_4) & {Q_1}^2 - {Q_2}^2 + {Q_3}^2 - {Q_4}^2 & 2 * (Q_3Q_4 + Q_1Q_2) \\ 2 * (Q_2Q_4 + Q_1Q_3) & 2 * (Q_3Q_4 - Q_1Q_2) & {Q_1}^2 - {Q_2}^2 - {Q_3}^2 + {Q_4}^2 \end{bmatrix} \quad (2.14)$$

Once the rotation matrix is constructed, the particle coordinates in global space are converted to the local space as shown in eq.2.7. To advance the particle temporally and spatially, its quaternion is updated as,

$$\widehat{\mathrm{Q}}^{n+1} = \hat{Q}_{instant} * \widehat{\mathrm{Q}}^{n} \quad (2.15)$$

In the above shown quaternion multiplication formulation, 'n' represents the time step. $\widehat{\mathrm{Q}}^0$ at the initial time step is the quaternion resulting from the particle's initial orientation. With the updated quaternion, the particle's local coordinates are transformed into its global space equivalent by applying the transformation,

$$\vec{X}_{global} = R^T * \vec{X}_{local} + \vec{X}_{centroid} \quad (2.16)$$

It is noteworthy that the inverse of the rotation matrix is equal to its transpose (i.e. $R^{-1} = R^T$) as the matrix is orthogonal. Spatial advancement of the particle also involves the translational motion. At any given time step, once the computations involving the rotational kinematics are done for the particle's surface coordinates, the final step in temporal and spatial advancement - taking the translational displacement into account - is executed as follows,

$$\vec{X}_{global} = \vec{X}_{global} + \vec{v} * \Delta t + \vec{\delta}_{total} \quad (2.17)$$

where, $\vec{v}$ is the linear (or) translation velocity vector for the given particle and $\vec{\delta}_{\mathrm{total}}$ is the total relocation vector resulting from the hard sphere model(refer section 2.4.5 and 2.5.1). The sequence

of computations illustrated in this section are applied to all the surface coordinates for all particles in the domain.

2.4. Particle-Particle Collision Detection

2.4.1. Overview of Collision Detection

Historically, collision detection has been a significant part of computer graphics and computational geometry. It refers to techniques involved in determining if two objects are in contact (merely touching) or have interpenetrated each other while undergoing specified actions. Popular applications wherein collision detection plays a vital role are robotics, computational biology, and in the development of gaming engines. In CFD-DEM, granular particles are under the influence of fluid forces in a confined computational domain. This leads to the possibility of encountering particles colliding with one another at multiple points. Detecting and resolving the collisions are essential to accurately represent the post-collision kinematics of the particles, subsequently effecting realistic changes in the flow-field through coupled CFD solvers. The representation of the geometry, in many ways, outlines the techniques that would be applicable to model collisions between the particles in CFD-DEM. Common practices have modeled particles as spheres as detecting collisions between them are fairly straightforward. Intuitively, since the particles are governed by the equations of motions and their interaction with fluid flow can also be modeled, collisions between particles can be accounted for by numerically solving the global spatial surface equations of the concerned particles. However, this method results in a lot of computational effort and the efficiency associated with this technique reduces as the number of particles increases.

Collision detection between non spherical particles have been, traditionally, not as conventional as that for spherical or polyhedral particles. For spherical and polyhedral particles,

bounding volume hierarchies that spatially approximate the particle with spheres or polyhedral are effective in detecting collision. The underlying assumption is that if two objects are possibly colliding, the projection of their bounding volumes, on all the coordinate axes, have to be overlapping[29]. The bounding volumes by themselves, can take one of the usually preferred geometrical forms, namely, Axis Aligned Bounding Boxes (AABBs) and Objected Oriented Bounding Boxes (OOBBs). An AABB is a computationally inexpensive method of monitoring possible contacts between particles. Each particle has a bounding volume aligned with the global coordinate axes, generated around its surface. For non-rotating particles the performance of AABB is similar to that of OOBB, however for rotating particles, the bounding volumes tightly aligned to the coordinate axes offer poor performance in detecting collisions, in addition to an increase in its volume, as illustrated in Figure 2-6 . An improved alternative to AABBs are the OOBBs which have a bounding volume generated along the body-fitted (local) axes. While the particle is in rotation, the OOBBs also align in close approximation with the rotated surface, as depicted in Figure 2-7. Comparatively, OOBBs offer a better approximation of the particle surface and an enhanced performance in collision detection, however, they are slightly more expensive to generate.

A lesser known method of creating bounding volumes is the Discrete Oriented Polytopes (K-DOPs), with ‘K’ denoting the number of edges constituting the bounding volume. These bounding volumes are more compact than OOBBs and offer a better approximation by minimizing the area between the bounding volume and the actual particle’s surface. Nevertheless, they are complex to generate and would require increased computation effort to detect overlaps[10], [30].

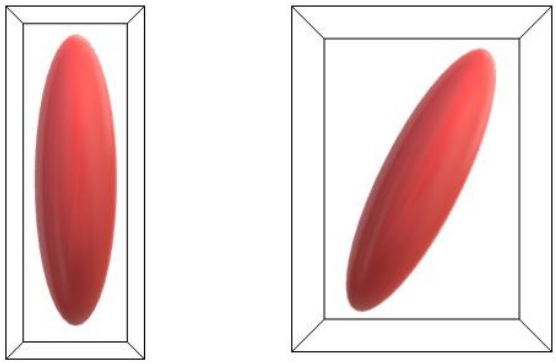

Figure 2-6 : Axis Aligned Bounding Box (note the increase in the size of the bounding volume when the particle rotates)

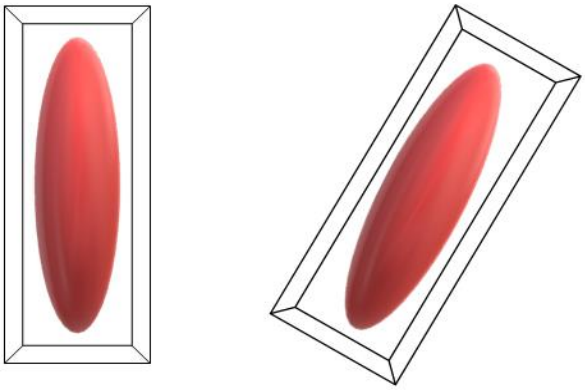

Figure 2-7 : Object Oriented Bounding Box (note the alignment of the bounding volume with the rotated axis of the particle)

For complex particle shapes, these methods perform poorly in collision detection. Another method employed to reduce computational effort is to assume spatio-temporal coherence. This stipulates that the objects/particles do not move or traverse for large distances spatially and temporally. Specifically, the search for possible contacts doesn't have to be performed at every iteration. However, for this assumption to hold valid, the time step has to be sufficiently small; even so, in a myriad of cases, this assumption doesn't work well for dense particulate systems[29], [31].

For non-spherical particle shapes, distinct collision detection methods are preferred to the common methods. Specifically, in CFD applications, bounding volume hierarchies are generally not used, since the accuracy of the simulation takes precedence. Superellipsoids - constituting the broad classification of non-spherical particle geometries - have their surfaces represented by a continuous function. Intuitively, to detect possible contact/inter-penetration between two particles, their surface functions can be mathematically solved at each time step. In computational terms, solving these non-linear equations is very expensive and sometimes may not yield a solution[19], [32]. Hence a discrete function representation manifests as a more favorable method. The surface of the particle is parameterized by arguments and vertices are generated at equally (or) unequally spaced intervals[33]. Furthermore, the collision detection algorithms applied to discrete function representations are more beneficial when coupled with collision resolution algorithms. Nevertheless, it is inefficient to have a solitary level for collision detection between superellipsoids particles; most collision models have multiple levels checking for possible contacts, with each level in the hierarchy decreasing the eventual number of collision pair checks.

The model developed in the present work consists of three levels in the hierarchy of collision detection. Initially, a neighborhood list is generated for each particle in the domain. This is effectively done by dividing/binning particles into separate cells on a background grid and monitoring the specified cell and the neighboring cells' particles. From Figure 2-8, all the particles in the surrounding cells are flagged for a possible collision check with 'P1'. At the next level in the hierarchy, an imaginary sphere with a diameter comparable to the largest dimension of P1 is used as its domain of influence and particles falling under this sphere of influence comprise the neighborhood list of P1. From Figure 2-8, 'P1' and 'P2' are flagged for a possible collision check, as 'P2' falls in the sphere of influence (depicted in black dashed lines) for 'P1'.

However, to detect collisions during intricate dynamics of particles, an additional level of collision detection is required. For any point on the surface of a superellipsoid, expressed in local coordinates, the surface function evaluates to unity. Likewise, for a point inside the superellipsoid surface its surface function evaluates to a value less than one, while any point outside the surface evaluates to a value greater than one. This particular feature of a superellipsoid is used to detect contact between a particle pair. Thus for the final collision check to detect collision between two superellipsoids, the surface coordinates of one particle are evaluated with respect to the surface function of the other particle. Depending on its magnitude, collisions between two particles can be determined accurately[2].

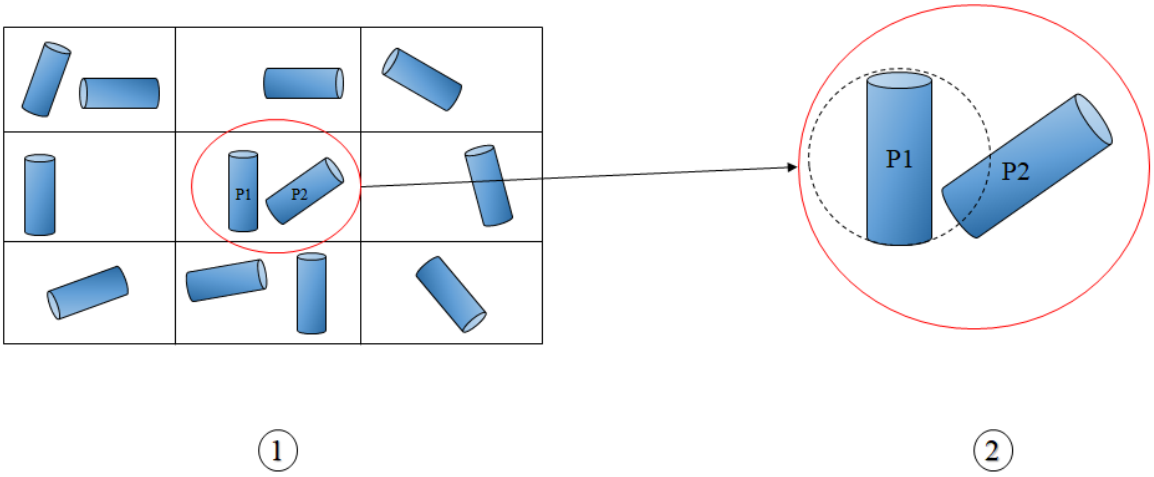

Figure 2-8 : Representation of the higher two levels in the hierarchy of collision detection

2.4.2. Collision Detection Algorithm

To effectuate the final level in the collision detection hierarchy, for any two particles possibly in contact - as determined in the previous stage in the hierarchy – the surface coordinates in global space for one particle are converted to the local space of the other particle. For surface

coordinate vectors of the two particles, in global space, delineated as $\vec{X}_{1,global}$ and $\vec{X}_{2,global}$ respectively,

$$\vec{X}_{12} = R_2 * \left(\vec{X}_{1,global} - \vec{X}_{1,centroid}\right) + R_2 * \left(\vec{X}_{1,centroid} - \vec{X}_{2,centroid}\right) \quad (2.18)$$

gives the equivalent position vector in the local space of the second particle. The rotation matrix R_2 is obtained from calculations shown in the previous section, utilizing the orientation vector of the second particle at the concurrent time step. $\vec{X}_{1,centroid}$ and $\vec{X}_{2,centroid}$ denote the global position vectors of the centroids of two particles, respectively, with the expression $(\vec{X}_{1,centroid} - \vec{X}_{2,centroid})$ representing the displacement vector. Similarly, for the other particle,

$$\vec{X}_{21} = R_1 * \left(\vec{X}_{2,global} - \vec{X}_{2,centroid}\right) + R_1 * \left(\vec{X}_{2,centroid} - \vec{X}_{1,centroid}\right) \quad (2.19)$$

gives the equivalent position vector in the local space of the first particle, for the global position vector of the second particle. Once these local space vectors are computed, they are evaluated with respect to the surface function of the other particle. Designating the components of the three dimensional local space vector $\vec{X}_{12}$ as,

$$\vec{X}_{12} = < x_{12}, y_{12}, z_{12} > \quad (2.20)$$

the surface function of the second particle is evaluated with respect to these components. Mathematically, the equation takes the form,

$$f_1 = \left(\left(\frac{x_{12}}{a_1}\right)^{2/\varepsilon_2} + \left(\frac{y_{12}}{a_2}\right)^{2/\varepsilon_2}\right)^{\varepsilon_2/\varepsilon_1} + \left(\frac{z_{12}}{a_3}\right)^{2/\varepsilon_1} \quad (2.21)$$

where, a_1, a_2, a_3 are the semi-major axes lengths, along the three dimensions, of the second particle. Correspondingly, ε_1 and ε_2 represent the squareness parameters of the second particle.

In similar fashion, the surface function evaluation f_2 for the local space vector $\vec{X}_{2,local}$ is carried out with respect to the dimensional and squareness parameters of the first particle. If f_1 and f_2 have a magnitude equal to or less than one, then there is a conclusive contact between the two particles. It has to be noted that, although the process of collision detection has been illustrated for vectors, the application to an array of surface coordinates follows the same procedure. The sequence of operations described in this section are shown in Figure 2-9.

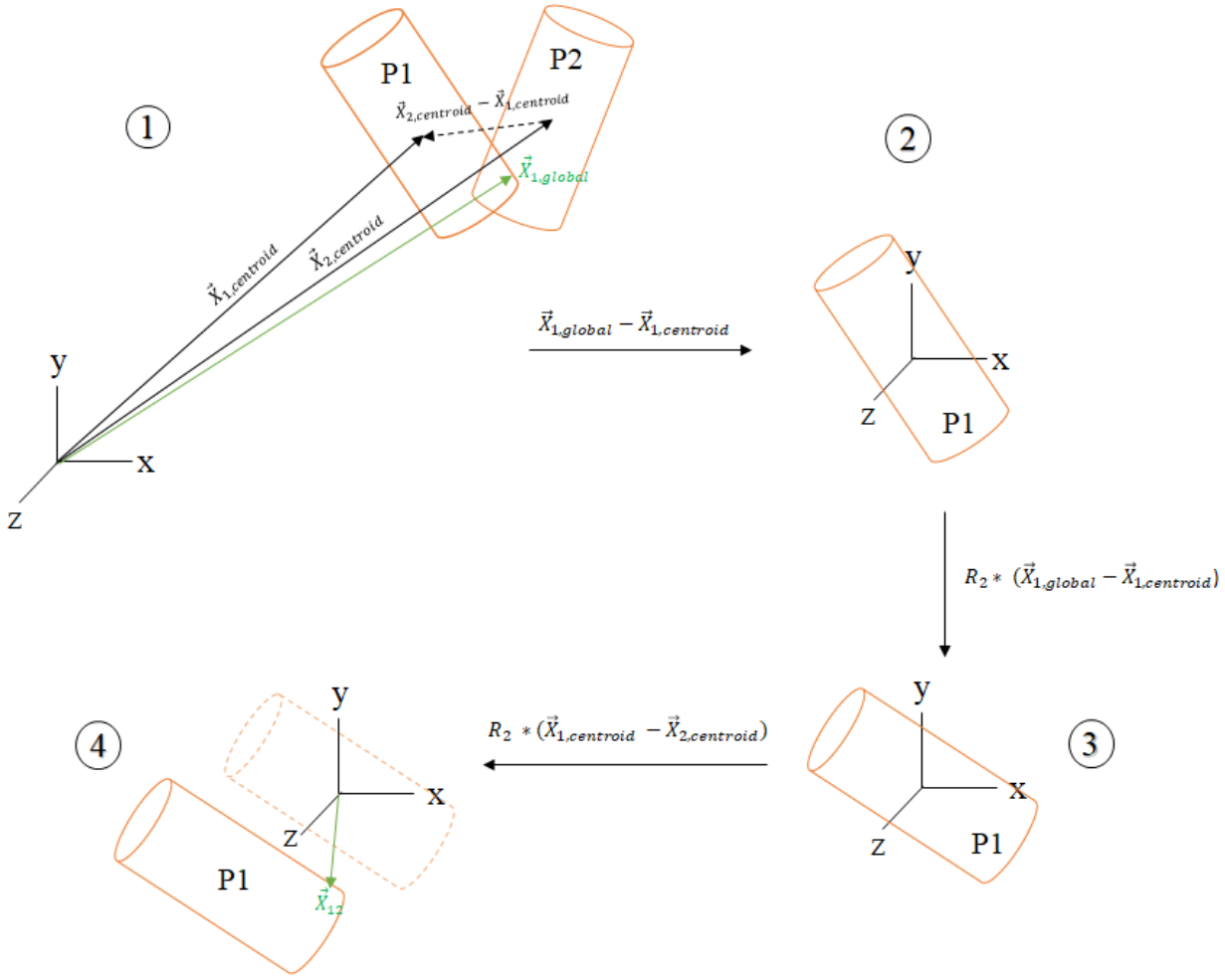

Figure 2-9 : Illustration of the steps involved in conversion of a global position vector of one particle into a vector in the local space of the other particle

2.4.3. Computation of the Contact Point

For CFD–DEM applications, a collision detection algorithm is supplemented by information about the collision such as, the contact point, contact normal and the overlapping distance. While resolving collision between particles, the aforementioned information is of significance. The contact point is the geometrical center of the overlapping volume[34]. All the surface coordinates that evaluate to a value equal to or less than one, after the computations for collision detection, are stored separately in an array. The local coordinates are then converted back to the global space by reversing the transformations. For instance, if $\vec{X}_{12}$ is a local coordinate vector of the first particle, that has a surface function value - with respect to the local space of the second particle - equal to or less than 1, then the global space equivalent is obtained by,

$$\vec{X}_{1,local} = \vec{X}_{12} - \left(R_2 * \left(\vec{X}_{1,centroid} - \vec{X}_{2,centroid}\right)\right) \quad (2.22)$$

$$\vec{X}_{1,global} = \left({R_2}^T * \vec{X}_{1,local}\right) + \vec{X}_{1,centroid} \quad (2.23)$$

Similarly, for a local coordinate vector of the second particle ($\vec{X}_{21,local}$), the global space equivalent is obtained. It has to be noted that in the majority of cases, there is an array of coordinate vectors that satisfy the surface function condition, in the event of a collision. Hence the inverse transformations from local space to global space are applied to an array of coordinates and not just a vector.

Once the global positional coordinate arrays are assembled, the mean of all the coordinates along each dimension results in the geometric center of the overlapping volume and is consequently chosen as the mean contact point in global space. For instance, if $\bar{\bar{X}}_{1,global}$ is the global positional coordinate array consisting of all the colliding vertices of the first particle and

$\bar{\bar{X}}_{2,global}$ is that of the second particle – wherein each colliding surface vertex is computed in the exact same manner as $\vec{X}_{1,global}$ with the corresponding rotational matrix and centroid vector – the global contact point (depicted as $\vec{X}_{mcp}$) is a result of the expression given below,

$$\vec{X}_{mcp} = \frac{1}{k}\sum_{K} \bar{\bar{X}}_{1,global} + \frac{1}{m}\sum_{M} \bar{\bar{X}}_{2,global} \qquad (2.24)$$

where, 'K' refers to the number of colliding vertices detected on the first particle and 'M' refers to the number of colliding vertices detected on the second particle. In most scenarios, it is highly probable that 'K' might not be equal to 'M' because of slight errors arising due to floating point calculations (or) the nature of contact. The summation operator basically adds to a '3×1' vector in the '3×k' or '3×m' array in a three-dimensional computation.

