PACIFIC EARTHQUAKE ENGINEERING RESEARCH CENTER

Parallel Level-Set DEM (LS-DEM) Development and Application to the Study of Deformation and Flow of Granular Media

Peng Tan

Nicholas Sitar

Department of Civil and Environmental Engineering

University of California, Berkeley

PEER Report No. 2022/06

Pacific Earthquake Engineering Research Center
Headquarters at the University of California, Berkeley
December 2022

PEER 2022/06
December 2022

Parallel Level-Set DEM (LS-DEM) Development and Application to the Study of Deformation and Flow of Granular Media

Peng Tan

Nicholas Sitar

Department of Civil and Environmental Engineering
University of California, Berkeley

PEER Report No. 2022/06

Pacific Earthquake Engineering Research Center
Headquarters, University of California, Berkeley
December 2022

ABSTRACT

We present a systematic investigation of computational approaches to the modeling of granular materials. Granular materials are ubiquitous in everyday life and in a variety of engineering and industrial applications. Despite the apparent simplicity of the laws governing particle-scale interactions, predicting the continuum mechanical response of granular materials still poses extraordinary challenges. This is largely due to the complex history dependence resulting from continuous rearrangement of the microstructure of granular material, as well as the mechanical interlocking due to grain morphology and surface roughness. X-Ray Computed Tomography (XRCT) is used to characterize the grain morphology and the fabric of the granular media, naturally deposited sand in this study. The Level-Set based Discrete Element Method (LS-DEM) is then used to bridge the granular behavior gap between the micro and macro scale. The LS-DEM establishes a one-to-one correspondence between granular objects and numerical avatars and captures the details of grain morphology and surface roughness. However, the high-fidelity representation significantly increases the demands on computational resources. To this end a parallel version of LS-DEM is introduced to significantly decrease the computational demands. The code employs a binning algorithm, which reduces the search complexity of contact detection from $O(n^2)$ to $O(n)$, and a domain decomposition strategy is used to elicit parallel computing in a memory- and communication-efficient manner. The parallel implementation shows good scalability and efficiency.

High fidelity LS avatars obtained from XRCT images of naturally deposited sand are then used to replicate the results of triaxial tests using the new, parallel LS-DEM code. The result show that both micro- and macro-mechanical behavior of natural material is well captured and is consistent with experimental data, confirming experimental observation that the primary source of peak strength of sand is the mechanical interlocking between irregularly shaped grains. Specifically, triaxial test simulations with a flexible membrane produce a very good match to experimentally observed relationships between deviatoric stress and mobilized friction angle for naturally deposited sand. We then explore the viability of modeling dynamic problems with a new formulation of an impulse-based LS-DEM. The new formulation is stable, fast, and energy conservative. However, it can be numerically stiff when the assembly has substantial mass differences between particles. We also demonstrate the feasibility of modeling deformable structures in the rigid body framework and propose several enhancements to improve the convergence of collision resolution, including a hybrid time integration scheme to separately handle at rest contacts and dynamic collisions. Finally, we extend the impulse-based LS-DEM to include arbitrarily shaped topographic surfaces and exploit its algorithmic advantages to demonstrate the feasibility of modeling realistic behavior of granular flows. The novel formulation significantly improves performance of dynamic simulations by allowing larger time steps, which is advantageous for observing the full development of physical phenomena such as rock avalanches, which we present as an illustrative example.

Key Words: Level-Set DEM (LS-DEM), parallel programming, modeling granular media, triaxial compression, rock falls, rock avalanches.

ACKNOWLEDGMENTS AND DISCLAIMER

We thank Prof. José Andrade and Dr. Reid Kawamoto for sharing their LS-DEM code and their assistance in mastering its contents. We also thank Dr. Michael Gardner and Dr. Estephan Garcia for their constructive advice in the early phases of the code development. Dr. Yuqing Gao kindly assisted in the formatting of this report.

Funding for this research was provided by the National Science Foundation under grant CMMI-1853056, the Edward G. and John R. Cahill Chair, the Berkeley-France Fund, the US Geological Survey Cooperative Agreement G17AC00443, and the Pacific Earthquake Engineering Center (PEER), solicitation TSRP 2018-01.

CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS AND DISCLAIMER v

TABLE OF CONTENTS vii

LIST OF TABLES xi

LIST OF FIGURES xiii

1 INTRODUCTION 1

1.1 Motivation and Background 1

1.2 Report Structure 3

2 PARALLEL IMPLEMENTATION OF LS-DEM WITH HYBRID MPI+OPENMP 5

2.1 Introduction 5

2.2 Implementation Steps 6

2.2.1 Contact Detection Complexity and Binning Algorithm 6

2.2.2 Domain Decomposition Strategy 6

2.2.3 Contact Detection and Resolution Algorithm 8

2.2.4 Search Complexity of the Binning Algorithm 9

2.2.5 Normal Contact Force Resolution 10

2.2.6 Tangential/Traction Force Resolution 14

2.2.7 Discrete Equations of Motion 15

2.3 Hybrid MPI+OpenMP Design Considerations 18

2.3.1 Data Abstraction and Motivation for the Use of Data Structure 19

2.3.2 Border/Ghost Layers Communications 19

2.3.3 Across-Block Migration 20

2.3.4 Dynamic Workload Distribution 21

2.3.5 Modeling a Flexible Membrane in DEM 21

2.3.6 Memory Management 23

2.3.7 Limitations of the Parallel Implementation 24

2.3.8 Programming Environment 25

2.4 Implementation Details and Experience 25
2.4.1 Parallel Implementation of the Membrane 25
2.4.2 Memory Consumption Minimization and Memory Leak Prevention 26
2.4.3 Code Simplification and Edge Case Avoidance 28
2.4.4 Computational Effort Reduction 29
2.4.5 Position Reasoning and Linked-List Data Structure 29
2.5 Comparison Between Different Implementations 31
2.5.1 Original Implementation of LS-DEM 31
2.5.2 Two Binning Algorithm Implementations 31
2.5.3 Border/Ghost Communication Algorithm 32
2.5.4 Iterate Over Bins vs. Iterate Over Grains 35
2.5.5 Across-Block Migration 35
2.6 Numerical Experiments 37
2.6.1 Numerical Experiments on a Small Dataset 37
2.6.2 Limitations (Strong Scalability) of Domain Decomposition Strategy 43
2.6.3 Weak Scalability of Domain Decomposition Strategy 45
2.7 Conclusions 48
3 LS-DEM MODELING OF NATURALLY DEPOSITED SAND IN TRIAXIAL COMPRESSION 49
3.1 Introduction 49
3.2 LS Model Configuration 49
3.2.1 Numerical Apparatus 49
3.2.2 Model Scaling 50
3.2.3 Model Parameters 51
3.3 Numerical Considerations 51
3.3.1 Numerical Integration Scheme 51
3.3.2 Flexible Membrane Modeling 56
3.3.3 Numerical Stability of Angular Velocity Integration 59
3.4 Parametric Study and Model Calibration 60
3.4.1 LS Avatar Node Density 60
3.4.2 Confining Pressure 62

3.4.3 Normal and Shear Stiffness Ratio . . . 62
3.4.4 Grain Shape . . . 63
3.4.5 Initial Void Ratio . . . 65
3.5 Virtual Triaxial Compression Test . . . 67
3.5.1 Macroscopic Frictional Behavior . . . 68
3.5.2 Evolution of Mean Coordination Number . . . 69
3.5.3 Shear Band Evolution . . . 70
3.5.4 Evolution of the Soil Fabric . . . 76
3.5.5 Strain-Controlled Cyclic Loading Tests . . . 77
3.5.6 Computational Effort . . . 79
3.6 Conclusions . . . 80
4 IMPULSE-BASED LS-DEM FOR DYNAMIC PROBLEMS . . . 83
4.1 Introduction . . . 83
4.2 Formulation of Impulse-Based Rigid Body Dynamics . . . 84
4.3 Impulse-Based LS-DEM Implementation . . . 89
4.3.1 Contact Model . . . 89
4.3.2 Discrete Equations of Rigid Body Motion in Impulse-Based LS-DEM . . . 89
4.3.3 Collision Resolution Algorithm . . . 90
4.3.4 Time Stepping Algorithm . . . 91
4.4 Numerical Challenges of the Impulse-Based LS-DEM . . . 95
4.4.1 Contact with Rigid Boundaries . . . 95
4.4.2 Modeling Deformable Structures . . . 96
4.4.3 Weighted Relative Velocity Implementation . . . 100
4.4.4 Poorly Reconstructed Avatars . . . 101
4.5 Parallel Implementation of Impulse-Based LS-DEM . . . 101
4.6 Numerical Tests . . . 104
4.6.1 Performance Speedups with Impulse-Based LS-DEM . . . 104
4.6.2 Efficiency Analysis of Parallel Impulse-Based LS-DEM . . . 110
4.6.3 Numerical Modeling of Rock Avalanches . . . 112
4.7 Conclusion . . . 113
5 CONCLUSIONS AND FUTURE WORK . . . 117
REFERENCES . . . 119