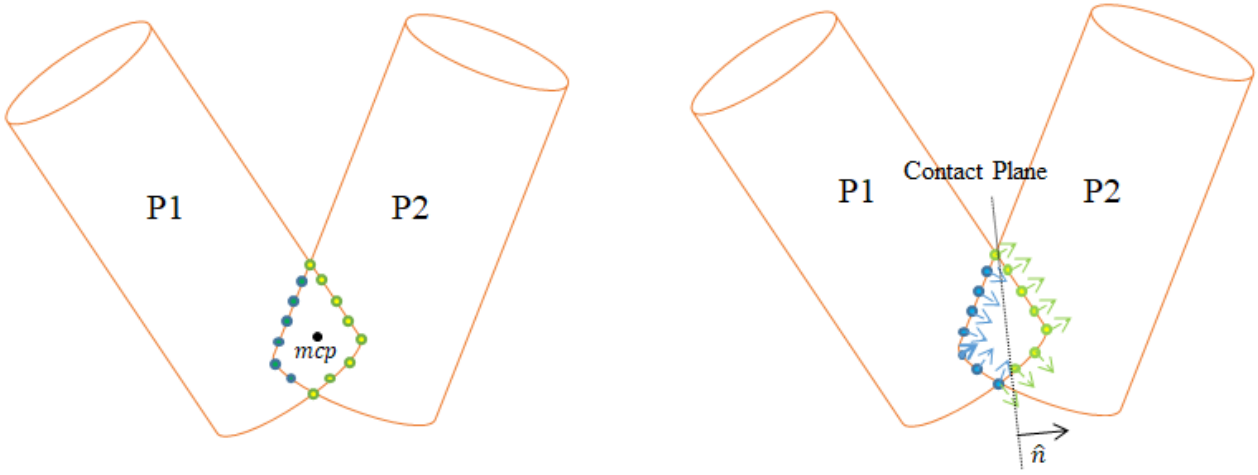

Figure 2-10 : Computation of the mean contact point (left) and the contact normal (right). Note that the surface vertices and surface normals for P1 are shown in blue whereas that for P2 are shown in green

2.4.4. Computation of the Contact Normal

The evaluation of the contact normal – normal to the plane of contact – also involves the use of the local coordinate vectors in contact with the other particle. Identical to the terminology used in the previous paragraphs, if $\vec{X}_{12}$ is a local coordinate vector of the first particle, that has a surface function value - with respect to the local space of the second particle - equal to or less than 1, then the vector is first converted to the local space of the corresponding particle.

$$\vec{X}_{1,local} = \vec{X}_{12} - \left(R_2 * \left(\vec{X}_{1,centroid} - \vec{X}_{2,centroid}\right)\right) \tag{2.25}$$

Considering $\vec{X}_{1,local}$ to be composed of three components denoted as,

$$\vec{X}_{1,local} = < x_1, y_{1,} z_1 > \tag{2.26}$$

the gradient of the surface function of the respective particle is then evaluated with respect to the local coordinate vector $\vec{X}_{1,local}$ to result in the outward pointing contact normal of the surface at that particular node.

$$\overrightarrow{\nabla f_1} = \vec{\nabla} f|_{(x_1, y_1, z_1)} \tag{2.27}$$

The individual components of the gradient vector are given by the following expressions.

$$\left.\frac{\partial f}{\partial x}\right|_{(x1,y1,z1)} = \frac{2}{\varepsilon_1}\left(\left(\frac{x_1}{a_1}\right)^{2/\varepsilon_2} + \left(\frac{y_1}{a_2}\right)^{2/\varepsilon_2}\right)^{\frac{\varepsilon_2 - \varepsilon_1}{\varepsilon_2}} * \left((x_1)^{\frac{2-\varepsilon_2}{\varepsilon_2}}\right) * \left((a_1)^{\frac{-2}{\varepsilon_2}}\right) \tag{2.28a}$$

$$\left.\frac{\partial f}{\partial y}\right|_{(x1,y1,z1)} = \frac{2}{\varepsilon_1}\left(\left(\frac{x_1}{a_1}\right)^{2/\varepsilon_2} + \left(\frac{y_1}{a_2}\right)^{2/\varepsilon_2}\right)^{\frac{\varepsilon_2 - \varepsilon_1}{\varepsilon_2}} * \left((y_1)^{\frac{2-\varepsilon_2}{\varepsilon_2}}\right) * \left((a_2)^{\frac{-2}{\varepsilon_2}}\right) \tag{2.27b}$$

$$\left.\frac{\partial f}{\partial z}\right|_{(x1,y1,z1)} = \frac{2}{\varepsilon_1} * \left((z_1)^{\frac{2-\varepsilon_1}{\varepsilon_1}}\right) * \left((a_3)^{\frac{-2}{\varepsilon_1}}\right) \quad (2.27c)$$

where, a_1, a_2, a_3 are the semi-major axes lengths, along the three dimensions, of the first particle. Correspondingly, ε_1 and ε_2 represent the squareness parameters of the first particle. Similar to the computation of the contact point, this process is repeated for all the contact vertices satisfying the surface equation of the other particle. Eventually, this results in an array of surface normal vectors for one particle, in the local space pertaining to that particle. An averaged local surface normal is computed by averaging (mean) all the local surface normal vectors along the three dimensions.

$$\overrightarrow{n_1} = \frac{1}{K}\sum_{k=1}^{K} \vec{\nabla} f|_{(x_k,y_k,z_k)} \quad (2.29)$$

where 'K' refers to the number of vertices of the first particle in contact or overlap with the second particle. $\vec{\nabla}f|_{(x_k,y_k,z_k)}$ denotes the 'k^{th}' surface normal vector of the contact node of the first particle. The contact normal is desirable in global space as it is intuitively easier to implement the collision resolution algorithm in global space. Hence, the averaged surface normal in local space is converted to global space by accounting for the change in orientation.

$$\vec{n}_{1,global} = {R_1}^T * \overrightarrow{n_1} \quad (2.30)$$

In a similar manner, the averaged surface normal is computed for the second particle in global space ($\vec{n}_{2,global}$). The rotation matrix and the dimensional and squareness parameters are accordingly used. The contact normal of the collision between the concerned two particles, is the resulting vector from the average of the two global surface normal vectors.

$$\vec{n} = 0.5 * \left(\vec{n}_{1,global} + \left(-\vec{n}_{2,global}\right)\right) \quad (2.31)$$

The direction of the averaged global surface normal from the second particle is reversed to be consistent with the direction of the resulting contact normal. The contact normal always points in the direction of the other particle from the concerned particle's surface. It is noteworthy that these operations are carried out for an array of colliding vertices and the resulting averaged surface normal for each particle is used to calculate the mean contact normal of collision in eq. 2.31. To facilitate the computation of forces and impulses in the respective collision models, the averaged surface normal vector is normalized as follows to obtain the unit normal vector,

$$\hat{n} = \frac{\vec{n}}{|\vec{n}|} \tag{2.32}$$

Evidently, the accuracy of the collision detection, the computation of contact points, and the contact normal are dependent on the discretization (or) the number of vertices on the surface [18]. A simplified schematic representing the mathematical formulation involved in computing the contact point and the contact normal is depicted in Figure 2-10.

2.4.5. Determination of Penetration Depth in Particle–Particle Collisions

A constituent of any collision model, the penetration depth is required to accurately resolve the collisions. The penetration depth is a requisite to process collisions in a time-driven collision model. Inherently the event-driven model does not require the penetration depth as the collision is processed instantaneously. However, if the numerical time step is too large, the particles might overlap with each other and as a consequence, the model requires the penetration depth to push the particles apart post collision.

Concomitant to collision detection algorithms such as geometric potential and common normal methods is the penetration depth[16], [22]. The collision detection algorithm used in this

DEM model does not yield the penetration depth by itself. However, since a discrete function representation is used to represent the particle geometry, the information about the surface vertices is attainable. The collision detection algorithm is based on the evaluation of a global surface coordinate of a particle in terms of the local coordinate space of the other particle; surface coordinates evaluating to a value less than or equal to one are flagged distinctly. The penetration depth is obtained by calculating the maximum projected distance along the normal between colliding vertices of the two particles. The method of calculating the magnitude of penetration is very similar to the one employed in the aforementioned popular algorithms.

Denoting the flagged global vertices of one particle as $\vec{X}_{1,flag}$ and that of the other particle as $\vec{X}_{2,flag}$ the penetration depth is calculated as,

$$\delta = \left|\left(\vec{X}_{2,flag} - \vec{X}_{1,flag}\right).\hat{n}\right| \tag{2.33}$$

The above computation is done for each pair of colliding vertices between two particles. The maximum magnitude of penetration is given by the maximum projection of vectors joining any two colliding vertices, along the contact normal, the mathematical formulation of which is given by,

$$\delta_{max} = \underset{i=1:K}{max}\left[\underset{j=1:K}{max}\left|\left(\vec{X}_{2i,flag} - \vec{X}_{1j,flag}\right).\hat{n}\right|\right] \tag{2.34}$$

with 'K' denoting the number of surface vertices that have been flagged as 'in contact' with the other particle. The minimum number of vertices that could be flagged is one and the maximum number of vertices that could theoretically be flagged is the maximum number of surface vertices on the particle. The collision algorithm is devised in such a way that the magnitude of penetration is calculated only when there is at least one vertex from each particle in contact with the other

particle. Otherwise, the magnitude of penetration is set to zero and the relocation vector would be a null vector. In practical terms, this assumption makes sense: in cases where one particle detects collision and the other doesn't – stemming from imperceptible differences in floating point calculations - the magnitude of penetration would be zero since the particles would not be overlapping. If there was a considerable deformation, the detection algorithm would result in surface vertices detected from both particles. As an eventual step in the collision detection model, the penetration depth calculated is represented in the form of a vector so that the entire process of spatial and temporal advancement could be vectorized.

$$\vec{\delta} = \delta_{max}\,\hat{n} \tag{2.35}$$

In eq.2.34, $\vec{\delta}$ depicts the relocation vector essential to push the particles away completely during collisions.

2.5. Particle-Wall Collision Detection

In wall-bounded domains, it is necessary to detect collision of particles with walls. In spherical DEM simulations collisions between particles and wall boundaries can be detected by checking if the magnitude of the vector from the particle's center of mass to the cell centroid of the wall boundary evaluates to a value less than the radius of the spherical particle. However, this technique is not applicable for non-spherical particles as the particles do not exhibit rotational symmetry. Traditionally, the detection techniques used for superellipsoids are similar to the particle-particle detection method. The biggest drawback of this technique is the fact that the walls have to be discretized in a dense fashion. This would increase the number of vertices on the surface which in turn would increase the computational time and the memory required. An alternative technique is to use a minimization function that monitors the distance from the superellipsoid's

surface to the wall. If the function evaluates to a value less than the threshold set, a contact is detected. However, the distance function computation requires significant computation and in some cases, direct solution of the function involves non-linear solution methods. Thus, a different technique has to be put into practice to detect collisions of particles with walls.

In the current framework, a collision detection technique based on the methods derived from computational geometry has been implemented. Whenever a particle's center of mass is in one of the three computational cells closest to the wall in any given direction, the particle is flagged to be checked for a possible collision. Subsequently, each of the particle's surface vertex is checked for collision with the wall. In any computational domain with walls, information about the wall normal is available. The vector between the center of wall cell ('wall node') and each of the surface vertices is projected along the inward pointing wall normal. If the expression evaluates to a negative value, the particle has extended beyond the wall boundary in the given direction and a collision is detected. For any given surface vertex of the flagged particle denoted as $\vec{X}_{global}$, the collision detection with a wall boundary is carried out as follows,

$$eval = (\vec{X}_{global} - \vec{X}_{wall\ node}) . \hat{n}_{wall} \tag{2.36}$$

If '$eval$' is negative, a collision is detected and the particular vertex $\vec{X}_{global}$ is stored in a separate array. An array of all colliding surface vertices is assembled and stored to calculate the global contact point. The computation of global contact point in the case of a particle-wall collision is straightforward as there aren't transformations involved in converting coordinates from the local to the global frame of reference. The contact point is the geometric mean of the colliding vertices and is given by,

$$\vec{X}_{mcp,wall} = \frac{1}{K}\sum_{K}\bar{\bar{X}}_{1,global} \tag{2.37}$$

where, 'K' represents the total number of surface vertices of the particle in collision with the wall boundary. Following the computation of the contact point, the contact normal – a requisite to resolve collisions – has to be worked out, the calculation of which is quite uncomplicated compared to the technique utilized in particle-particle collision detections. The contact normal for any particle-wall collision would simply be the normal of the wall boundary. In order to facilitate calculations in the collision resolution model, the inward pointing wall normal is reversed.

$$\hat{n}_w = -\hat{n}_{wall} \tag{2.38}$$

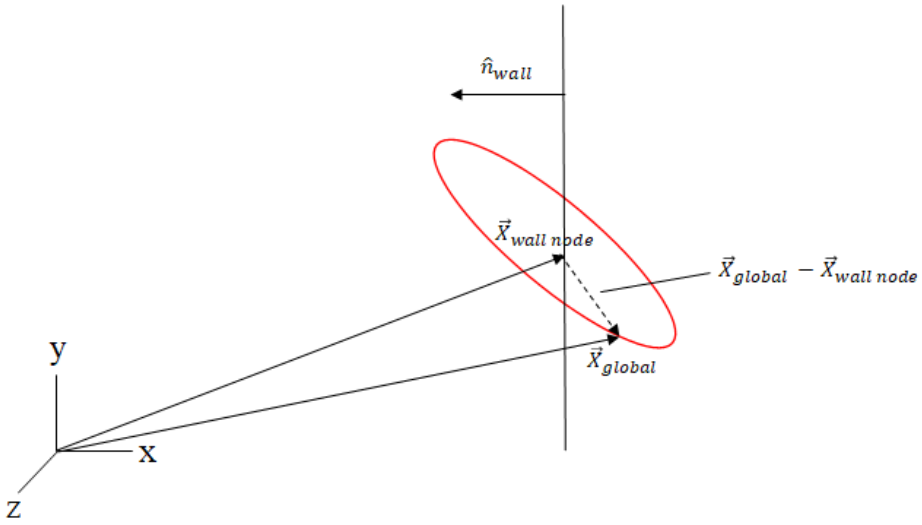

Figure 2-11 : Representation of the technique involved in detecting collisions between a particle and a wall

2.5.1. Determination of Penetration Depth in Particle-Wall Collisions

Similar to the particle-particle collision model, the magnitude of penetration is auxiliary information from the particle-wall collision model. The advantage of using a discrete function

representation is that the maximum magnitude of penetration can be computed with relative ease compared to a continuous function representation. The colliding vertices are flagged during the particle –wall collision detection process and they are assembled in the form of a two dimensional array. For any given constituent vector, the magnitude of penetration is computed as follows.

$$\delta = \left|(\vec{X}_{global} - \vec{X}_{wall\ node}) \,.\, \hat{n}_{wall}\right| \tag{2.39}$$

The terminologies for the components in the above expression follow from the descriptions in the previous section elucidating on the particle-wall contact detection technique. The maximum penetration depth is quantified with a reduction operation carried out over all the colliding surface vertices, similar to that in particle-particle collisions.

$$\delta_{max\,wall} = \underset{k=1:\,K}{max}\left|(\vec{X}_{k,global} - \vec{X}_{wall\ node})\ .\hat{n}_{wall}\right| \tag{2.40}$$

In an attempt to generalize the calculations, the maximum magnitude of penetration calculated is recast in the form of a vector, as characterized in the following equation.

$$\vec{\delta}_{wall} = \delta_{max\,wall}\ \hat{n}_w \tag{2.41}$$

The computation of the relocation vector is essential to ensure the particles do not move beyond the computation domain. The vector is added cumulatively to the relocation vector for a given particle, following similar computations ensuing from particle-particle collisions.

$$\vec{\delta}_{total} = \vec{\delta} + \vec{\delta}_{wall} \tag{2.42}$$

Chapter 3

3. Collision Resolution

3.1. Particle-Particle Hard Sphere Modeling

Resolving collisions refers to the resolution of the contact forces acting on the particles in the case of a binary collision. Additionally, in a CFD simulation, there are fluid forces, body forces and surface forces acting on the particle, which affect the resolution of the particle dynamics. In order to resolve the force due to collision, numerical solution models, event-driven models and time-driven models have been popularly used. An event-driven model based on impulse methods - representative of a non-deforming particle model - and a time-driven model - representative of a spring-dashpot system - have been employed in the current framework to resolve the collisions. An impulse based model is used to update the trajectories of non-spherical particles based on the instantaneous change in momentum. Inherently, an impulse based model does not need to account for the inter-penetration while relocating the particles in collision. However, due to the effect of numerical time step and particle dynamics, information about the penetration depth is required to resolve collisions completely. The collision resolution model is simplistic in implementation and allows for frictional/tangential kinematics to be incorporated in the resolution algorithm. [21], [35], [36]

The underlying concept in an impulse based resolution model is that when two objects collide, there is a considerable magnitude of reaction forces acting on the objects for an infinitesimally short period of time, thereby producing impulses. Mathematically, an impulse is represented as,

$$\overrightarrow{J_r} = \int \overrightarrow{F_r}\, dt \tag{3.1}$$

where, $'\overrightarrow{F_r}'$ represents the aggregate force acting on the object. For two particles involved in a collision, the impulses acting on them effect a change in their velocities. To conserve energy and momentum, the impulse direction is reversed for particle which has the averaged contact normal directed outward from its surface as illustrated in Figure 3-1.

3.1.1. Frictionless Impact

From Newton's second law of motion, the relation between the linear velocity and the force acting on a particle is as follows,

$$\overrightarrow{F_r} = m\frac{d\vec{v}}{dt} \tag{3.2}$$

$$\int \overrightarrow{F_r}\, dt = m\int_{v_{pre}}^{v_{post}} d\vec{v} \tag{3.3}$$

$$\overrightarrow{J_r} = m\,.\left(\vec{v}_{post} - \vec{v}_{pre}\right) \tag{3.4}$$

In the above equations, $\vec{v}_{post}$ and $\vec{v}_{pre}$ refer to the pre-collision and post-collision linear velocities of the particle. The relative contact force acting between the particles (represented by $\overrightarrow{F_r}$) accounts for the total collision force acting on the particle during a binary collision. The impulse vector $\overrightarrow{J_r}$ can be decomposed into the impulse magnitude $'j_r'$ and the unit contact normal of the plane of collision.

$$\overrightarrow{J_r} = j_r.\hat{n} \tag{3.5}$$

In the above expression, $\hat{n}$ is the unit contact normal. Utilizing these expressions, the change in linear velocity for a particle can be expressed as,

$$\vec{v}_{post} = \vec{v}_{pre} + \frac{j_r}{m}.\hat{n} \tag{3.6}$$

Similarly, the relation between the angular velocity and the torque acting on a particle is given as,

$$\vec{T} = \bar{\bar{I}}\,\frac{d\vec{\omega}}{dt} \tag{3.7}$$

$$\int \vec{T}\,dt = \bar{\bar{I}} \int_{\omega_{pre}}^{\omega_{post}} d\vec{\omega} \tag{3.8}$$

The torque acting on a particle can be expressed in terms of the force acting on the particle and the perpendicular distance from the application of the force.

$$\vec{T} = \vec{F_r}\ x\ \vec{r} \tag{3.9}$$

$$\int \left(\vec{F_r}\ x\ \vec{r}\right) dt = \bar{\bar{I}} \int_{\omega_{pre}}^{\omega_{post}} d\vec{\omega} \tag{3.10}$$

$$\vec{J_r}\ x\ \vec{r} = \bar{\bar{I}} * \left(\vec{\omega}_{post} - \vec{\omega}_{pre}\right) \tag{3.11}$$

The change in angular velocity can then be calculated by the expression,

$$\vec{\omega}_{post} = \vec{\omega}_{pre} + j_r\,.\bar{\bar{I}}^{-1} * (\vec{r}\ x\ \hat{n}) \tag{3.12}$$

For a case of binary collision, the translational and rotational velocity for the two particles are thus updated as,

$$\vec{v}_{1,post} = \vec{v}_{1,pre} - \frac{j_r}{m_1}.\hat{n} \tag{3.13a}$$

$$\vec{v}_{2,post} = \vec{v}_{2,pre} + \frac{j_r}{m_2}.\hat{n} \tag{3.13b}$$

$$\vec{\omega}_{1,post} = \vec{\omega}_{1,pre} - j_r\,.\bar{\bar{I}}_1^{-1} * (\vec{r}_1\, x\, \hat{n}\,) \tag{3.13c}$$

$$\vec{\omega}_{2,post} = \vec{\omega}_{2,pre} + j_r\,.\bar{\bar{I}}_2^{-1} * (\vec{r}_2\, x\, \hat{n}\,) \tag{3.13d}$$

In order to make use of these equations to resolve the collisions, the magnitude of impulse ('j'), has to be determined. At the contact point, the pre-collision and post-collision relative velocities are related – along the contact normal - as,

$$\vec{V}_{r,post}\,.\hat{n} = -(COR) * \left(\vec{V}_{r,pre}\,.\hat{n}\,\right) \tag{3.14}$$

where, $\vec{V}_{r,post}$ and $\vec{V}_{r,pre}$ are the relative velocities after and before the instance of collision respectively. The coefficient of restitution (COR) is a numeric quantity that is given by the expression,

$$COR = \sqrt{\frac{Total\ Kinetic\ Energy\ of\ the\ system\ post\ collision}{Total\ Kinetic\ Energy\ of\ the\ system\ before\ collision}} \tag{3.15}$$

Note that 'system' in this case refers to the case of a binary collision under consideration. For a perfectly elastic collision, COR takes the value of unity whereas, for a perfectly inelastic collision, a COR of zero is attributed. The total velocity of a particle at the contact point is given by,

$$\vec{V}_p = \vec{v} + \vec{\omega}\, x\, \vec{r} \tag{3.16}$$

For the two particles in contact their total velocities before collision are,

$$\vec{V}_{p1,pre} = \vec{v}_{1,pre} + \vec{\omega}_{1,pre}\, x\, \vec{r}_1 \quad (3.17a)$$

$$\vec{V}_{p2,pre} = \vec{v}_{2,pre} + \vec{\omega}_{2,pre}\, x\, \vec{r}_2 \quad (3.17b)$$

Subsequently, the particles' total velocities post collision are given by,

$$\vec{V}_{p1,post} = \vec{v}_{1,post} + \vec{\omega}_{1,post}\, x\, \vec{r}_1 \quad (3.18a)$$

$$\vec{V}_{p2,post} = \vec{v}_{2,post} + \vec{\omega}_{2,post}\, x\, \vec{r}_2 \quad (3.18b)$$

The relative velocities before and after the collision are hence determined by,

$$\vec{V}_{r,post} = \vec{V}_{p2,post} - \vec{V}_{p1,post} \quad (3.19)$$

$$\vec{V}_{r,pre} = \vec{V}_{p2,pre} - \vec{V}_{p1,pre} \quad (3.20)$$

Using these expressions for the relative velocities before and after collision, and substituting accordingly in eq.3.14,

$$\left(\vec{V}_{p2,post} - \vec{V}_{p1,post}\right).\hat{n} = -(COR) * \left(\left(\vec{V}_{p2,pre} - \vec{V}_{p1,pre}\right).\hat{n}\right) \quad (3.21)$$

$$\left(\left(\vec{v}_{2,post} + \vec{\omega}_{2,post}\, x\, \vec{r}_2\right) - \left(\vec{v}_{1,post} + \vec{\omega}_{1,post}\, x\, \vec{r}_1\right)\right).\hat{n}$$

$$= -(COR) * \left(\left(\left(\vec{v}_{2,pre} + \vec{\omega}_{2,pre}\, x\, \vec{r}_2\right) - \left(\vec{v}_{1,pre} + \vec{\omega}_{1,pre}\, x\, \vec{r}_1\right)\right).\hat{n}\right) \quad (3.22)$$

But from eq.3.13, the post collision velocities can be expression in terms of the pre-collision velocities as follows,

$$\left(\left(\vec{v}_{2,pre} + \frac{j_r}{m_2}.\hat{n} + \left(\vec{\omega}_{2,pre} + j_r.\bar{\bar{I}}_2^{-1} * (\vec{r}_2 \, x \, \hat{n})\right) x \, \vec{r}_2\right)\right.$$

$$\left. - \left(\vec{v}_{1,pre} - \frac{j_r}{m_1}.\hat{n} + \left(\vec{\omega}_{1,pre} - j_r.\bar{\bar{I}}_1^{-1} * (\vec{r}_1 \, x \, \hat{n})\right) x \, \vec{r}_1\right)\right).\hat{n}$$

$$= -(COR) * \left(\left(\left(\vec{v}_{2,pre} + \vec{\omega}_{2,pre} \, x \, \vec{r}_2\right) - \left(\vec{v}_{1,pre} + \vec{\omega}_{1,pre} \, x \, \vec{r}_1\right)\right).\hat{n}\right) \tag{3.23}$$

Simplifying this further and using the expression for relative velocities pre-collision (eq.3.17 and eq.3.20), the magnitude of impulse is determined by the following expression.

$$j_r = \frac{-(1 + COR) * \left(\vec{V}_{r,pre}.\hat{n}\right)}{\frac{1}{m_1} + \frac{1}{m_2} + \left(\left(\left(\bar{\bar{I}}_1^{-1} * (r_1 \, x \, \hat{n})\right) x \, \vec{r}_1\right) + \left(\left(\bar{\bar{I}}_2^{-1} * (r_2 \, x \, \hat{n})\right) x \, \vec{r}_2\right)\right).\hat{n}} \tag{3.24}$$

Once the magnitude of impulse is determined, the post collision velocities can be arrived at by evaluating the formulations given in eq.3.13.

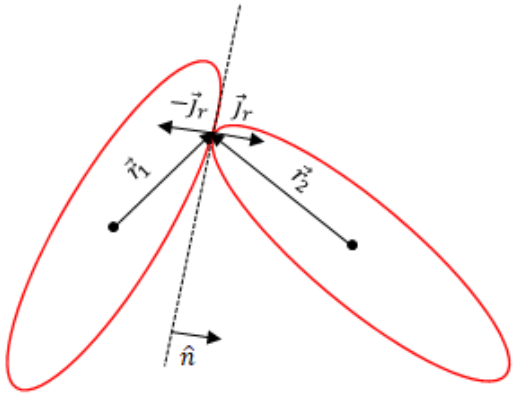

Figure 3-1 : Collision resolution representation by impulses for particle-particle collision

3.1.2. Frictional Impact

In reality, collisions between particles involve tangential kinematics (friction forces). When friction forces act on particles, there is a loss in kinetic energy. The loss in kinetic energy can also be accounted for by modifying the COR magnitude in the friction-less impact modeling. However, in order to replicate the actual kinematics and subsequent update of translational and rotational velocities, frictional impact modeling is required. One of the advantages of employing an impulse-based resolution model is that, it allows for the incorporation of tangential kinematics in a manner similar to the friction-less (normal) impact. In this framework, a Coulomb friction model is implemented [37]. In a coulomb friction model, the friction forces can be illustrated by the use of static and dynamic friction forces. Static friction assumes that when a body is in contact with another body, the forces acting tangentially to each other would be equal and opposite resulting in both the bodies being stationary. When the external forces are large enough, the static friction force is no longer sufficient to maintain the stationary state and thus, gives way to the dynamic friction force ensuing in the motion of the two bodies. The friction forces could be visualized using the Coulomb friction cone, which implies that the reaction force acting on the body - in this case, the cone - in contact with a surface would result in a tangential dynamic motion if the magnitude of friction force falls outside the cone (Figure 3-2). The direction of application of the impulses on the particles is referenced by the unit tangent vector acting along the collision plane.

$$\hat{t} = \frac{\vec{V}_{r,pre} - (\vec{V}_{r,pre} \cdot \hat{n})\hat{n}}{\left|\vec{V}_{r,pre} - (\vec{V}_{r,pre} \cdot \hat{n})\hat{n}\right|} \tag{3.25}$$

Similar to frictionless impact a tangential impulse vector is defined based on the tangential impulse magnitude. The formulation of the tangential kinematics is very similar to the equivalent

model for spheres with the collision forces replaced by the collision impulses for non-spherical particles. The Coulomb friction model elucidates that the frictional model consists of the static and the dynamic forces. These magnitude of these forces can be represented by the contact tangent and their corresponding coefficients, the mathematical representation of which is shown below.

$$f_s = \mu_s \, . f_r \tag{3.26}$$

$$f_d = \mu_d \, . f_r \tag{3.27}$$

In the above equations, μ_s and μ_d refer to the static and dynamic friction coefficients. $f_r = \left|\overrightarrow{F_r}\right|$ represents the magnitude of the resulting reaction force magnitude during collision. These formulations are developed with the consideration that $\mu_s > \mu_d$. Integrating eqs.3.26 and 3.27 for the time of collision, results in the following equivalent expressions for the impulses.

$$\int_{t_{begin}}^{t_{end}} f_s \, dt = \mu_s \, . \int_{t_{begin}}^{t_{end}} f_r \, dt \tag{3.28}$$

$$j_s = \mu_s \, . j_r \tag{3.29}$$

Similarly, for the dynamic component of the total frictional impulse,

$$\int_{t_{begin}}^{t_{end}} f_d \, dt = \mu_d \, . \int_{t_{begin}}^{t_{end}} f_r \, dt \tag{3.30}$$

$$j_d = \mu_d \, . j_r \tag{3.31}$$

In these expressions, 't_{begin}' and 't_{end}' point to the temporal start and end of the collision instance with 'j_r' representing the magnitude of the normal contact impulse. Using these expressions and the unit contact tangent, the frictional impulse vector can be defined as,

$$\overrightarrow{J_f} = \begin{cases} \left(-m\,\vec{V}_{r,pre}\,.\,\hat{t}\;\right)\hat{t} & if\;\; \vec{V}_{r,pre}\,.\,\hat{t} = 0 \\ -j_d\,.\,\hat{t} & otherwise \end{cases} \tag{3.32}$$

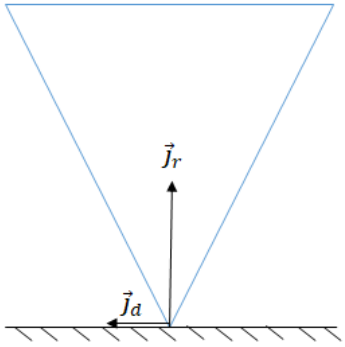

Figure 3-2 : Friction cone representing the Coulomb friction model

The impulse-based Coulomb friction model assumes that the static frictional impulse ensures the particles do not slide against each other as long as the magnitude of external forces is small enough. If the magnitude of external forces are substantial, the static frictional impulse gives way to the dynamic frictional impulse and the particles begin sliding against each other. In the developed framework, it is assumed that the particles do not deform during collisions but instead, slide against each other as soon as they are in contact. Hence eq.3.26, eq.3.28 and eq.3.30 have been neglected in the implementation. Also, the first relation of eq.3.32 has been neglected as it deals with stationary deformation of particles. Following the computation of the frictional impulses, the particle trajectories are stipulated by updating their velocities. It has to be noted that the post collision velocities ensuing from the frictionless impact dynamics are updated by the frictional impulse computed.

$$\vec{v}_{1,post} = \vec{v}_{1,post} - \frac{j_d}{m_1}.\hat{t} \tag{3.33a}$$

$$\vec{v}_{2,post} = \vec{v}_{2,post} + \frac{j_d}{m_2}.\hat{t} \tag{3.33b}$$

$$\vec{\omega}_{1,post} = \vec{\omega}_{1,post} - j_d \,.\, \bar{\bar{I}}_1^{-1} * (\vec{r}_1 \; x \; \hat{t}\,) \tag{3.33c}$$

$$\vec{\omega}_{2,post} = \vec{\omega}_{2,post} + j_d \,.\, \bar{\bar{I}}_2^{-1} * (\vec{r}_2 \; x \; \hat{t}\,) \tag{3.33d}$$

3.2. Particle-Particle Soft Sphere Modeling

In soft sphere modeling, particles are allowed to deform during collisions. Inherently, a soft sphere model is time consuming when compared to a hard sphere model because of the resolution of collisions in successive time steps. This implies that the collision information and the collision resolution technique is repeated over a number of time steps to completely resolve the collisions. The soft sphere model is, in many ways, replicative of the real-life scenarios when elastic bodies collide with each other. The model assumes that when a particle deforms, the forces acting on the particle due to collision can be modeled as a spring-dashpot system. These forces are then used to resolve the collisions using Newton's laws of motion. The soft sphere model is beneficial when applied to multiple particles in the computational domain as it resolves multiple contacts on the particles' surfaces, unlike the hard sphere model, which resolve multiple simultaneous collisions sequentially. It is formulated based on the change in momentum effected by the reaction forces acting on the particle. Newton's laws of motion stipulate that the change in momentum is given by,

$$m\frac{d\vec{v}}{dt} = \sum_{j=1}^{n} \vec{F}_{c,j} + \vec{F}_f + \vec{F}_g \tag{3.34}$$

where, $\vec{F}_{c,j}$ represents the collision forces acting on a particular particle due to one of its nearest colliding neighbor 'j'. The total collision force is the discrete summation of all the reaction forces acting on that particle, effected by all of its colliding neighbors - the maximum number of which

is denoted as 'n'. Particle collisions with walls could also contribute to the total collision force, in which case the eventual force vector is added to the total particle-particle collision force vector. $\vec{F}_f$ and $\vec{F}_g$ represent the fluid forces and the gravitational force acting on the particle. An essential element contributing to the temporal advancement of velocities that has been neglected in the formulation of the forces is the total force arising from non-colliding neighbors. This particular force is the consequence of disturbance waves from neighboring particles, at a substantial distance from the particle in consideration. In majority of the models, this term is omitted from the calculations as the magnitude of the force turns out to be less in comparison to the other forces if the numerical time step is small enough.

A linear soft sphere collision model consists of normal force components and tangential force components[38]. Theoretically, the normal and tangential force components are assumed to act over a contact area due to the nature of the collision. In the case of a sphere, the normal force is used to calculate the post collision linear velocities and the tangential force - acting along the contact plane - gives rise to a torque which subsequently imparts rotation to the particles. However, in the case of non-spherical particles, the summation of the normal and the tangential forces constitute the total force, which is used to update the linear velocities post collision. The total force additionally is utilized to calculate an effective torque which is then employed in the update for rotational velocities. The linear model prescribes that the reaction force along normal and tangential directions should proportionately increase with the penetration depth between particles. The normal and tangential components of the normal collision force consist of a 'restoration' force governed by a stiffness coefficient and a 'damping' term governed by a dashpot. A characteristic representation of a linear spring-dashpot time-driven collision resolution mechanism is shown in Figure 3-3.

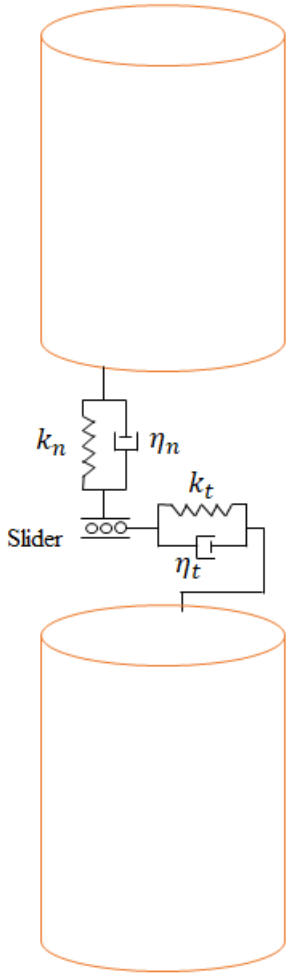

Figure 3-3 : Representation of particle-particle soft sphere collision model

The normal force acting on the particle during collision is expressed as follows,

$$\vec{F}_n = -k_n \vec{\delta}_n - \eta_n \vec{v}_n \tag{3.35}$$

where, k_n denotes the normal stiffness coefficient and η_n represents the normal damping coefficient. In order to carry out the soft sphere collision dynamics, the stiffness coefficient has to be pre-defined and should be provided as an input to the model. $\vec{\delta}_n$ denotes the relocation vector

that is calculated based on the maximum magnitude of penetration along the contact normal. It follows a similar estimation to that of the maximum magnitude of penetration calculations shown in section 2.4.5. From eq.2.33, the maximum magnitude of penetration along the normal can be obtained and subsequently, the relocation vector is expressed as,

$$\vec{\delta}_n = \delta_{max} \,.\, \hat{n} \tag{3.36}$$

The relative normal velocity, depicted as '$\vec{v}_n$' is the relative velocity of the colliding particles along the contact normal. The vector is arrived at by the following expression,

$$\vec{v}_n = \left(\vec{V}_{r,pre} \,.\, \hat{n}\right)\hat{n} \tag{3.37}$$

where, $\vec{V}_{r,pre}$ as described in eq.3.20 is the relative total velocity of the concerned particles before collision. '$\hat{n}$' refers to the contact normal perpendicular to the plane of contact between the two particles.

The damping coefficient can be approximated by an expression involving the stiffness coefficient and the reduced masses of the particle as described in Navarro and Souza[39].

$$\eta_n = 2 * \gamma * \sqrt{m_{red} * k_n} \tag{3.38}$$

where, 'm_{red}' refers to the reduced mass of the two colliding particles. Reduced mass can be thought of as an 'effective' mass quantity that influences the reaction force computation. The reduced mass term can be computed as,

$$m_{red} = \frac{m_1 m_2}{m_1 + m_2} \tag{3.39}$$

In the instance of a binary collision, m_1 and m_2 represent the masses of the two particles. In generic terms, the reduced mass combines the masses of the 'i^{th}' and 'j^{th}' particles. In the majority of cases simulated in this research work, all the particles have the same mass. Hence, the reduced mass during a collision is simply half the mass of one of the two colliding particles.

Since the damping coefficient is responsible for reduction in energy of the system, the coefficient of restitution – a measure of the energy before and after collision – has to be incorporated in the calculations. This is brought about in the form of 'γ' which can be expanded as,

$$\gamma = -\frac{ln(COR)}{\sqrt{\pi^2 + (ln(COR))^2}} \tag{3.40}$$

A COR value of 1 translates into a collision with no loss of energy. In the expression shown above, the theoretical interpretation is reproduced accurately as the 'γ' term evaluates to zero which in turn reduces the damping coefficient down to zero accordingly.

Similarly, the auxiliary component of the total collision force is the tangential force acting on the particles along the contact plane. The tangential force is expressed almost identical to the normal component, as follows,

$$\vec{F}_t = -k_t \vec{\delta}_t - \eta_t \vec{v}_t \tag{3.41}$$

where, k_t denotes the tangential stiffness coefficient and η_t represents the tangential damping coefficient. Akin to the normal force components, the tangential stiffness coefficient has to be pre-defined and should be provided as an input to the model. $\vec{\delta}_t$ denotes the relocation vector along the tangential direction. A noticeable difference between the normal and tangential force components is the method employed in the calculation of the respective relocation vectors. While

the normal relocation vector is calculated based on the maximum magnitude of penetration along the contact normal, the tangential relocation vector has to be integrated over the total collision time.

$$\vec{\delta}_t^{\,n} = \vec{\delta}_t^{\,n-1} + \vec{v}_t \,.\, \Delta t \tag{3.42}$$

where, 'n' denotes the time step of collision. The numeric value of 'n' starts from one – representing the first instance of collision - and accrues to a maximum of the collision time. However, such a computation involves the storage of the relocation vector for consecutive time steps - and in most simulations, for a large number of particles - which usually leads to increased book-keeping and memory use. Hence, an approximation is made to the tangential relocation vector [40] as shown below.

$$\vec{\delta}_t = \tau_c \,.\, \vec{v}_t \tag{3.43}$$

where, 'τ_c' represents the time of collision. The approximation is formulated on the basis of similar expressions derived in Tsuji et al [41] as detailed below,

$$\tau_c = \frac{\pi}{\sqrt{k_n \,(1-\gamma^2\,)/m_{red}}} \tag{3.44}$$

The approximation ensures that the memory resources used are less, without a significant loss in accuracy. The tangential damping coefficient is calculated similar to the normal damping coefficient, except that the normal stiffness coefficient is replaced by the tangential stiffness coefficient.

$$\eta_t = 2 * \gamma * \sqrt{m_{red} * k_t} \tag{3.45}$$

The calculation of 'γ' remains the same as it is assumed that the tangential and normal COR are the same. The unit tangent vector computed for the hard-sphere frictional model is based off the relative tangential velocity vector before collision. All the constituent terms in the tangential velocity expression shown below, are calculated in the exact same manner as described in the hard sphere modeling (section 3.1.2).

$$\vec{v}_t = \vec{V}_{r,pre} - \left(\vec{V}_{r,pre} \,.\, \hat{n}\right)\hat{n} \tag{3.46}$$

Additionally, the tangential force calculations assume that there is sufficient tangential deformation. In certain cases, the penetration depth could be relatively high along the tangential direction, which could lead to high magnitudes of tangential forces. This might result in unphysical scenarios where the particles might seemingly gain energy. Hence, a limit is imposed on the tangential forces. In physical terms, the limit ensures that the particles do not deform tangentially to a large extent and instead, move into the sliding friction regime. The conditional limit is formulated as,

$$\vec{F}_t = \begin{cases} -k_t\,\vec{\delta}_t - \eta_t \vec{v}_t & if\ \left|\vec{F}_t\right| < \mu\left|\vec{F}_n\right| \\ -\mu\left|\vec{F}_n\right| . \dfrac{\vec{v}_t}{|\vec{v}_t|} & otherwise \end{cases} \tag{3.47}$$

A further realistic model would be to account for the non-linear mechanics in the spring-dashpot model. The mathematical formulation for a non-linear system has the forces progressively increasing in non-linear trends with increasing deformation between particles [5], [42]. For the scope of this study, we will use linear models.

The total collision force acting on the particle is given by the summation of the normal and tangential contact forces.

$$\vec{F}_c = \vec{F}_n + \vec{F}_t \tag{3.48}$$

An update in the linear velocities is carried out using the total collision force acting on the particle's surface area. DEM models employing spheres require only the normal force to update the linear velocities of the particles. The developed framework has a clear distinction when compared to spherical DEM models in the fact that the total collision force comprises of the normal and tangential contact forces.

Another aspect of spherical DEM models is that the torque calculations constitutes only the tangential forces, since the normal forces act along the line joining the contact point and the center of mass. However, in non – spherical particles, the normal forces might not necessarily be acting along the above mentioned 'line'. As a consequence, the total force has to be taken into account when calculating the net torque acting on the particle's surface. Generally, the rotational velocity is updated as,

$$\bar{\bar{I}} \frac{d\vec{\omega}}{dt} = \sum_{j=1}^{n} \vec{T}_{c,j} + \vec{T}_f + \vec{T}_{external} \tag{3.49}$$

where, $\bar{\bar{I}}$ is the inertia tensor of the particle. $\vec{T}_f$ and $\vec{T}_{external}$ are the torques imparted by the fluid and external sources. The external sources include the effect of particles not in the zone of influence of the concerned particle. Typically, the zone of influence could be visualized as an imaginary sphere constructed with its radius equal to the maximum dimension of the particle. If the velocities are defined in global space, the inertia tensor is also defined in global space to be consistent with the frames of reference. The torque vector $\vec{T}_{c,j}$ is the cumulative torque acting on a particle due to its colliding neighbors. For a given binary collision, the torque is calculated as,

$$\vec{T}_c = \vec{r}\, x\, \vec{F}_c \tag{3.50}$$

In the above equation, $\vec{r}$ refers to the vector - in global space – between the contact point and the center of mass of the particle. The total force is assumed to act perpendicular to this vector, resulting in a torque (or) moment acting on the particle's surface. The total force and the torque computed subsequently are used to update the linear and rotational velocities of the particles in addition to the fluid, external and gravitational forces. The soft sphere model is computationally intensive as the particle collisions are resolved in multiple time steps. However, the model guarantees a stable solution, especially in dense particulate flows and in packing/piling simulations. Also since the model involves the use of forces to estimate particle trajectories as opposed to the impulse resolution model employed in hard sphere model, the soft sphere model is readily compatible with CFD solvers.