LIST OF TABLES

2.1 Serial fraction as a function of the number of processors for simulations with $4,763$ grains. 43

2.2 Runtime breakdown for simulations of $42,684$ grains in a $600 \times 600 \times 600$ domain, with bin size 50, simulations ran for $1,000$ steps, (time is in seconds). 44

2.3 Estimated workload for different numbers of processors, and the corresponding theoretical speed-up. N_{Lower}, N_{Upper}, N_{Middle}, N_{True}: lower bound, upper bound, mean, and measured speed-up. 45

2.4 Domain angularity parameters for studies of weak scalability 47

3.1 Unit conversion with scaling factor k (μm/pixel) 51

3.2 Microparameters used in LS-DEM of triaxial compression test 53

4.1 List of model parameters and values used for the simulations. 107

4.2 Comparison of CPU time and speed up between different methods. 108

LIST OF FIGURES

1.1 SEM image of the depositional fabric in a sand shoal in San Francisco Bay, California (courtesy ENGEO). 2

1.2 Mobilized friction angle and stiffness in triaxial tests on undisturbed and disturbed samples of shoal sand from the San Francisco Bay (after Garcia et al. (2022). . . . 2

2.1 Domain decomposition strategy with eight sub-domains, blue dots are boundary bins that require communication between processors. 7

2.2 Illustration of the neighbor search in binning algorithm with the computational effort reduced by half due to symmetry 10

2.3 (a) Interaction between two contacting grains, where $d_k^{j,i}$ denotes the scalar penetration of the k-th node on grain i to the geometry of grain j, $\hat{\mathbf{n}}_k^{j,i}$ is the unit normal of penetration $d_k^{j,i}$. (b) Contact forces between two grains are different. 12

2.4 (a) Flexible membrane representation, $\sim 15,000$ balls. (b) Confining pressure normal to faces. (c) Hexagonal patterns of bonded spheres. (d) Membrane-grain interaction. 22

2.5 Flowchart of parallel strategy for using an extra membrane processor to handle the membrane-grain interactions. 26

2.6 Flowchart of parallel strategy to assure that the membrane-grain interactions are handled the same way by all processors. 27

2.7 Grain search using linked list data structure 30

2.8 Flowchart of the naive binning algorithm where bins are basic elements in iteration. 33

2.9 Flowchart of the binning algorithm where the grains are basic elements in iteration. 34

2.10 The first border/ghost communication step; exchange left and right layers. 35

2.11 Border/ghost communication, step two; exchange up and bottom layers. 36

2.12 Schematic of data containers for basic grain information in across-block migration. 37

2.13 Arrangement of the data containers for contact history in the across-block migration. 38

2.14 Illustration of the mechanism of MPI_Alltoall(). 38

2.15 Comparison between the actual speed-up and the theoretical speed-up for a series of small numerical experiments to examine strong scalability. 39

2.16 Relationship between the run-time and the number of processors for a series of small numerical experiments to examine weak scalability. 40

2.17 Run time decomposition for a simulation with 4763 grains. 41

2.18 Corrected speed up relationship considering actual workload on processors, study of strong scalability. 46

2.19 Simulation times for problems of different sizes and a fixed workload for processors, study of weak scalability. 46

3.1 Isotropic consolidation (a) before consolidation; (b) after consolidation; (c) membrane deformation. 50

3.2 Grain assembly reconstruction from (a) low-resolution ($\sim 70,000$ grains, $9 \sim 10\mu m$/voxel); (b) high-resolution ($\sim 17,000$ grains, $4.3\mu m$/voxel). 52