The limitation on the time step is imposed by the physical parameters affecting the reaction force calculations. Since the particles' collisions with other particles or the wall are resolved over multiple time steps, there is a possibility that the reaction force calculated may turn out to be insufficient to exert an influence on the dynamics of the particle in the precise manner. On that account, it is quite possible to encounter scenarios where the particles move beyond the computational domain during collision with walls. In other instances, the particles might move through the entire volume of one another, resulting in an erroneous representation of the particle dynamics. To circumvent this issue, as a rule-of-thumb, the numerical time steps of the particles is kept about fifteen times lower that the time of collision as given in Muller et al. [43]. The estimated collision time for the particles can be calculated as,

$$\Delta t_{min} = \frac{\pi}{\sqrt{\frac{k_n}{m_{red}} - \left(\frac{\eta_n}{2\,.\,m_{red}}\right)^2}} \tag{3.51}$$

The ratio of the minimum collision time to the numerical time step is expressed as,

$$\Delta t_{ratio} = \frac{\Delta t_{min}}{\Delta t} \tag{3.52}$$

If the ratio is less than 15, the simulation time has to be reduced. Another solution would be to reduce the stiffness coefficient or increase the mass although, modifying the physical parameters of the system is not possible in most cases. Therefore, the soft sphere model is inherently more computationally intensive as it imposes a restriction on the time step.

3.3. Particle-Wall Collision Resolution

Particle-wall collision resolution using the hard sphere and the soft sphere models closely follow the procedure outlined for the respective particle-particle collision resolution, with the simplifications implied in section 2.5. Appendix A summarizes the procedures used for particle-wall collision resolution with the hard and soft sphere models.

Chapter 4

4. Results and Discussion

The implementation details of the non-spherical particle treatment in the flow solver GenIDLEST- an in-house fluid solver developed by Tafti[44] - is given in Appendix B. Verification and validation of the developed collision model for non-spherical particles is carried out in this chapter. In addition, since the cost of simulation is an area of concern for non-spherical particles, some measures of performance - namely, the additional overhead of a non-spherical collision model over a spherical particle is discussed, and the parallel efficiency and speed up as compared to a serial computation is given. Model verification is performed by monitoring the total energy of a system of colliding non-spherical particles. Model validation is done by using the non-spherical particle collision model and comparing the results to a spherical collision model in a fluidized bed.

4.1. Verification and Validation of the Collision model

4.1.1. Energy Conservation

Under ideal conditions, when particles collide, the total energy of the system has to be conserved. For a physically consistent impact model, there has to be no additional sink or source term of energy acting on the particles. In time-driven collisions, an increase in spring energy (potential) is accompanied by a decrease in the kinetic energy of the particles during collision and vice versa, for a perfectly elastic collision. For particulate matter, the total energy of the system accounts for the potential and kinetic energy of particles. For a given particle, its potential energy follows the familiar expressions for objects influenced by gravity and is given by,

$$PE = mgh \tag{4.1}$$

In the above equation, 'm' refers to the mass of the particle, 'g' represents the acceleration due to gravity (9.81 m/s) and 'h' represents the vertical displacement of the particle from ground (or) base. The total kinetic energy of a non-spherical particle undergoing translation and rotation has components dependent on its linear and rotational velocities. Since the total energy of the system is represented in global space, the rotational and translational energy have to be accordingly expressed in terms of the global inertia tensor and velocities.

$$KE = TKE + RKE \tag{4.2}$$

$$TKE = 0.5 * m \,.\, |\vec{v}|^2 \tag{4.3}$$

$$RKE = 0.5 * \{(I_{xx} * \omega_x{}^2) + (I_{yy} * \omega_y{}^2) + (I_{zz} * \omega_z{}^2) + (2 * [(I_{xy} * \omega_x * \omega_y) + (I_{yz} * \omega_y * \omega_z) + (I_{xz} * \omega_x * \omega_z)])\} \tag{4.4}$$

In the above expressions, TKE and RKE refer to the translational kinetic energy and rotational kinetic energy of the given particle. As described in the preceding paragraphs, the total energy is the summation of the scalar magnitudes of potential energy, translational kinetic energy and the rotational kinetic energy.

$$TE = PE + TKE + RKE \tag{4.5}$$

The total energy of the system is verified for test cases consisting of particles undergoing collisions with each other and the wall boundaries. Both event-driven and time-driven models are employed to resolve the collisions and the total energy is monitored for the duration of the

simulation. In the event-driven model, the formulation for frictionless impact (sections 3.1.1) are used because the inclusion of friction would result in a loss of energy inherently. The time-driven model however does not need to be modified as the damping factor reduces to zero for a perfectly elastic collision. The verification of both models are carried out at two different coefficients of restitution (COR) – specifically, 0.4 and 1.0 - for particle-particle collisions and particle-wall collisions. The verification tests are done for binary collisions and a multi-particulate system undergoing translational and rotational motion. In any collision model, the emphasis is on change in kinetic energy of particles through the collision mechanics. Hence, the potential energy is neglected (gravity is not activated in the simulations depicted here). Also, the DEM models are uncoupled from the CFD solver and hence, the particles are not influenced by the fluid.

To reinforce the energy-conservative property [21], [36] of impulse based event-driven model, simulations involving collisions are carried out as depicted in Figure 4-1 and Figure 4-4. In these cases, the COR of particle-particle collision and particle-wall collision is set to a value of 0.4 and 1.0; correspondingly, the total kinetic energy of the system is observed. The particles are supplied with an initial velocity equal in magnitude and opposite in direction. The velocity magnitudes are normalized (U_ref) by the reference value of 1 m/s. They are allowed to collide with each other and with the wall subsequently. While Figure 4-1 represents the collision mechanics at a specified COR of 0.4, Figure 4-4 depicts that at a specified COR of 1.0. Figure 4-2 and Figure 4-5 illustrate the temporal progression of total kinetic energy of the system. The magnitude of total kinetic energy has been normalized by the initial kinetic energy to better portray the change in energy at the lower COR. With the specified initial velocities, the particles collide with each other at 0.09 s, approximately. Figure 4-2 shows a decrease in total kinetic energy of the system from an initial magnitude of unity to a magnitude of 0.16 (approximately). The square root

of the ratio of the final and initial magnitude of energy, at 0.09 s, evaluates to 0.4, which corresponds to the specified value of COR. From Figure 4-5, at the same time instance (0.09 s), there is no change in the total kinetic energy of the system, which validates the functioning of the model at a COR of 1.0.

Another validation strategy by which the specified COR of collisions could be cross verified is by monitoring the change in velocities. For a case of binary collision wherein particles exhibit only translation motion along one axis, the ratio of post collision velocity to pre collision velocity equates to the COR. In Figure 4-1, the particles start with an initial non dimensional velocity magnitude of 2 units, which decreases to 0.8 non-dimensional units following the initial collision at 0.09 s. The ratio of the post collision and pre collision velocity magnitude in this case equates to 0.4. Furthermore from Figure 4-4, it is discernible that the magnitude of post collision velocity stays the same as that of pre collision velocity, implying the collisions are perfectly elastic as expected at a COR of 1.0.

Similar to the event-driven model, the time-driven model's collision mechanics are governed by prescribing the COR. A decrease in the value of COR effects an alteration to the damping coefficient, which in turn, results in a decrease in energy of the system. Validation tests using the time-driven collisions models have been carried out at COR values of 0.4 and 1.0. These simulations are similar to those carried out using the hard sphere model. Hence, Figure 4-1 also represents the temporal progression of the particles undergoing collisions with each other and the wall – resolved by the time-driven model - at a COR of 0.4. Figure 4-3 shows the change in the normalized total kinetic energy of the system with time. At 0.09 s, it can be observed that the magnitude of total kinetic energy reduces to 0.16 from an initial value of unity, as governed by a COR of 0.4.

The time evolution of particles undergoing collisions with each other and the wall at a specified COR of 1.0 has been portrayed in Figure 4-4 (since the dynamics are very similar to the event-driven model). Additionally, Figure 4-6 depicts the normalized total energy for the above mentioned case. Analogous to the event-driven model, the time-driven collision model calculates the post collision velocities in accordance with conservation of energy and momentum, as the magnitude of the total kinetic energy remains constant with time. It can be observed from these results that the event-driven and time-driven models are valid models to resolve collisions between non-spherical particles.

The simulations described in the previous paragraphs consist of ellipsoidal particles with linear velocity acting along one dimension. While these cases verify the total energy and momentum for a rudimentary scenario, a non-spherical particle usually undergoes much more complicated kinematics. To verify the conservation of energy for a multi-particulate system of particles exhibiting translation and rotational motion, a simulation was carried out with 50 cylinders in a wall-bounded domain. The cylinders are assigned an initial linear velocity with components normalized by ‘U_ref’(1 m/s) and acting along all three dimensions. As they collide with the wall and other cylinders, they acquire rotational velocities. Since the COR of particle-particle collisions and particle-wall collisions is unity, in this case, the total kinetic energy of the system is expected to be constant for the entire duration of the simulation. Figure 4-7, shows the spatial and temporal advancement of the cylinders and Figure 4-8 corresponds to the normalized total kinetic energy of the system varying with time. It can be observed that although the particles undergo both translational and rotational motions when colliding with each other and the wall boundary, the total kinetic energy of the system remains constant as specified.

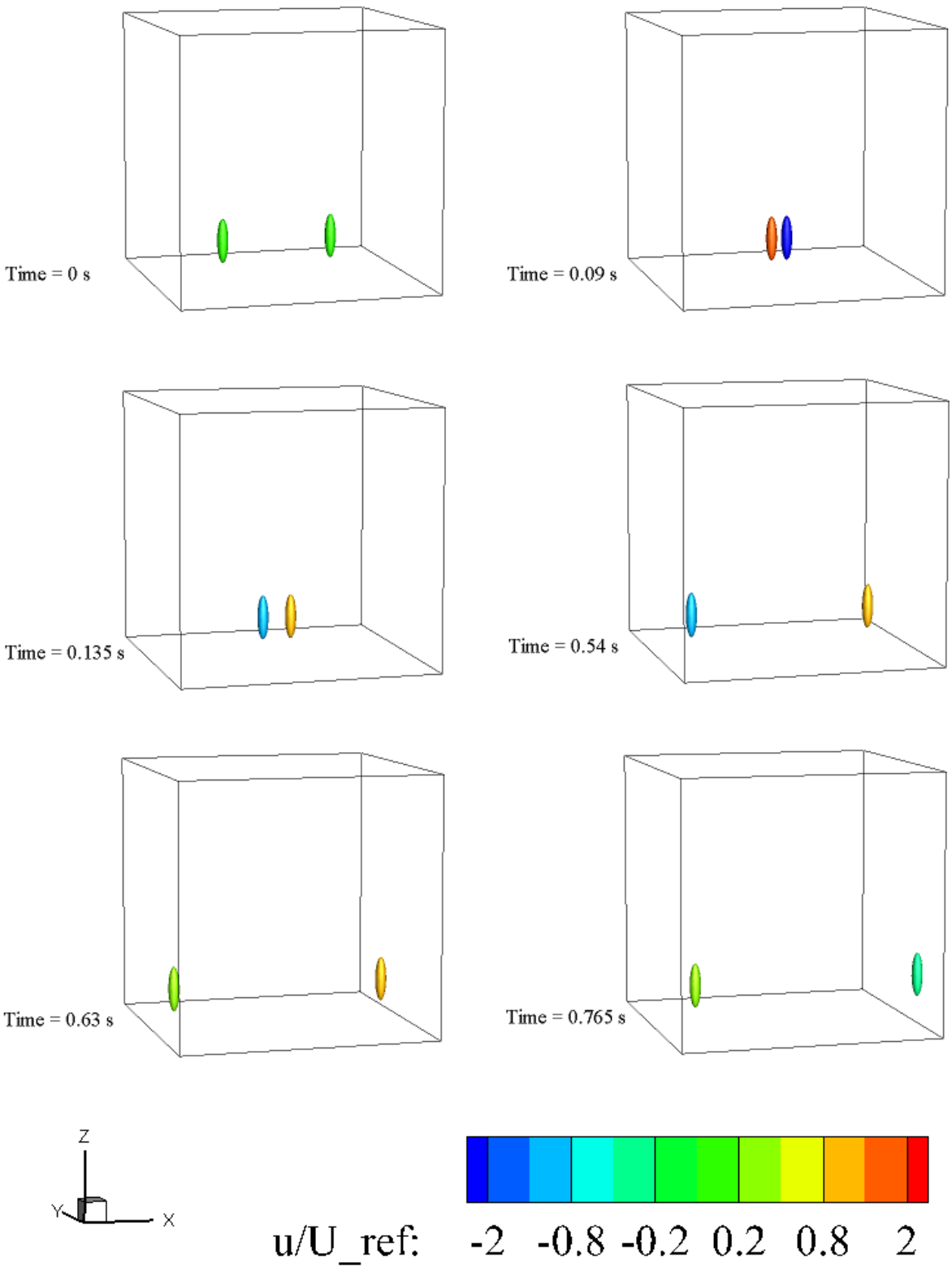

Figure 4-1 : Collisions between two ellipsoids and the wall boundary at a COR of 0.4

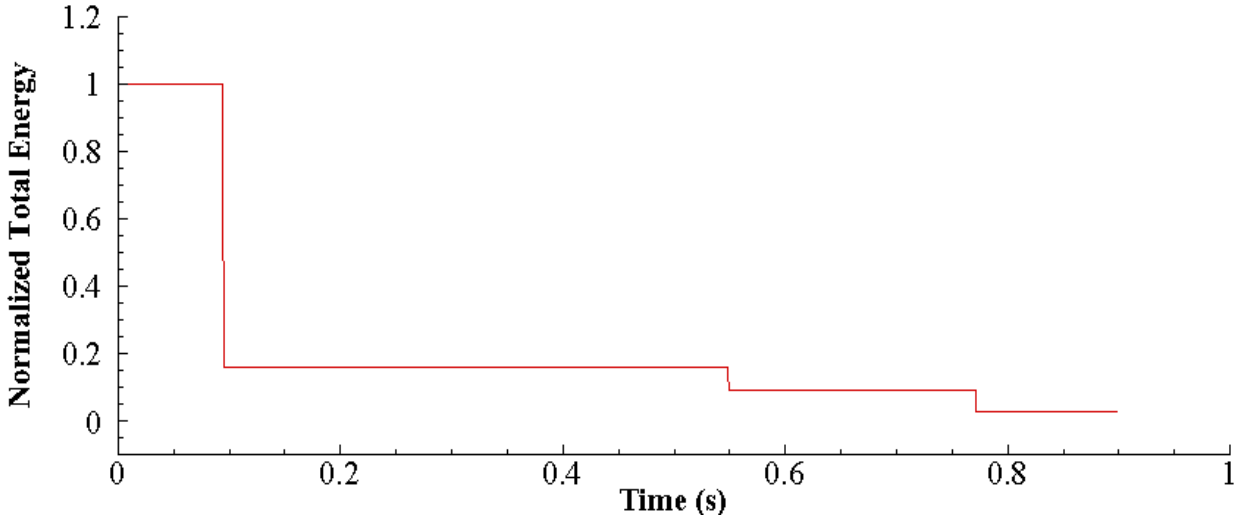

Figure 4-2 : Temporal progression of the total kinetic energy of the system with ellipsoids undergoing collisions modeled using Hard Sphere model at a COR of 0.4

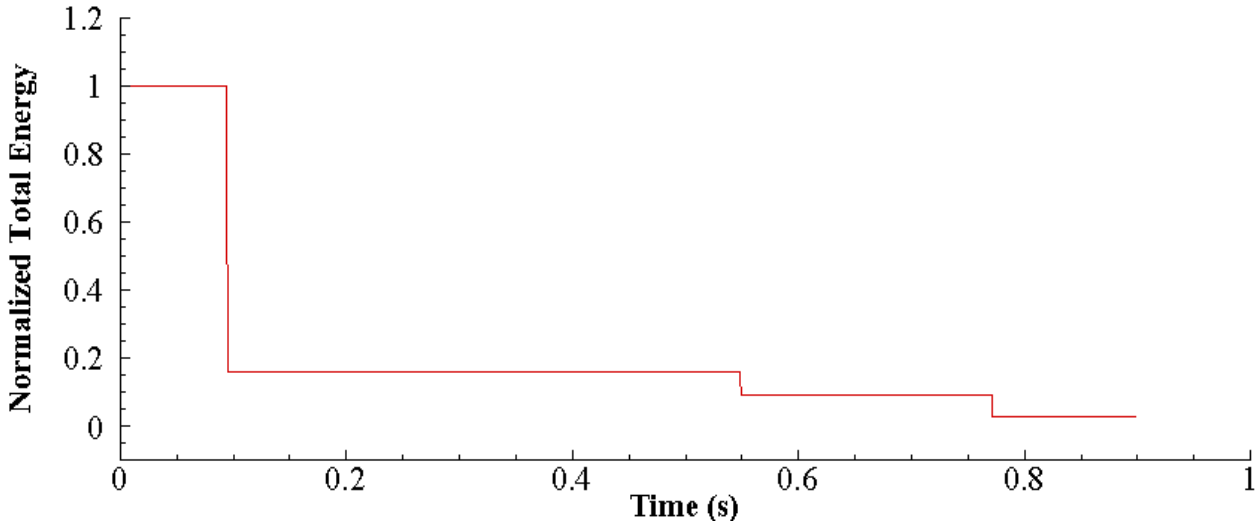

Figure 4-3 : Temporal progression of the total kinetic energy of the system with ellipsoids undergoing collisions modeled using Soft Sphere model at a COR of 0.4.

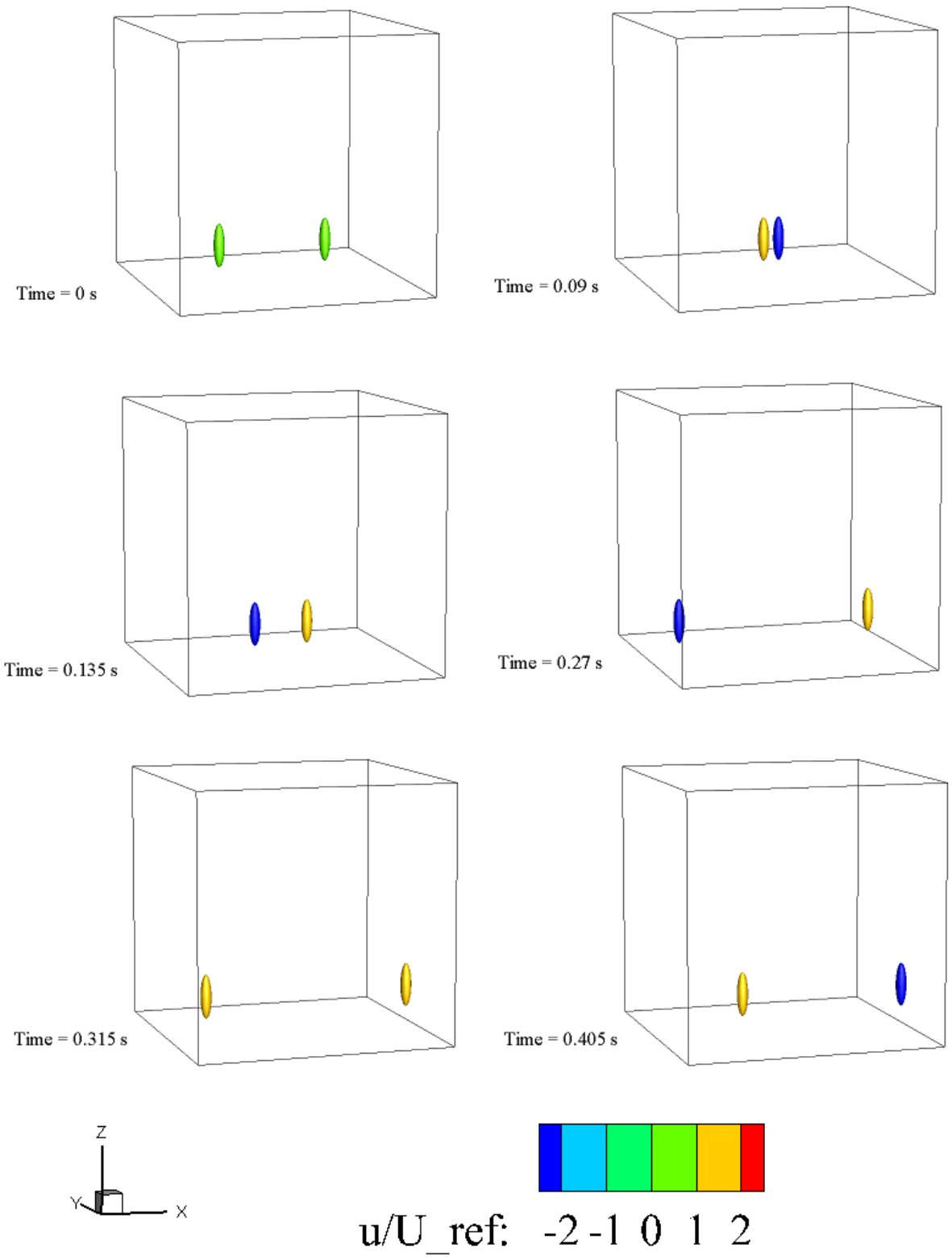

Figure 4-4 : Collisions between two ellipsoids and the wall boundary at a COR of 1.0

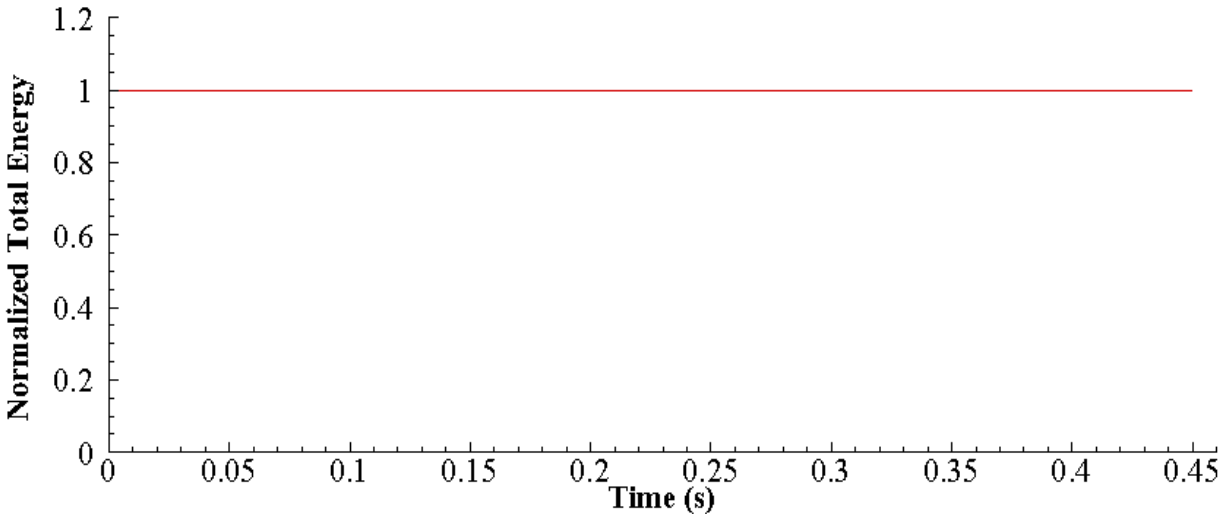

Figure 4-5 : Temporal progression of the total kinetic energy of the system with ellipsoids undergoing collisions modeled using Hard Sphere model at a COR of 1.0

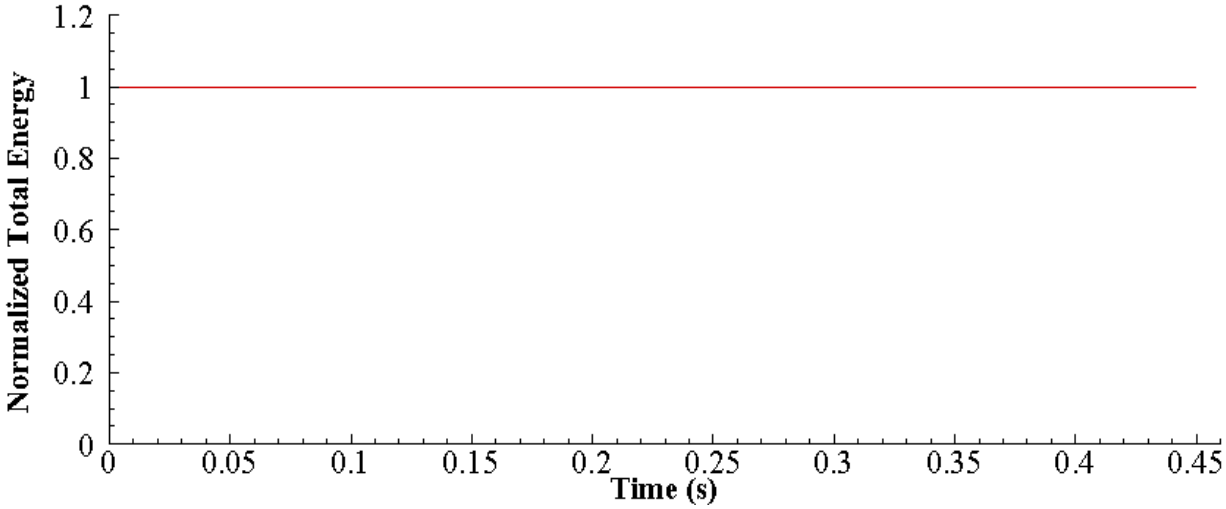

Figure 4-6 : Temporal progression of the total kinetic energy of the system with ellipsoids undergoing collisions modeled using Soft Sphere model at a COR of 1.0.

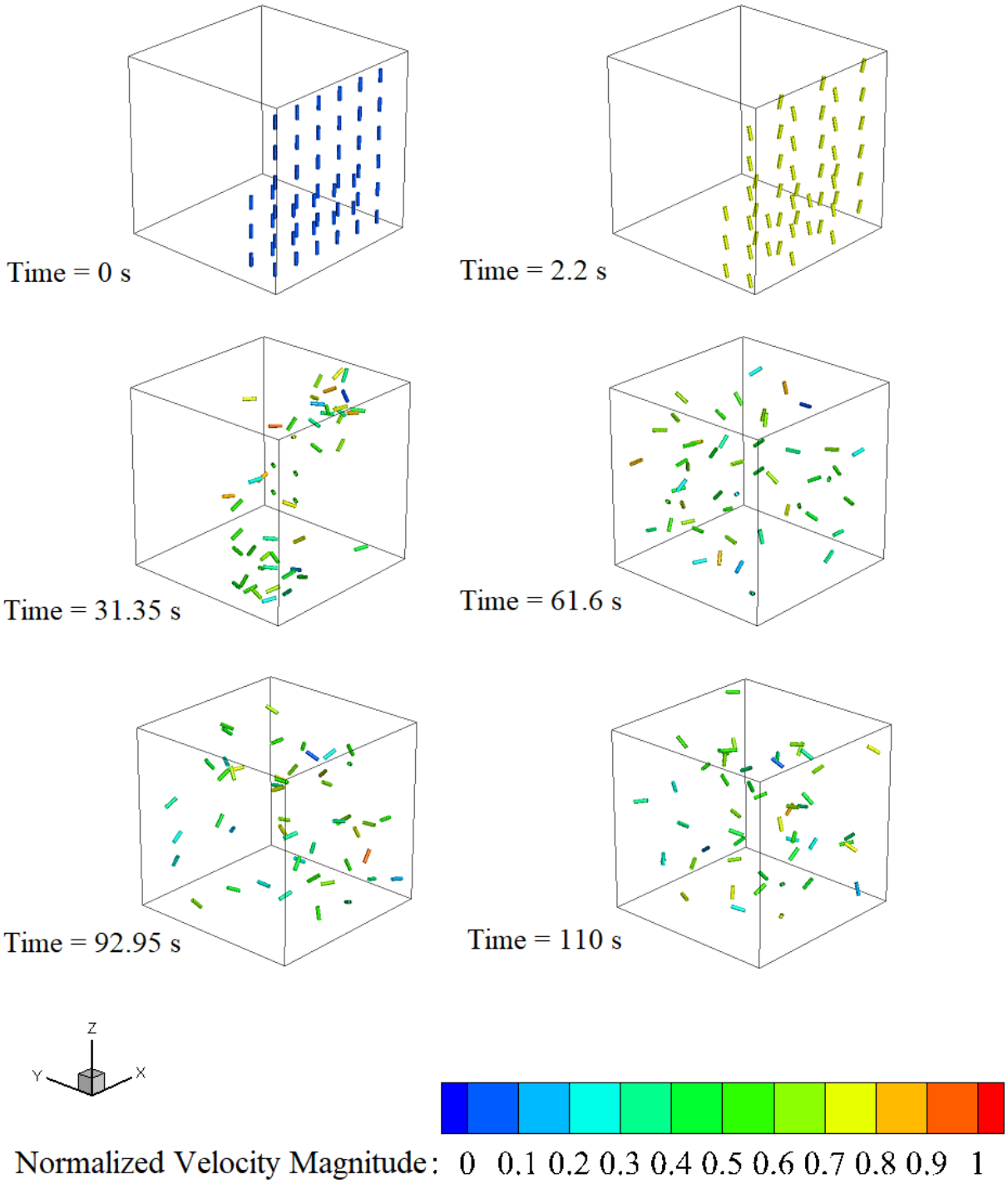

Figure 4-7 : Collisions in a cylindrical multi - particulate system modeled using Hard Sphere model at a COR of 1.0

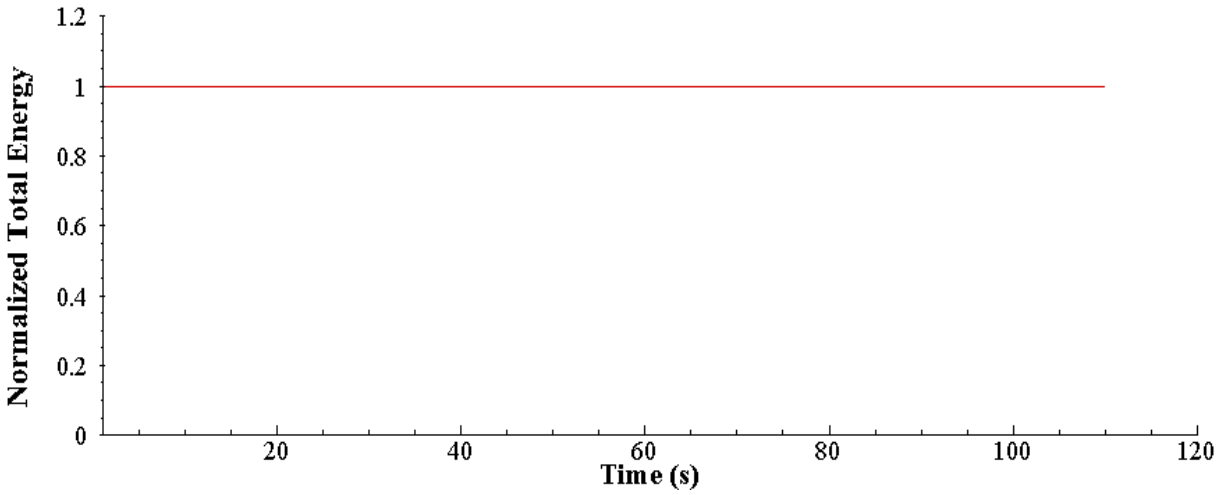

Figure 4-8 : Temporal progression of the total kinetic energy of the multi - particulate system undergoing collisions modeled using Hard Sphere model at a COR of 1.0.

4.1.2. Validation with Spherical Particle CFD-DEM and Experiments

The tests described in the previous section had been carried out in vacuum. The primary purpose of the DEM modeling model for non-spherical particulate systems is to couple the model with a fluid solver. In GenIDLEST, there are existing formulations to solve for the particle kinematics of spherical particles based on the collision and fluid forces. Since spheres can also be generated by discrete function representation of the superellipsoid surface function, the validation of the collision model has been performed in a fluidized bed which includes fluid forces such as drag and buoyancy for spherical particulate systems.

The experiments conducted by Muller et al.[43] on a fluidized bed have been employed in these validation tests. The fluidized bed measures 44 mm long, 160 mm high and has a 10 mm thickness. The simulated case replicates a superficial velocity ('U_f') of 0.9 m/s which is three times the minimum fluidization velocity for the particles. The computational domain used in these validation tests consist of 12 x 32 x 2 cells along the x, y, and z directions (x, y, z represent the

horizontal, vertical and lateral axes, respectively). Figure 4-9 provides an illustration of the computation domain generated. The walls along x and z directions were set to no – slip conditions. The top wall is set to an out flow boundary condition whereas the bottom wall represents a permeable surface with a vertical inlet velocity whose magnitude equals the fluidization velocity. In the experiments poppy seeds were used as granular particles. The properties of the particles are shown in Table 4-1. 9240 particles are initialized in the domain and their initial velocities are set to the superficial velocity (U_f = 0.9 m/s). Particles gain potential energy due to the initial velocities but they soon decelerate due to gravity and drag forces acting on their surfaces. As they begin to decelerate, they lose their upward momentum and fall along the sides of the domain. The fluid medium acting on the particles is air and the concerned properties are given in Table 4-2.

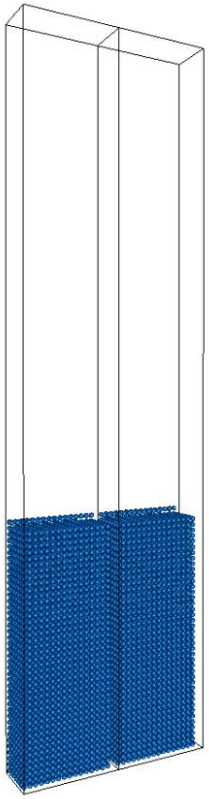

Figure 4-9 : Computational domain generated for validation tests

Table 4-1 : Physical properties of the particles

Particle Property	**Value**
Diameter	1.2 mm
Density	1000 kg / m^3
Friction coefficient	0.1
COR	0.98
Stiffness	8 N/m

Table 4-2 : Properties of the fluid medium

Fluid Property	**Value**
Density	1.205 Kg / m^3
Viscosity	1.8 x 10^{-5} Kg / m s

The validation cases described in this section are run for 10 seconds and the results were averaged for the last 5 seconds. A time-driven model is used to simulate these cases. Since the time-driven model is inherently restrictive on the time step, the simulations would take significant time to be done with, particularly in cases wherein the numerical time step is considerably low. To circumvent this issue and to ensure the physics of the simulation remains reasonably accurate, the numerical time step value used in Elghannay et al.[45] has been increased to 50 μs. In order to resolve the collisions accurately, the ratio of minimum collision time to the numerical time step (eq.3.52) has to evaluate to 15 at the very least. If a numerical time step of 50 μs is used, accompanied by a stiffness coefficient of 800 N/m, the ratio becomes 1.56. Hence, the stiffness coefficient for the particle-particle collision and particle-wall collision has been decreased to 8

N/m. The COR provided as an input to the collision model for particle-particle collisions and particle-wall collisions is 0.98. Furthermore, since GenIDLEST had already possessed the capability to simulate spherical particles, the same simulation with identical parameters was carried out with the spherical DEM model in GenIDLEST.

Time averaged results from the spherical DEM model (using spherical particle assumption for collision detection and resolution) and the non-spherical DEM model (using non-spherical treatment of particle collisions) are presented in this section. The difference between the two models is that the computations done on the particles by spherical DEM model involve only the centroid and the radius of the spherical particle. On the other hand, computations done by the non-spherical DEM model involve all the surface vertices on the particle and the collision detection and resolution method outlined in this thesis. Figure 4-10 presents a comparison between the time averaged fluid vertical velocity, in the mid plane across z axis, resulting from both the models. Additionally, Figure 4-11 and Figure 4-12 illustrate the time averaged particle velocities in the vertical direction and the averaged void fraction, respectively. From these results the averaged fluid velocities, particle velocities and the void fraction values of the non-spherical DEM model compare well with that of the spherical DEM model. The difference between the mean values of these variables, rising out of the two models, evaluates to 0.3% on average.

Table 4-3 shows a comparison of the wall clock time taken for DEM computations by the respective models running on two cores. It has to be noted that the wall clock time listed accounts only for that taken by the DEM models and does not account for the time taken by the fluid solver. The results corroborate the fact that the non – spherical DEM model is computationally more expensive; in this case, the spherical DEM model is roughly 1.5 times faster than the non-spherical DEM model.

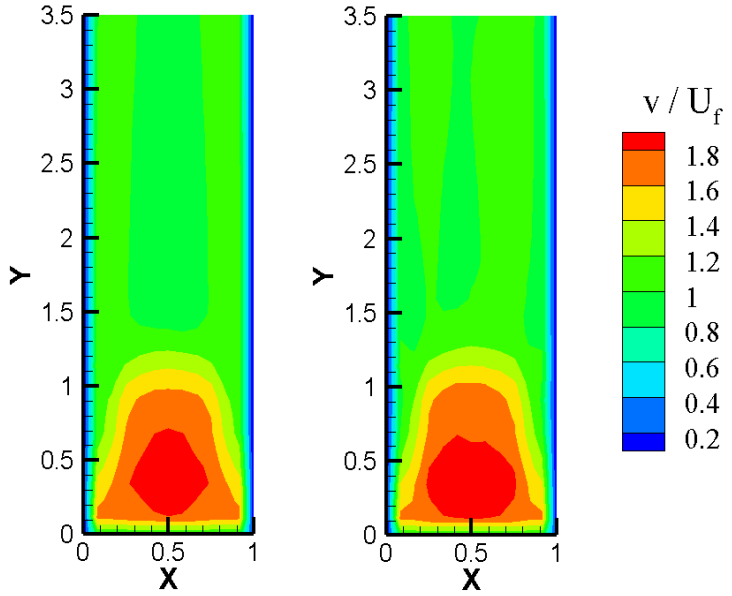

Figure 4-10 : Comparison of the averaged fluid vertical velocity in the mid plane between the non spherical DEM (left) and spherical DEM (right)

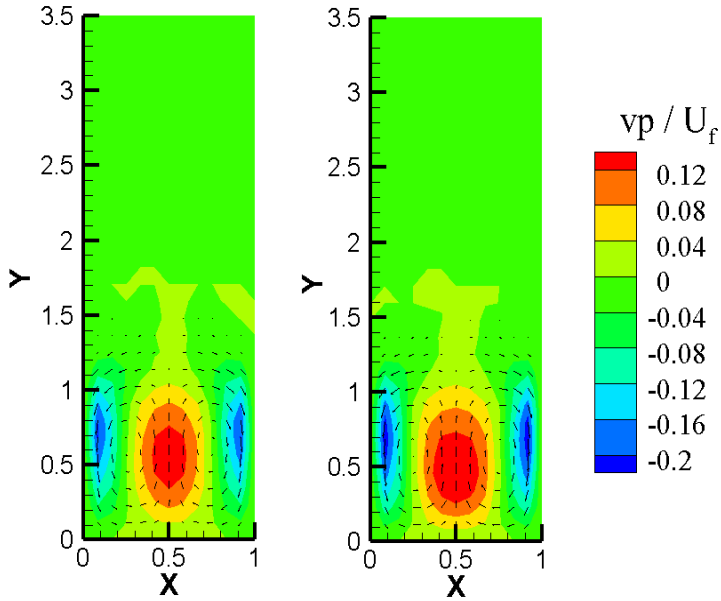

Figure 4-11 : Comparison of the averaged particle vertical velocity in the mid plane between the non spherical DEM (left) and spherical DEM (right)

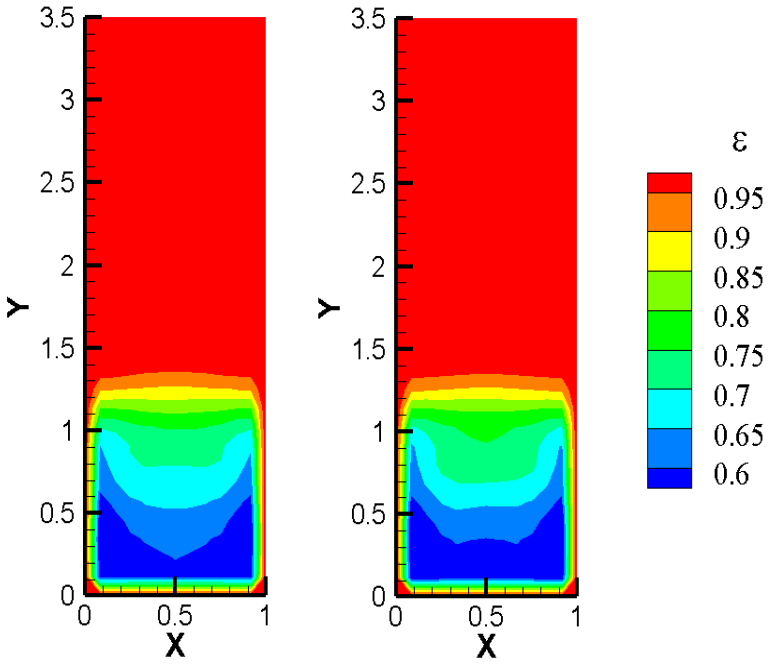

Figure 4-12 : Comparison of the averaged void fraction in the mid plane between the non-spherical DEM (left) and spherical DEM (right)

Table 4-3 : Comparison of wall clock time taken for spherical and non-spherical DEM fluidized bed computations

Model	**Wall clock time taken for respective DEM computations on two computational cores**
Spherical DEM	1207040 seconds (or) 13.97 days
Non – spherical DEM	1661787 seconds (or) 19.23 days

To validate the model with experiments, the time-averaged void fraction and particle velocities are compared at specific locations. The averaged void fraction 'ε', in the center z-plane is computed at y = 16.4 mm and 32.2 mm above the permeable wall. From the depicted values in Figure 4-13, it is observed that the averaged void fraction matches well with the results of Muller et al.[43] at a height of 16.4 mm. At a height of 32.2 mm, the void fraction values measure up to

the simulation results obtained by Muller et al. [43]. However, the experimental measurements do not compare well with any of the simulation results - inclusive of that by Muller et al. [43] - as evidenced by Figure 4-13.