3.3 The effect of reducing confining pressure due to change of scaling factor. 57

3.4 The influence of membrane sphere radius. 58

3.5 The effect of membrane stiffness K_m. 59

3.6 The influence of restricting coordination number to maintain numerical stability. . . 61

3.7 The influence of avater node density. 62

3.8 The influence of the confining pressure; the simulations with $\boldsymbol{\sigma}_c = 200$kPa and $\boldsymbol{\sigma}_c = 500$kPa ended earlier due to membrane distortion. 63

3.9 The influence of the normal to shear stiffness ratio, K_s/K_n. 64

3.10 Left: An assembly of spherical particles. Right: An assembly of LS reconstructed avatars. Both specimens have identical void ratio. 65

3.11 The influence of the internal friction coefficient of spherical grains on simulated stress-strain response. 66

3.12 The effect of void ratio, considering both the grain shape and reconstruction resolution fidelity. 67

3.13 Comparison of deviatoric stress, mobilized friction angle and volumetric strain between low- and high-resolution numerical model specimens and experimental data. 69

3.14 Evolution of coordination number during deviatoric loading. 70

3.15 Single shear band pattern obtained from high-resolution specimen between $0 \sim 15\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$. 72

3.16 Grains with rotation greater than the mean rotation by two standard deviations (2σ) obtained from high-resolution specimen between $0 \sim 15\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$. 73

3.17 Shear band pattern obtained from low-resolution specimen between $0 \sim 30\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$. 74

3.18 Force chain evolution obtained from low-resolution specimen between $0 \sim 30\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$, forces change directions when they pass through the shear band. 75

3.19 Shear band pattern obtained from low-resolution specimen between $0 \sim 30\%$ axial shortening, with $K_{nbb} = K_{sbb} = 100N/m$. 76

3.20 Force chain evolution in the low-resolution specimen model between $0 \sim 30\%$ axial shortening, with $K_{nbb} = K_{sbb} = 100N/m$. 76

3.21 Surface plots of the distribution density of deviatoric fabric tensor and the anisotropy factor of grain inside the shear bands. 78

3.22 Surface plots of the distribution density of deviatoric fabric tensor and the anisotropy factor of grain outside the shear bands. 78

3.23 Spherical histogram of grain inside the shear bands. 78

3.24 Spherical histogram of grain outside the shear bands. 79

3.25 Mobilized friction angle in two numerical specimens under strain-controlled loading and unloading cycles. 79

4.1 Collision between bodies b_A and b_B: $\mathbf{V}_A$ and $\mathbf{V}_B$ are linear velocities of the center of mass, $\boldsymbol{\omega}_A$ and $\boldsymbol{\omega}_B$ are angular velocities, $\mathbf{r}_A^i$ and $\mathbf{r}_B^i$ are the vectors from the center of mass of bodies to the contact point where the i-th impulse $\mathbf{P}^i$ applied, $\mathbf{u}_A^i$ and $\mathbf{u}_B^i$ are the velocities at the contact points, and $\mathbf{n}^i$ is the contact normal. 84

4.2 Energy conservative property of impulse-based collusion resolution, kinetic energy (blue) and elastic energy (red) during compression and separation phases. The coefficient of restitution is 0.5. 91

4.3 Left: A small grain is sandwiched by two larger grains. Right: A small grain is resting on rigid plane while colliding against a large grain. The collision resolution algorithm for these cases might take extremely large number of iterations. 95

4.4 Flowchart of the time stepping integration, post-collision velocities were corrected by checking grain-boundary contact, membrane sphere velocity was corrected after it advanced to the next time step to ensure impenetrability. 97

4.5 Collision time sequence: gray grain obtains post-collision velocity, temporarily translates to the position marked in yellow, subject to repulsive forces and correct velocities to satisfy impenetrability constraints. 98