Moreover, the average particle velocities are calculated at horizontal locations 15 mm and 35 mm above the permeable wall and compared with the experimental measurements, as illustrated in Figure 4-14. At horizontal locations 15 mm above the permeable wall, the dimensional particle velocities in the vertical direction compare well with the experimental and simulation measurements of Muller et al. [43]. At 35 mm above the permeable wall, the values estimated by the spherical DEM and non-spherical DEM are well below the resulting experimental and simulation values at the center. The predicted trends are also quite dependent on the drag correlations used. For current purposes it is sufficient to show that the non-spherical collision model predicts the same mean bed dynamics as that resulting from the spherical particle DEM.

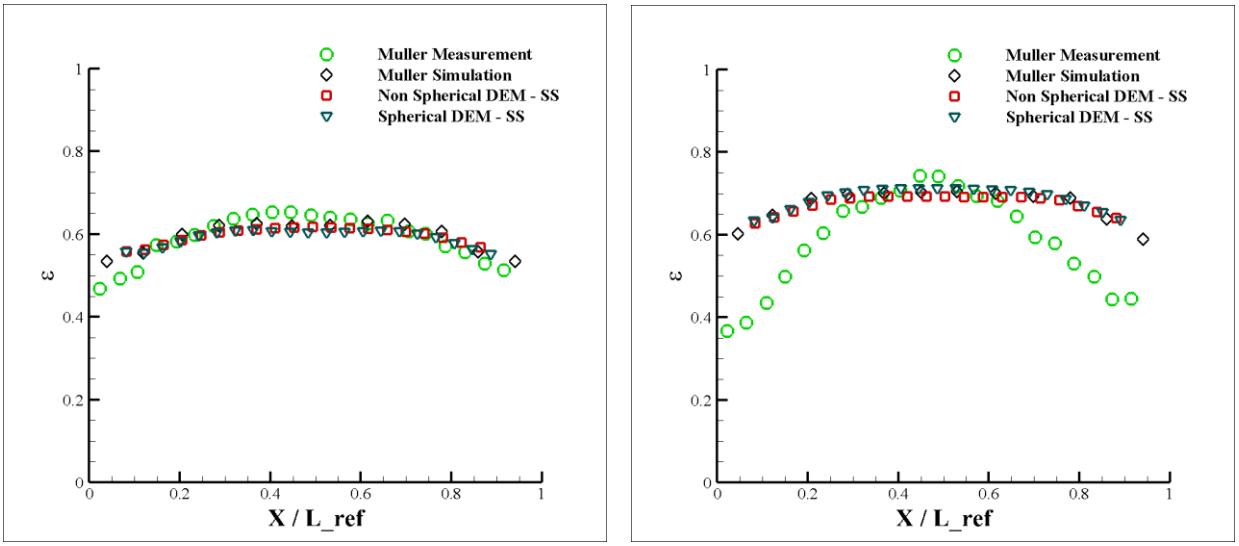

Figure 4-13 : Comparison of averaged void fraction values at 16.4 mm (left) and 32.2 mm (right) with literature

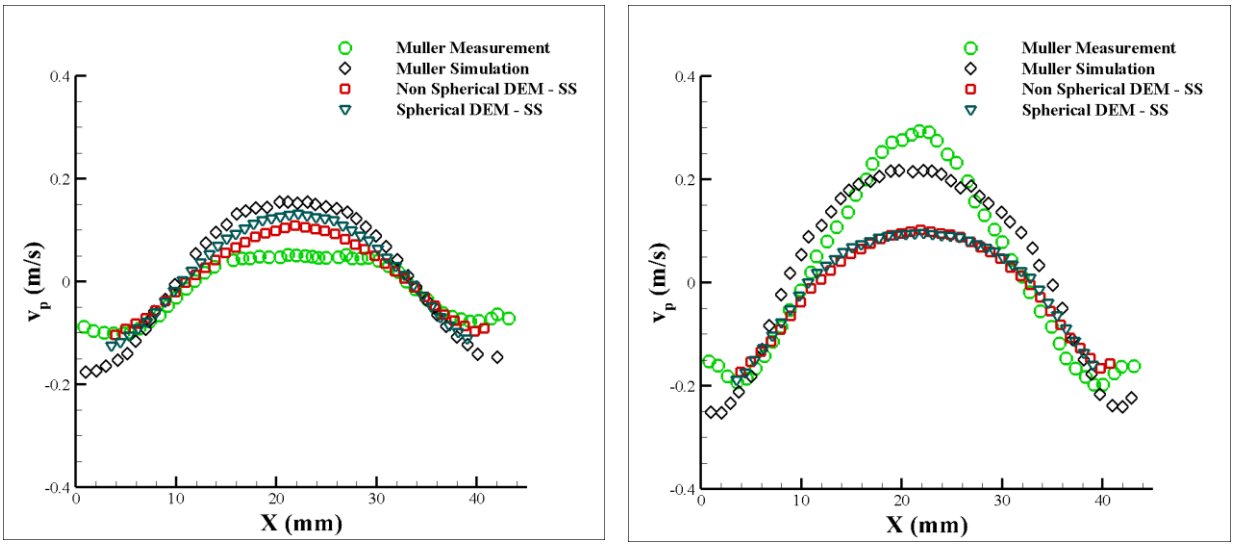

Figure 4-14 : Comparison of the averaged particle vertical velocity at 15 mm (left) and 35 mm (right) with literature

4.2. Computational Performance Metrics

The computational performance of the non-spherical collision model is evaluated in terms of parallel seep up and efficiency. The speed up and efficiency of the model when parallelized is evaluated test cases involving non-spherical particles dynamically influenced by gravity in the absence of any fluid. Another aspect of the model influencing wall clock time is the number of surface vertices on the particle. This aspect has an effect on the accuracy of the simulation and the total time taken for simulation as the number of floating point operations done in the collision algorithms depends on the number of surface vertices. It has been noted that for a non-spherical particulate simulation, most of the wall clock time is spent in resolving collisions as opposed to spherical particulate simulation wherein most of the wall clock time is spent in neighbor search algorithm[46].

4.2.1. Speed Up and Efficiency of the Model

In computational terms, scalability refers to the measurement of speedup with respect to the number of processors used. A parallelizable code's performance is generally evaluated by carrying out strong and weak scalability tests. In a strong scalability test, the problem size remains the same while increasing the number of processors. A weak scalability test on the other hand measures the performance of the code when the problem size increases in proportion to the number of processors. The former is a better measure of the scalability of the software, whereas the latter is more representative of the scalability of the hardware architecture.

The performance of the non – spherical collision model is assessed on a parallel architecture through a strong scalability test. The problem size - in this case, the number of particles and the computational domain – is fixed while varying the number of processors. In a scalability test, an important aspect of measuring the performance is the speedup. Speedup is defined as the ratio of time taken for the simulation to complete in a serial computation to that in a parallelized setup. The ratio can be expressed as,

$$S = \frac{T_s}{T_p} \tag{4.6}$$

where, T_s represents the total simulation time taken in a serial computation and T_p represents the total simulation time taken in a parallel computation on 'p' processors. The computational domain used in these tests is shown in Figure 4-15 and in Figure 4-16. An eight block set up is initialized with 250 particles each; each ellipsoid has 400 vertices approximating the particle's geometry. Gravitational force is the solitary force acting on the particles along the vertical direction. Collisions between particles and the wall are resolved using the soft sphere model. The physical parameters approximating the particles' kinematics are shown in Table 4-4. The simulation is run

for a total of 2.6 s of actual physical time, which would allow for the particles to be evenly spread among the blocks, thereby resulting in a reasonable load distribution among the processors. The problem is run on 2, 4, and 8 processors, in addition to a serial computation done on one processor.

Another supplementary metric that can be utilized to measure the performance of a parallelizable problem is efficiency. Efficiency is defined as the ratio of observed speedup to theoretical speed up. Ideally, with the increase in number of processors, the efficiency of a simulation has to be 100 %. However, due to parallel overhead, the efficiency of a simulation in practice is smaller. The efficiency of a parallelizable simulation, run on 'p' processors is given by,

$$E = \frac{T_s}{p * T_p} \tag{4.7}$$

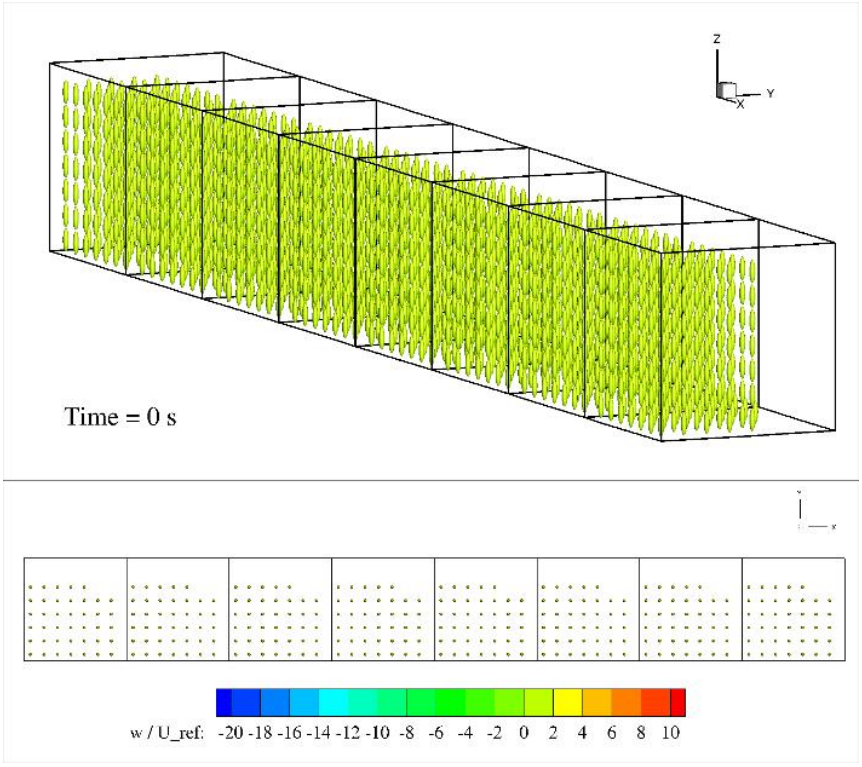

Figure 4-15 : Computational domain at t = 0 s, viewed from a perspective (top) and from the top (bottom)

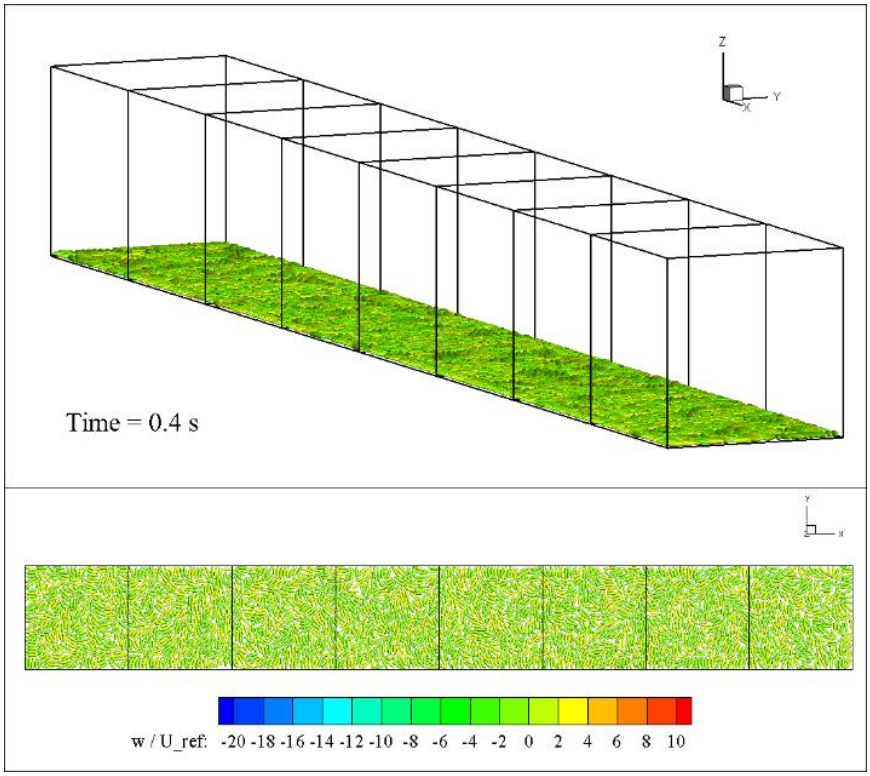

Figure 4-16 : Computational domain at t = 0.4 s, viewed from a perspective (top) and from the top (bottom)

Table 4-4 : Physical Properties of the ellipsoids

Particle Property	**Value**
Major axis length	4 mm
Minor axis length	1 mm
Density	2650 kg / m^3
Friction coefficient	0.5
COR	0.5
Stiffness	40 N/m

Table 4-5 provides a concise summary of the wall clock time observed for each simulated case and the corresponding speedup and efficiency. It can be discerned that the efficiency decreases with increasing number of processors, whereas the speedup increases with increasing number of processors. The observed values of speedup and efficiency are comparable to the expected theoretical values. The trend noticed in these scalability tests is in good accordance with similar results in literature [46].

The values of speedup and efficiency in Table 4-5 have been calculated based on eq.4.6 and eq.4.7. For example, the speedup and efficiency for the simulation run on 8 processors has been calculated as follows,

$$S = \frac{T_1}{T_8} = \frac{213323}{29716} = 7.179$$

$$E = \frac{T_1}{8 * T_8} = \frac{213323}{8 * 29716} = 0.8973 (or)\ 89.73\ \%$$

Table 4-5 : Calculation of speedup and efficiency of the non-spherical DEM model

No. of processors	**Wall clock time taken (in seconds)**	**Speedup**	**Efficiency (%)**
1	213323	-	-
2	110241	1.934	96.7
4	56907	3.749	93.72
8	29716	7.179	89.73

4.2.2. Factors influencing time-to-solution

This section attempts to establish the sensitivity of time-to-solution to the number of vertices used to define the particle surface and to the number of collisions. The number of surface vertices play a major role in the computation time. As the number of surface vertices increase, the accuracy of collision resolution increases but at the cost of more floating point operations. However, in a majority of simulations, beyond a certain required minimum, the increase in the overall accuracy of the simulation as the number of surface vertices increase is marginal. Time-to-solution also depends on the number of collisions that have to be modeled – a dense suspension will take longer than a dilute suspension.

In these simulation cases, spherical particles are generated and each sphere's surface is approximated by 400, 900 and 1600 surface vertices, constituting three different simulations. Spherical particles are chosen owing to the fact that these particles form a denser packing and are comparatively better distributed among the processors. Also, spherical particles offer an upper bound on the number of collision instances due to their densely packed nature. The computational setup is analogous to that shown in Figure 4-15, except that 1600 spheres are generated in the domain and equally distributed among the 8 blocks. In all the tests described in this section, 8 processors have been utilized to simulate the kinematics. Gravitational forces act on the particle, along the vertical direction, and the fluid medium is de-activated in these tests. The motive of these tests is to provide a ballpark metric for wall clock time normalized by the number of collisions and number of processors.

Table 4-6 summarizes the results and observations. It has to be noted that the values labeled 'Total no. of collisions' does not represent the number of instances of solitary collision

between particles. As the time-driven model resolves collision in successive time steps for every scenario of collision between two particles (or) collision between a particle and wall, this metric is rather a cumulative estimate of each binary collision. For instance, if two particles collide at timestep ‘n’ and the collision is completely resolved at timestep ‘n+9’, the total number of collisions would accrue up to 20 as the two particles are in contact with each other for 10 time steps.

From the results shown in Table 4-6 it can be noted that there is a clear linear dependency on the wall clock time per collision per processor as evidenced by the respective values ensuing from simulations with 400 and 1600 surface vertices. The number of particle wall collisions and particle-wall collisions increase which indicates a slightly more accurate simulation, however, there are no discernible differences in the spatial and temporal advancement of the spherical particles among the three different test cases.

Table 4-6 : Normalization of wall clock time with respect to the number of collision instances and number of processors

No. of surface vertices on each particle	**Wall clock time (in seconds)**	**No. of particle-particle collision instances**	**No. of particle-wall collision instances**	**Total no. of collision instances**	**Wall clock time per collision instance per processor (in µs)**
400	7645	62415940	10567446	72983386	13.1
900	18972	80681123	12208878	92890001	25.5
1600	37098	86898067	13750284	100648351	46.07

From Table 4-7, it can be observed that the wall clock time between the spherical and non – spherical DEM model roughly scales by a factor of 3. In this simulation, since the fluid medium

is deactivated, the variation in wall clock time between the two models is only dependent on the collision instances.

Table 4-7 : Comparison of wall clock time with respect to number of collision instances for spherical and non-spherical DEM models

Model	**Wall clock time (in seconds)**	**No. of particle-particle collision instances**	**No. of particle-wall collision instances**	**Total no. of collision instances**	**Wall clock time per collision instance per processor (in µs)**
NSP – DEM (400 surface vertices on each particle)	7645	62415940	10567446	72983386	13.1
Spherical DEM	2592	60685720	10224988	70910708	4.56

The scalability and computational performance tests, described in sections 4.2.1 and 4.2.2 respectively, were carried out on Intel Xeon E5-2670 CPUs clocking at 2.6 GHz and consisting of 64 GB of memory in total.

Chapter 5

5. Conclusion and Future Work

5.1. Conclusion

In this research work, a new framework to simulate non spherical particle dynamics has been developed. Superellipsoids – a broad class of non – spherical particle geometries - are generated by representing their surfaces with a discrete function representation. Collisions are detected by evaluating the surface vertex of a particle with respect to the surface function of another particle in possible contact. A novel collision detection algorithm to detect collisions between particles and walls is incorporated. Following detection of a collision, the collisions are resolved by event-driven and time-driven models. In an event-driven model, an impulse based collision resolution technique is employed whereas in a time-driven model, a linear spring-dashpot model is utilized to resolve collisions.

The non-spherical DEM model implementation is verified by using conservation of energy principles through multiple simulated collisions of cylinders and ellipsoids. The temporal progression of energy is monitored at various coefficients of restitution and the resulting magnitudes of total kinetic energy are verified. The non-spherical DEM model is coupled with a fluid solver – GenIDLEST. The developed model is validated with results from literature. In the non-spherical particle DEM model, spheres are generated and the resulting void fraction and particle velocities at specific locations in the computational domain are compared with experimental measurements. It is observed that the void fraction values and the particle velocities match well with the experimental results at lower vertical distances from the plate. A comparison

of the wall clock time taken by the spherical and non – spherical DEM is presented – in the presence of fluid medium - and it is noted that the non-spherical DEM is roughly 2.5 times slower than the spherical DEM model, on average, largely due to the increase in floating point operations.

The parallel scalability of the non-spherical DEM model is evaluated. Strong scalability tests, wherein the problem size is maintained constant as the number of processors are increased, are carried out with non-spherical particles. The implementation and execution of the simulation on a parallel architecture is evaluated in terms of efficiency and speedup from one to eight processors using 2000 particles. A parallel efficiency of nearly 90% is maintained up to eight processors illustrating that approximately 250 particles with 400 vertices are sufficient to give good parallel scalability.

Another aspect of non–spherical particles, influencing the accuracy and the wall clock time of simulations, is the number of surface vertices used to approximate the surface. It is noted that the accuracy of results does not differ significantly with increasing number of surface vertices. The simulation time increases linearly with the number of surface vertices. Metrics that delineate the wall clock time utilized per collision per processor have been documented in this research work. These tests provide an estimate of the expected wall clock times for non-spherical particulate simulations.

The development of a three dimensional DEM model that accounts and resolves collisions between non-spherical particles enables the capability to accurately calculate the drag correlations and heat transfer coefficients in granular assemblies following successful coupling with GenIDLEST.

5.2. Future Work

Although the non-spherical DEM model developed is coupled with a fluid solver – GenIDLEST – formulations to compute the drag forces / correlations for non – spherical particles are non – existent. The inclusion of the appropriate formulations would empower the functioning and applicability to non-spherical particle-fluid interactions. Furthermore, the functioning of the time-driven model is dependent on the mathematical formulations of the collision forces. Presently, the restoration force model is assumed to be linear. Non – linear relations to calculate the reaction forces on particles can be incorporated. The accuracy and wall clock time of that simulation can be compared to that ensuing from a linear model. Efforts can be made in the future to improve the efficiency and speedup of the non-spherical DEM model by reducing the message passing involved in certain circumstances.

References

[1] P. A. Cundall and O. D. L. Strack, “Discussion: A discrete numerical model for granular assemblies,” *Géotechnique*, vol. 30, no. 3, pp. 331–336, 2009.

[2] C. Hogue, “Shape representation and contact detection for discrete element simulations of arbitrary geometries,” *Eng. Comput.*, vol. Vol. 15, no. Issue: 3, p. pp.374-390, 1998.

[3] G. Lu, J. R. Third, and C. R. Müller, “Discrete element models for non-spherical particle systems: From theoretical developments to applications,” *Chem. Eng. Sci.*, vol. 127, pp. 425–465, 2015.

[4] H. P. Zhu, Z. Y. Zhou, R. Y. Yang, and A. B. Yu, “Discrete particle simulation of particulate systems: Theoretical developments,” *Chem. Eng. Sci.*, vol. 62, no. 13, pp. 3378–3396, 2007.

[5] Z. Y. Zhou, D. Pinson, R. P. Zou, and A. B. Yu, “Discrete particle simulation of gas fluidization of ellipsoidal particles,” *Chem. Eng. Sci.*, vol. 66, no. 23, pp. 6128–6145, 2011.

[6] D. O. Njobuenwu and M. Fairweather, “Dynamics of single, non-spherical ellipsoidal particles in a turbulent channel flow,” *Chem. Eng. Sci.*, vol. 123, pp. 265–282, 2015.

[7] J. R. Williams and A. P. Pentland, “SUPERQUADRICS AND MODAL DYNAMICS FOR DISCRETE ELEMENTS IN INTERACTIVE DESIGN,” *Eng. Comput.*, vol. 9, no. 2, pp. 115–127, 1992.

[8] Barr, “Superquadrics and Angle-Preserving Transformations,” *IEEE Comput. Graph. Appl.*, vol. 1, no. 1, pp. 11–13, 1981.

[9] B. Muth, M. K. Müller, P. Eberhard, and S. Luding, “Collision detection and administration

methods for many particles with different sizes," no. Md, pp. 1–18, 2007.

[10] M. Teschner *et al.*, "Collision Detection for Deformable Objects," *Eurographics State-of-the-Art Rep.*, 2004.

[11] J. R. Williams and R. O 'connor, "Discrete Element Simulation and the Contact Problem," vol. 6, no. 4, pp. 279–304, 1999.

[12] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE J. Robot. Autom.*, vol. 4, no. 2, pp. 193–203, 1988.

[13] K. Dong, C. Wang, and A. Yu, "A novel method based on orientation discretization for discrete element modeling of non-spherical particles," *Chem. Eng. Sci.*, vol. 126, pp. 500–516, 2015.

[14] D. Höhner, S. Wirtz, and V. Scherer, "Experimental and numerical investigation on the influence of particle shape and shape approximation on hopper discharge using the discrete element method," *Powder Technol.*, vol. 235, pp. 614–627, 2013.

[15] A. Wachs, L. Girolami, G. Vinay, and G. Ferrer, "Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part I: Numerical model and validations," *Powder Technol.*, vol. 224, pp. 374–389, 2012.

[16] X. Lin and T.-T. Ng ', "Contact Detection Algorithms For Three-Dimensional Ellipsoids in Discrete Element Modelling," *Int. J. Numer. Anal. Methods Geomech.*, vol. 19, no. March, pp. 653–659, 1995.

[17] D. Höhner, S. Wirtz, and V. Scherer, "A study on the influence of particle shape and shape

approximation on particle mechanics in a rotating drum using the discrete element method," *Powder Technol.*, vol. 253, pp. 256–265, 2014.

[18] G. Lu, J. R. Third, and C. R. Müller, "Critical assessment of two approaches for evaluating contacts between super-quadric shaped particles in DEM simulations," *Chem. Eng. Sci.*, vol. 78, pp. 226–235, 2012.

[19] P. W. Cleary, N. Stokes, and J. Hurley, "Efficient Collision Detection for Three Dimensional Super-ellipsoidal Particles Paul W. Cleary and Nick Stokes Jarrod Hurley," *World*, pp. 1–7.

[20] A. P. J.R. Williams, "Superquadrics object representation for dynamics of multi-body structures," in *Proceedings of ASCE Structures*, 1989.

[21] X. Tang, A. Paluszny, and R. W. Zimmerman, "Energy conservative property of impulse-based methods for collision resolution," *Int. J. Numer. Methods Eng.*, vol. 95, no. 6, pp. 529–540, 2013.

[22] G. T. Houlsby, "Potential particles: a method for modelling non-circular particles in DEM," *Comput. Geotech.*, vol. 36, no. 6, pp. 953–959, 2009.

[23] D. Müller and T. M. Liebling, "Detection of Collisions of Polygons by Using Triangulation," in *Contact Mechanics*, M. Raous, M. Jean, and J. J. Moreau, Eds. Boston, MA: Springer US, 1995, pp. 369–372.

[24] P. A. Cundall, "Formulation of a Three-dimensional Distinct Element Model - Part I. A Scheme to Detect and Represent Contacts in a System Composed of Many Polyhedral Blocks," *Int. J. Rock Mech. Min. Sci. Geomech.*, vol. 25, no. 3, pp. 107–116, 1988.

[25] J. R. Williams and R. O'connor, "A linear complexity intersection algorithm for discrete element simulation of arbitrary geometries," *Eng. Comput.*, vol. 12, no. 2, pp. 185–201, 1995.

[26] J. F. Favier, M. H. Abbaspour-Fard, M. Kremmer, and A. O. Raji, "Shape representation of axi-symmetrical, non-spherical particles in discrete element simulation using multi-element model particles," *Eng. Comput. (Swansea, Wales)*, vol. 16, no. 4, pp. 467–480, 1999.

[27] J. F. Ferellec and G. R. McDowell, "A method to model realistic particle shape and inertia in DEM," *Granul. Matter*, vol. 12, no. 5, pp. 459–467, 2010.

[28] A. Jaklič, A. Leonardis, and F. Solina, *Segmentation and Recovery of Superquadrics*, vol. 20. 2000.

[29] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi, "I-COLLIDE: an interactive and exact collision detection system for large-scale environments," *Proc. 1995 Symp. Interact. 3D Graph.*, pp. 189–197, 1995.

[30] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-DOPs," *IEEE Trans. Vis. Comput. Graph.*, vol. 4, no. 1, pp. 21–36, 1998.

[31] B. C. Vemuri, Y. Cao, and L. Chen, "Fast Collision Detection Algorithms with Applications to Particle Flow," *Comput. Graph. Forum*, vol. 17, no. 2, pp. 121–134, 1998.

[32] J. R. Williams and A. Pentland, "Object Representation for Design-Unifying Cubes and Spheres," *Proc. COMPLAS 89, 2nd Int. Conf. Comput. Plast.*, p. 14, 1989.

[33] D. Höhner, S. Wirtz, and V. Scherer, "A study on the influence of particle shape on the

mechanical interactions of granular media in a hopper using the Discrete Element Method," *Powder Technol.*, vol. 278, pp. 286–305, 2015.

[34] A. Džiugys and B. Peters, "An approach to simulate the motion of spherical and non-spherical fuel particles in combustion chambers," *Granul. Matter*, vol. 3, no. 4, pp. 231–266, 2001.

[35] D. Baraff, "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies," *Comput. Graph. (ACM).*, vol. 23, no. 3, pp. 223–232, 1989.

[36] X. H. Tang, A. Paluszny, and R. W. Zimmerman, "An impulse-based energy tracking method for collision resolution," *Comput. Methods Appl. Mech. Eng.*, vol. 278, pp. 160–185, 2014.

[37] P. Lötstedt, "Coulomb Friction in Two-Dimensional Rigid Body Systems," *ZAMM - J. Appl. Math. Mech. / Zeitschrift für Angew. Math. und Mech.*, vol. 61, no. 12, pp. 605–615, 1981.

[38] Y. You and Y. Zhao, "Discrete element modelling of ellipsoidal particles using super-ellipsoids and multi-spheres: A comparative study," *Powder Technol.*, vol. 331, pp. 179–191, 2018.

[39] H. A. Navarro and M. P. de Souza Braun, "Determination of the normal spring stiffness coefficient in the linear spring-dashpot contact model of discrete element method," *Powder Technol.*, vol. 246, pp. 707–722, 2013.

[40] H. A. Elghannay and D. K. Tafti, "Alternate tangential impact treatments for the soft-sphere collision model," *Part. Sci. Technol.*, vol. 0, no. 0, pp. 1–11, 2019.

[41] Y. Tsuji, T. Kawaguchi, and T. Tanaka, “Discrete particle simulation of two-dimensional fluidized bed,” *Powder Technol.*, vol. 77, no. 1, pp. 79–87, 1993.

[42] Z. Y. Zhou, R. P. Zou, D. Pinson, and A. B. Yu, “Dynamic simulation of the packing of ellipsoidal particles,” *Ind. Eng. Chem. Res.*, vol. 50, no. 16, pp. 9787–9798, 2011.

[43] C. R. Müller *et al.*, “Validation of a discrete element model using magnetic resonance measurements,” *Particuology*, vol. 7, no. 4, pp. 297–306, 2009.

[44] D. Tafti, “Genidlest - A scalable parallel computational tool for simulating complex turbulent flows,” *Proc. ASME Fluids Eng. Div.*, vol. 256, 2000.

[45] H. Elghannay, D. Tafti, and K. Yu, “Evaluation of physics based hard-sphere model with the soft sphere model for dense fluid-particle flow systems,” *Int. J. Multiph. Flow*, vol. 112, 2018.

[46] B. Yan and R. A. Regueiro, “A comprehensive study of MPI parallelism in three-dimensional discrete element method (DEM) simulation of complex-shaped granular particles,” *Comput. Part. Mech.*, vol. 5, no. 4, pp. 553–577, 2018.

Appendices

Appendix A: Particle-wall Collision Resolution

A. Particle-wall Collision Resolution

A.1. Particle-Wall Hard Sphere Modeling

The event-driven collision resolution model employed in particle-wall collisions is similar to the particle-particle collision resolution model. The assumption of an impulse resolving collisions is implemented in wall collisions; collisions are resolved instantly as a large magnitude of force acts in a short interval of time. Akin to the particle-particle hard sphere modeling, the particle-wall collisions can be modeled with the inclusion of friction, in addition to the frictionless impact. Figure A-1 shows a representative case of frictionless impulse acting on a particle due to collision with wall.

A.1.1. Frictionless Impact

Following similar methodology as explained for particle-particle collisions, the wall collision hard sphere impulse is expressed as,

$$j_{wall} = \frac{-(1 + COR_{wall}) * \left(\vec{V}_{r,pre} \,.\, \hat{n}_w\right)}{\frac{1}{m_1} + \left(\left(\bar{\bar{I}}_1^{-1} * (r_1 \; x \; \hat{n}_w)\right) \; x \; \vec{r}_1\right) . \; \hat{n}_w} \tag{A.1}$$

where, $\hat{n}_{wall}$ is the normal of collision obtained from the collision model. An interesting feature of the wall collision event-driven model is that the mass and inertia of the wall are neglected. This is done considering the fact that the impulse expression entails the calculation of their inverses. Since the wall is assumed to have an infinitely large mass and moment of inertia compared to the

particles, their inverses are close to zero and can be neglected in comparison with the particle's mass and inertia. The other terms in the expression are equivalent to eq.3.24. Another advantage of the wall collision event-driven model is that the wall remains stationary post collision. Hence, the post collision velocities are updated only for the particle, as illustrated below.

$$\vec{v}_{1,post} = \vec{v}_{1,post} - \frac{j_{wall}}{m_1}.\hat{n}_w \tag{A.2}$$

$$\vec{\omega}_{1,post} = \vec{\omega}_{1,post} - j_{wall}.\bar{\bar{I}}_1^{-1} * (\vec{r}_1 \; x \; \hat{n}_w) \tag{A.3}$$

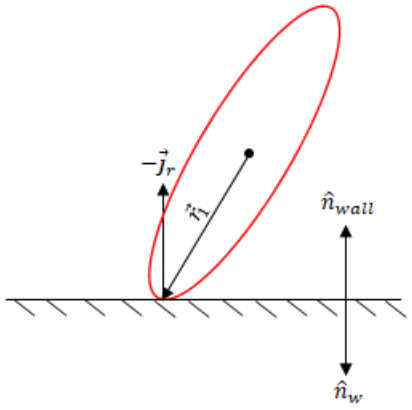

Figure A-1 : Representation of particle-wall collision resolution using impulses

A.2.1. Frictional Impact

Whenever a particle collides with a wall, the particle slides (or rolls) along the wall owing to the roughness elements on the wall. While simulating a particle's collision with a wall boundary it is of vital importance to recreate the detailed physics governing it. So as to incorporate the roughness factor in walls, the frictional impact model of the event-driven model to resolve collisions is essential. The friction impact of particle-wall collisions can be modeled using a

frictional impulse that arises and acts upon the particle's surface. The assumption while applying a frictional impulse to effectuate a change in particle dynamics is that the impulse vector is equal in magnitude and opposite in direction as compared to that acting on the wall. From the particle-particle frictional impact collision model, it is discerned that the frictional impulse vector consists of the static and dynamic impulses. For the sake of simplicity, without a significant loss in the accuracy of the simulation, the frictional impact particle-wall collisions model is developed with the assumption that the regime of application of frictional impulses is dynamic. The unit contact tangent acting along the collision plane is calculated as shown below.

$$\hat{t}_{wall} = \frac{\vec{V}_{r,pre} - \left(\vec{V}_{r,pre} \cdot \hat{n}_w\right) \hat{n}_w}{\left|\vec{V}_{r,pre} - \left(\vec{V}_{r,pre} \cdot \hat{n}_w\right) \hat{n}_w\right|} \tag{A.4}$$

An alternate method to calculate the tangent would be to determine the normal vector of the unit contact normal of the wall. Similar to the particle-particle collision dynamics, the frictional impulse is derived from the frictionless impulse, with the coefficient of friction acting as an additional scaling term. It has to be noted that the following expressions follow a similar procedure to that explained in section 3.1.2, wherein the impulses are the eventual consequence of the integration of forces over the collision time. Also, in the current framework, the static frictional impulse for the wall collision model is neglected, and the dynamic frictional impulse is computed.

$$j_{d,wall} = \mu_{d,wall} \cdot j_{wall} \tag{A.5}$$

Using the above expression for the frictional impulse magnitude and the unit contact tangent, the frictional impulse vector can be defined as,

$$\vec{J}_{f,wall} = \begin{cases} \left(-m\,\vec{V}_{r,pre} \cdot \hat{t}_{wall}\right) \hat{t}_{wall} & if\ \vec{V}_{r,pre} \cdot \hat{t} = 0 \\ -j_{d,wall} \cdot \hat{t}_{wall} & otherwise \end{cases} \tag{A.6}$$

As a consequence, a particle colliding with a wall under the impact of frictional impulses, starts to slide instantaneously. The trajectory of the particles are updated by the equations illustrated below.

$$\vec{v}_{1,post} = \vec{v}_{1,post} - \frac{j_{d,wall}}{m_1}.\hat{t}_{wall} \quad (A.7)$$

$$\vec{\omega}_{1,post} = \vec{\omega}_{1,post} - j_{d,wall}\,.\,\bar{\bar{I}}_1^{-1} * (\vec{r}_1\, x\, \hat{t}_{wall}) \quad (A.8)$$

The procedure of updating velocities is sequentially done for all particles colliding with the wall. Moreover, the velocities for a given particle are updated initially by the particle-particle collision model – comprising of both frictionless and frictional impact. Subsequently, the velocities for that particular particle is updated by the frictionless and friction impact impulses stemming from particle-wall collision. So, for any given particle undergoing collision with at least one another particle and the wall, its velocities are updated by four different expressions, sequentially. Furthermore, it has to be noted that the pre- collision total relative velocity term used in the particle-wall collision model (inclusive of frictionless and frictional impact) comprises of the particle's velocities only, since the wall is stationary. Mathematically, this can be formulated as,

$$\vec{V}_{r,pre} = -\vec{V}_{p1,pre} \quad (A.9)$$

In the above expression for the total relative velocity before collision, the total velocity of the particle can further be decomposed into its linear and rotational components.

$$\vec{V}_{p1,pre} = \vec{v}_{1,pre} + \vec{\omega}_{1,pre}\, x\, \vec{r}_1 \quad (A.10)$$

In majority of simulations, the particle-wall collision model developed based on the event-driven model is relatively easier to implement and parallelize. The computational time and memory required to simulate the particle dynamics is less compared to the time-driven model. However, a major drawback of this model is that the relocation vector applied to advance the particle spatially, could lead to situations wherein the particle does not reach a completely stationary state – assuming that the COR of the collision is close to zero. These instances are especially true when simulating dense particulate systems. The relocation vector resulting from a particle colliding with another internal particle can push the particle beyond the wall boundary, provided the particle is close enough and colliding with the wall. Because of this, a large time step could result in particles penetrating into the wall and being lost.

A.2. Particle-Wall Soft Sphere Modeling

Particle-wall collisions can also be modeled based on the time-driven model, in addition to event-driven models, as is the case with particle-particle collisions. In particle-wall soft sphere collision modeling, the wall can be thought of as a particle with infinite radius. The wall is modeled as a linear spring-dashpot system which can theoretically deform and provide a reaction force on the particle colliding with it. The reaction forces are calculated based on Newton's laws of motion. The assumption while using a soft-sphere model for particle-wall collisions is that the collision is resolved over an entire contact area or volume rather than a single point of contact.

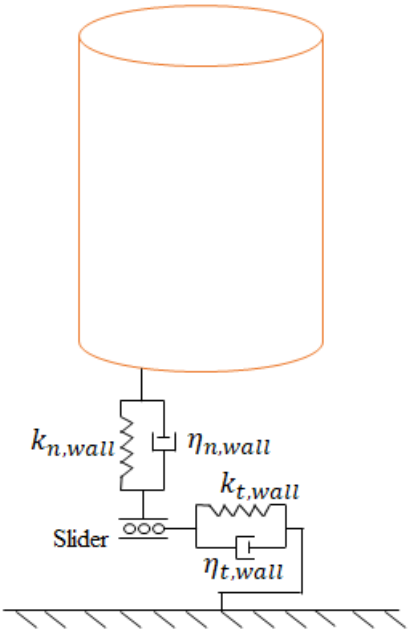

Figure A-2 : Depiction of time-driven impact model employed to resolve particle-wall collisions

Analogous to the particle- particle collision model, the particle-wall collision resolution is carried out using normal and tangential reaction forces acting on the particle's surface. A typical normal and tangential time-driven model employed in the framework for particle-wall collisions is depicted in Figure A-2. The normal force is given by the following expression,

$$\vec{F}_{n,wall} = -k_{n,wall}\,\vec{\delta}_{n,wall} - \eta_{n,wall}\,\vec{v}_{n,wall} \qquad (A.11)$$

where, $k_{n,wall}$ represents the normal stiffness coefficient for the wall. $\vec{\delta}_{n,wall}$ denotes the relocation vector which is a measure of the maximum magnitude of penetration of the particle beyond the wall boundary. $\vec{v}_{n,wall}$ is the relative normal velocity of the particle colliding with the wall and is expressed as,

$$\vec{v}_{n,wall} = \left(\vec{V}_{r,pre} \,.\, \hat{n}_w\right) .\ \hat{n}_w \qquad (A.12)$$

In the above expressions, $\vec{V}_{r,pre}$ is composed of the relative velocity of the particle alone. The computation of the damping factor is similar to the particle-particle contact and is illustrated below.

$$\eta_{n,wall} = 2 * \gamma * \sqrt{m_{red,wall} * k_{n,wall}} \quad (A.13)$$

where, 'm_{red}' refers to the reduced mass of the colliding particle and the wall. This effective mass contains only the mass of the particle as the mass of the wall is very large compared to it.

$$m_{red,wall} = m_1 \quad (A.14)$$

The other constituent term γ can be expressed as,

$$\gamma = -\frac{ln(COR_{wall})}{\sqrt{\pi^2 + (ln(COR_{wall}))^2}} \quad (A.15)$$

A COR_{wall} value of 1 translates into a collision with no loss of energy. This represents the theoretical case precisely as for a COR_{wall} value of 1, the damping effects reduce to zero. The tangential force can also be expressed in the form of its stiffness and damping coefficients in similar fashion to the normal reaction force.

$$\vec{F}_{t,wall} = -k_{t,wall}\,\vec{\delta}_{t,wall} - \eta_{t,wall}\vec{v}_{t,wall} \quad (A.16)$$

In this expression, the stiffness coefficient serves as an input. The relocation vector $\vec{\delta}_{t,wall}$ is slightly different when compared to the particle-particle evaluations. Just like particle-particle collisions, the tangential force acts along the contact plane. The tangential damping factor is calculated as,

$$\eta_{t,wall} = 2 * \gamma * \sqrt{m_{red} * k_{t,wall}} \quad (A.17)$$

The tangential velocity $\vec{v}_{t,wall}$ acting in the direction perpendicular to the normal relative velocity is also similarly computed, based on the technique involved in particle-particle collisions. The notable difference between the models being that the tangential velocity consists of the relative approach velocity of the particle only.

$$\vec{v}_{t,wall} = \vec{V}_{r,pre} - \left(\vec{V}_{r,pre} \,.\, \hat{n}_w\right) . \hat{n}_w \tag{A.18}$$

Another significant difference between the particle-particle and particle-wall soft-sphere models manifests in the calculation of the tangential relocation vector. For particle-wall collisions the tangential relocation vector has a distinct approximation. This approximation is based on the minimum collision time and the ratio computed based on the particle and wall collision parameters. The minimum collision time for particle-wall collision is computed as follows.

$$\Delta t_{min\,wall} = \frac{\pi}{\sqrt{\frac{k_{n,wall}}{m_{red}} - \left(\frac{\eta_{n,wall}}{2\,.\,m_{red}}\right)^2}} \tag{A.19}$$

The resulting magnitude of $\Delta t_{min\,wall}$ is directly used in the calculation of the collision time and the subsequent tangential relocation vector.

$$\tau_{c,wall} = \Delta t_{min\,wall} \tag{A.20}$$

$$\vec{\delta}_{t,wall} = \tau_{c,wall} \,.\, \vec{v}_{t,wall} \tag{A.21}$$

The technique employed in approximating the tangential relocation vector as given in Tsuji et al. [41] follows the reasoning that the computation of the cumulative tangential relocation for particle-wall collisions is time and memory intensive. As is the case with particle-particle time-driven model, the stiffness coefficient and the COR of collision has to be provided as input

parameters to resolve particle-wall collisions. The magnitude of the tangential force in some cases can evaluate to a considerably large magnitude. To ensure the particles do not seem to exhibit increased kinetic energy due to the higher magnitudes of forces, the tangential force is restricted to a certain limit – theoretically known as the sliding limit – beyond which the particle enter the sliding regime without undergoing further deformation.

$$\vec{F}_{t,wall} = \begin{cases} -k_{t,wall}\,\vec{\delta}_{t,wall} - \eta_{t,wall}\vec{v}_{t,wall} & if\ \left|\vec{F}_{t,wall}\right| < \mu\left|\vec{F}_{n,wall}\right| \\ -\mu\left|\vec{F}_{n,wall}\right| . \dfrac{\vec{v}_{t,wall}}{\left|\vec{v}_{t,wall}\right|} & otherwise \end{cases} \quad (A.22)$$

The equations presented in eq.3.32 are analogous to the equations depicted above. Eventually, the total force acting on the particle is calculated as,

$$\vec{F}_{c,wall} = \vec{F}_{n,wall} + \vec{F}_{t,wall} \quad (A.23)$$

$$\vec{F}_{c,total} = \vec{F}_{c} + \vec{F}_{c,wall} \quad (A.24)$$

The total collision force ensuing from the particle-wall collision model is successively added to the total collision force resulting from particle-particle collisions for the concerned particle. Effectively, the total collision force could be described as the total reaction force acting on a given particle due to the collisions encountered with its neighboring particles and that owing to the walls.

Similar to the particle-particle soft sphere model, the total particle-wall collision force contributes to a torque. The torque is then, added cumulatively to the torque arising out of particle-particle collisions to effectuate an update in the rotational velocity. This operation ensures the underlying physics behind the kinematics of particles colliding with neighboring particles and the wall remains intact.

$$\vec{T}_{c,wall} = \vec{r}\, x\, \vec{F}_{c,wall} \tag{A.25}$$

$$\vec{T}_{c,total} = \vec{T}_c + \vec{T}_{c,wall} \tag{A.26}$$

The advantage of this model over the event-driven model becomes apparent when working with dense particulate flows in wall bounded domains. Although the time step is more restrictive inherently and the computational resources required are higher, the collisions between particles and walls are completely resolved leading to less possibilities of scenarios wherein the particles move beyond the computational domain.

Appendix B: Numerical Implementation

B. Numerical Implementation

The DEM model developed has to be coupled with a fluid flow solver. An incompressible variable-property Navier-Stokes solver - GenIDLEST - is used to simulate the fluid flow. GenIDLEST - an abbreviation for Generalized Incompressible Direct and Large Eddy Simulation of Turbulence – is an in-house fluid solver developed by Tafti[44]. The general schematic of GenIDLEST is illustrated in Figure B-1. A general outline of the fluid solver methodology is described here. The computational domain is discretized in the form of a structured mesh and the incompressible Navier - Stokes equations are solved using a pressure based method. A pseudo-staggered grid formulation is adapted in which the fluid velocities and pressure are stored in the cell centers and the mass fluxes are stored at cell faces. The formulation helps to ensure the order of accuracy of the pressure solver does not decrease. The linear systems governing momentum and pressure equations are solved using a stabilized biconjugate gradient solver (BiCGSTAB). Subsequently, the particle trajectories are updated using a coupled technique as presented in Figure B-1.

DEM serves as an additional functionality of GenIDLEST and the existing model works well for spherical particles. The goal of this research work is to elevate the capability of the existing model to effectively simulate non-spherical particles. The numerical implementation of the new model has to be in accordance with the base framework of GenIDLEST so as to ensure the error-free functioning of the flow solver. The model developed has multiple subroutines – each carrying out a specified task – constituting the DEM framework (highlighted in blue in Figure B-1).

Furthermore, since GenIDLEST is a parallel code, the message passing buffers involved in transferring data between processors have to be modified to account for variable arrays containing information about the dynamics of non-spherical particles. This chapter provides a detailed note on the implementation and the overall congruence of the various subroutines working on non-spherical particle dynamics.

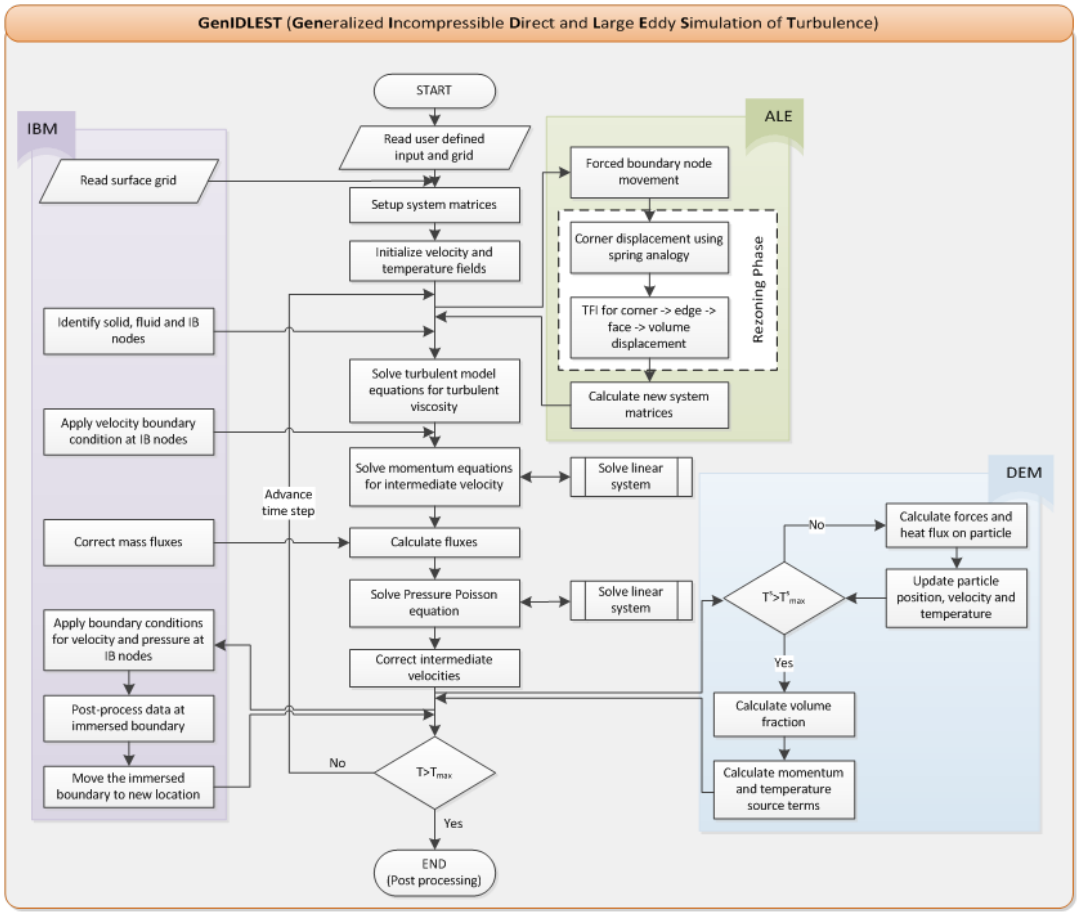

Figure B-1 : Schematic of GenIDLEST flow solver

For a CFD-DEM calculation involving non-spherical particles the model requires additional information about the geometry. In GenIDLEST, there are provisions to supply information about the particles in an input file, formatted in a manner suitable for accurate processing by the solver. A non-spherical particle's geometry is defined by its dimensional semi

major axis lengths and the squareness parameters. Since the geometry definition employed in the current model utilizes a discrete function representation to visualize the particles, the number of surface vertices on the particle's surface has to be specified in the input file. The discrete function representation used in the current framework assumes that the number of surface vertices along the z-axis is equal to that in the x-y plane. Hence the number of surface vertices in one slice of the non-spherical particle perpendicular to z-axis, is provided as input to the model. Denoting this value as 'N', the total number of vertices on the surface will be 'N^2'.

The computations done in GenIDLEST are non – dimensional. The position vectors, forces, moments, size of the particle, velocities and temperature are non-dimensional which dictates that the non-spherical particle parameters have to be non-dimensionalized with respect to the reference velocity and length scales.

The input parameters can be specified in a way such that each processor has its own distinct physical variables assigned to the particles. In GenIDLEST, the initial step is to read the input file on the master processor and send the necessary information to the other processors through a MPI send operation. Once the variables are received by the respective processors, the values of the axis lengths of the particles are non-dimensionalized with respect to the reference length.

The other aspect that has to be taken into account during the initialization of the simulation is the placement of the particles. The initial locations of the particles have to be regulated so that the particles do not overlap. An efficient way to accomplish this is to construct an imaginary sphere around a particle and as a consequence, ensure that no other particle's centroid falls inside this sphere. The imaginary sphere is constructed with its radius equal to the largest semi major axis length of the concerned particle. Also, the minimum length of each computational cell is at least 3 times the maximum semi major length of the particle. This ensures the particles do not extend

beyond one computational cell and the mapping between Cartesian coordinates and computational coordinates is smooth. Additionally, the computational domain consists of blocks of structured mesh. Each block is usually allocated to a processor for efficient distribution of load in a parallelized framework. In some cases, if the estimated parallel communication is relatively high, processors might be assigned more than one block to reduce the amount of time spent in message passing interfaces. Eventually, each particle is assigned a local and a global identification number (ID). The local ID refers to the identification number that a particle possesses with reference to the computational block it is in, while the global ID is unique and assigned with reference to all the blocks on all processors.

Spatial and temporal advancement of particles is done using the pertinent velocities arising from the CFD solver or that from the gravitational force. At any given time step, a 'search' operation is carried out for all the particles. Essentially, based on the information about the particle's location in the previous time step, a search operation is done to identify the new locations of the particles in the next time step. For instance, if a particle has been recorded to be at a discrete cell in the previous time step, a search operation is carried out to locate the particle in that cell or one of its neighboring cells. If the particle could not be found in either of these two instances of the search operation, a full global search in all computational cells of the domain is executed to locate the particle. If the particle has moved to another cell, belonging to another block of the domain computed on the same processor, a copy of the particle's requisite parameters is made on the same processor, so as to make it accessible for the new block. However, if the particle moves to a cell associated with another block on a different processor, the particle's information is passed to the new processor through MPI send/receive protocols. The buffer sizes for receiving particle information are assigned based on the data that needs to be sent. A reduction operation is carried

out on the size of the particle information being sent by each processor. The buffer size for receiving this information is allocated depending on the resulting sum of the size of data. Furthermore, if a particle cannot be found after the complete hierarchy of searches, it is deemed lost. On an additional note, there are four buffers that work on effective communication between processors, the details of which are elucidated in Table B-1. An interesting feature to note in the buffer descriptions is the mention of ghost particles. The particles in the nearest cells of the adjoining faces of another neighboring block are flagged as ghost particles for a given block. Ghost particles exist only for the purpose of carrying out the collision mechanics accurately without uncalled for changes being done to the particle's spatial and temporal advancement.

Table B-1 : Description of the buffers involved in message passing

Buffer Identifier	**Role**
Buffer 1	Responsible for transferring data of internal particles moving to another block on a different processor
Buffer 2	Responsible for copying data of internal particles moving to another block on the same processor
Buffer 3	Responsible for transferring data of ghost particles required for collision model computations with internal particles of another block on a different processor
Buffer 4	Responsible for copying data of ghost particles required for collision model computations with internal particles of another block on the same processor

Once the particle is found, the corresponding computational and Cartesian coordinates are updated. For a given processor, the particle information is unpacked after receiving and added to the list of internal particles already on that processor. The particle is assigned a new local ID but the global ID of the particle remains the same. The variables required to simulate the particle dynamics are deallocated. Determined from the new number of internal particles, the variable

arrays specific to particle dynamics and collision mechanics are reallocated and the previously existing internal and the newly acquired internal particles' data are stored accordingly. Furthermore, the variable arrays have to account for certain collision mechanics involving ghost particles. Therefore, the size of the particle arrays are reallocated to twice the number of internal and newly acquired particles in that particular block; this operation ensures there is sufficient memory available to work with the ghost particle collisions.

Following the reallocation of the variable arrays, the non-spherical collision model is activated. A collision model encompasses collision detection, collision resolution and the calculation of post collision velocities. As described in section 2.4, the collision detection hierarchy consists of three levels. The first level in the hierarchy labels particles in its own and the neighboring cells for a possible collision check, for a specified particle. The subsequent level further reduces the number of collision checks. Particles in the sphere of influence – imaginarily constructed with its radius specified by the largest dimension of the particle – are flagged and supplied as input to the collision detection algorithm. The third level consists of a neighbor list generation based on the evaluation of nearest particles in collision with the concerned particle. In this stage, the collision detection algorithm elucidated in section 2.4 is implemented and each particle is checked for collision with all the flagged particles, in its pertinent sphere of influence. If the particles are colliding, an eventual neighbor list is generated for that particle and the colliding particle's ID is stored in it. An array containing the local IDs of all particles colliding with a specified particle is created. During the collision detection stage, the surface vertices that are inside the other particle's surface or in contact with it are flagged and their global position vectors are stored in a separate array. This is done so as to facilitate the supplementary calculations encompassed in the collision modeling.

An interesting observation with the collision detection technique employed is that in some cases, collision may only be detected by one particle and not by the other due to the particle trajectories or an uneven distribution of surface vertices. For instance, due to the particle kinematics, an internal particle, say ‘P1’, on a block might detect collision with a ghost particle, say ‘P2’, from one of its neighboring blocks. As a consequence, when the collision is processed for P2 – now an internal particle for the respective block – by the corresponding processor, it fails to detect a collision. This could lead to an incorrect representation of particle dynamics as the velocity update post collisions is done only for internal particles in a given block, as shown in Figure B-2. Even though the particles are colliding and an update of the velocity is done for one of the particles, the other particle in the neighboring block would not experience an update owing to its failure to detect a collision. A cost inefficient way to solve this issue would be to pass the information of the particle detecting a collision – P1 - to the processer handling the neighboring block and replicate the collision, as if they were two internal particles. However, since the ghost particle’s information is already available during the collision detection stage, it is made use of to process the collision. For P2, particle P1 would be a ghost particle. Collisions are done with respect to local coordinate space of P1 and P2, on both processors, to ensure the dynamics are represented accurately. A schematic of the process is shown in Figure B-3.

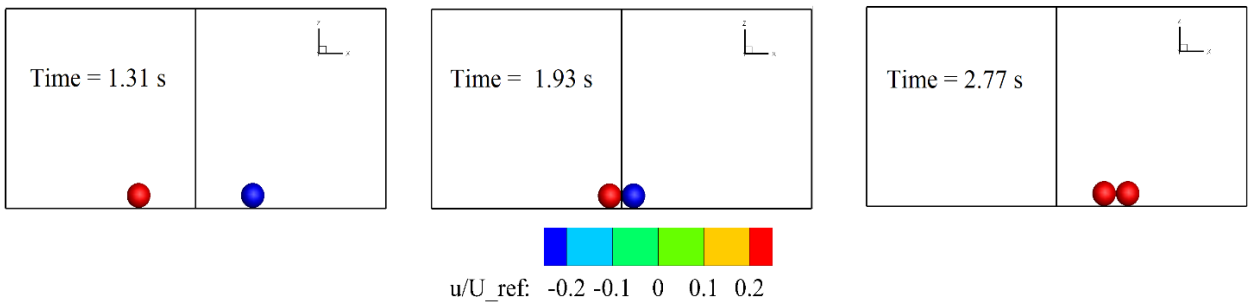

Figure B-2. Sequence of images showing the incorrect representation of ghost particle collision for two spheres

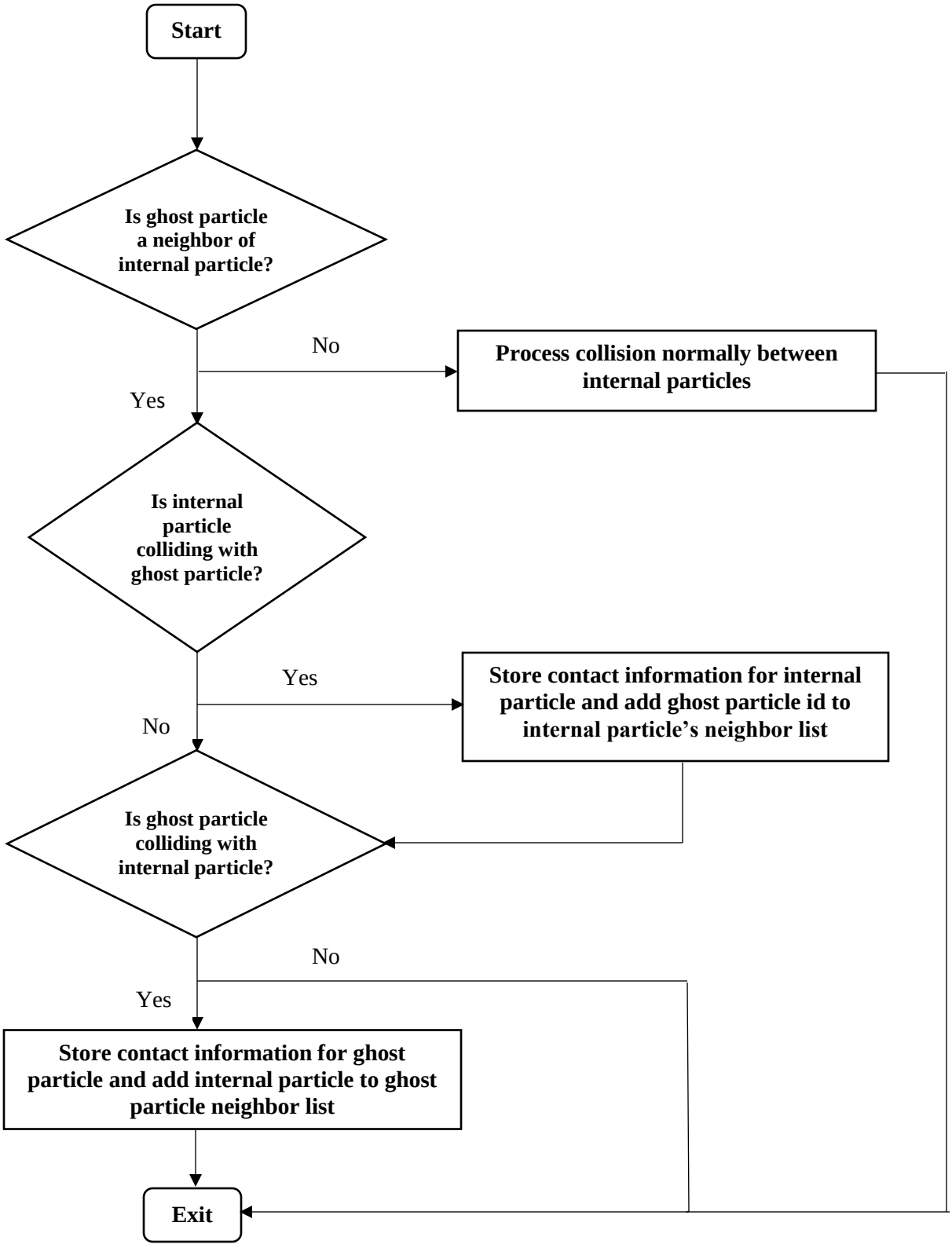

Figure B-3. Schematic of ghost - internal particle collision

The global position vectors are helpful when computing the contact normal and the mean contact point of the collision. The mathematical formulation of computing the contact normal and the contact normal are illustrated in section 2.4. Specifically, eq.2.19 to eq.2.31 give a comprehensive understanding of the methodology. Once the mean contact normal and the mean contact point are determined in global coordinate space, the information is stored in a vector and passed on to the respective models to resolve the collisions. Additionally, the maximum magnitude of penetration is calculated in this step. Since the information about the unit contact normal and the flagged surface vertices is available, the maximum magnitude of penetration and the ensuing relocation vector are computed and passed on to the collision resolution models.

Distinct flags are used to redirect the flow of the code to the appropriate collision model subroutines. If the hard sphere model is used in calculation, there is a possibility that the collision resolution might be carried out even in cases where the particles are moving away from the each other. To ensure the collisions are processed only when the particles approach each other, the relative velocity vector of the particles, before collision, is projected along the normal. If the resulting magnitude is negative, the collision is resolved. In the case of a soft sphere model, since reaction forces are used to resolve the collisions, the information about the relative velocities is inessential. Similarly, the particle-wall collision models are formulated based on this reasoning. The resulting velocities and forces are cumulatively added as calculated from each particle-particle binary collision or particle-wall collision.

The final velocity and force arrays are used to advance the particles spatially and temporally. If the hard sphere model is used, the velocities are directly updated. If the soft sphere model is used to resolve collisions, the forces stemming from the calculations are used to update the velocity arrays, comprising of the linear and rotational velocities. In an attempt to minimize

the computational effort and memory required for this process, the centroids of the particles are updated using the linear velocities and an Euler integration step. The rotational velocities are made use of to update the quaternions. The quaternion array contains four dimensional vectors characterizing the angular positions of particles. A conversion of quaternions to rotational matrices can be done and the positional vector of each surface vertex on a particle can be depicted, as explained in section 2.3. Whenever a visualization of the particles' motions is required, the quaternions and global position vector of the centroid of the respective particles are utilized to generate the complete surface of the geometry.