4.6 Flexible membrane model: grains falling into a flexible net. 99

4.7 Demonstration of the rigid boundary-grain interaction and flexible membrane interaction with the impulse-based LS-DEM code. 100

4.8 Two scenarios resulting in numerical instability. Left : One avatar is trapped by another; Right : Small avatar is completely enclosed in another. 101

4.9 Illustration of contact islands (indicated in different colors) in which a group of bodies is associated via a contact chain which can span over several sub-domains. For example, two red bodies in a same group could influence each other via alternate paths. 103

4.10 Flowchart of parallelized impulse-based method. 106

4.11 Initial configuration of 1600 very high-resolution grains in $800 \times 800 \times 800$ domain for impulse-based method with large time steps. 107

4.12 Specimen height and spread after 8s settlement under gravity with varying time steps. 108

4.13 Simulation time breakdown as a function of time step. 109

4.14 Computing time for islands with different numbers of collisions, all measured for 5 time steps and 10 sub-iterations for rigid boundary correction. 110

4.15 $155,016$ low-resolution grains in $1,200 \times 1,200 \times 1,200$ domain to benchmark parallel impulse-based LS-DEM. 111

4.16 Simulation time breakdown for parallel, impulse-based LS-DEM, modeled with $155,016$ grains, with 1, 8, 27, 64, 216 processors, respectively. 111

4.17 (a) Entire digitized topography. (b) A zoomed in region marked in yellow. (c) A zoomed in region concatenated with a horizontal plane. 113

4.18 Rock avalanche runout onto a horizontal surface. 114

4.19 Rock avalanche runout onto a sloping surface. 114

4.20 Rock avalanche impact on a single protection net. 115

4.21 Rock avalanche impact on two protection nets in a series. 115

1 Introduction

1.1 MOTIVATION AND BACKGROUND

The importance and the influence of the geologic setting and of the depositional environment on the mechanical properties of granular deposits, sands, gravels, and silts has been long recognized in geotechnical engineering (e.g. Terzaghi, 1955). As such, numerous investigators have examined the influence of the depositional fabric on the mechanical properties of sands prepared in the laboratory and showed that sand samples prepared by different methods exhibit different mechanical response even when the sand is compacted to the same relative density (see e.g.: Oda, 1972; Mulilis et al., 1977; Ochiai and Lade, 1983; Lam and Tatsuoka, 1988; Ishihara, 1993; Zlatovic and Ishihara, 1997; Yoshimine et al., 1998; Vaid et al., 1999; Wood and Maeda, 2008). However, while very informative and providing important insights, samples prepared in the laboratory do not achieve the intimate packing of grains exhibited in naturally deposited sands, which were the focus of the study presented in this report.

The opportunity for the study of the fabric and of the mechanical properties of naturally deposited sand arose as a result of a geotechnical investigation of the liquefaction potential of the hydraulic fill on Treasure Island in the San Francisco Bay. The island was constructed by hydraulic filling over a natural, sand shoal in the San Francisco Bay in the late 1930's. The investigation revealed that the shoal sand had much higher liquefaction resistance than the hydraulic fill, pointing to the difference in the depositional fabric of the two deposits as the most likely explanation for the observed difference in liquefaction resistance. A scanning electron microscope (SEM) photograph of an undisturbed sample of the shoal sand (Figure 1.1) shows that the sand is tightly packed, and the sand grains are arranged in an interlocking fabric. Garcia et al. (2022) used X-Ray Computed Tomography (XRCT) to obtain detailed images of the grain morphology and of the depositional fabric of the shoal sand and performed a series of small-scale triaxial tests to characterize the mechanical properties of the sand. The results of the triaxial tests (Figure 1.2) show that the peak mobilized friction angle of the undisturbed sand, with its intact depositional fabric, is significantly higher than that of the disturbed sample, and it also exhibits a much higher initial stiffness.

This new data set consisting of XRCT images and triaxial test results provided the opportunity to model the micromechanical behavior of the naturally deposited material and to explore the available methodology. Specifically, Kawamoto et al. (2016) transformed an XRCT image of a granular object into a mathematical descriptor (an "avatar") of the grain using a Distance Regularized Level-Set Evolution (DRLSE) formulation proposed by Li et al. (2005) and developed

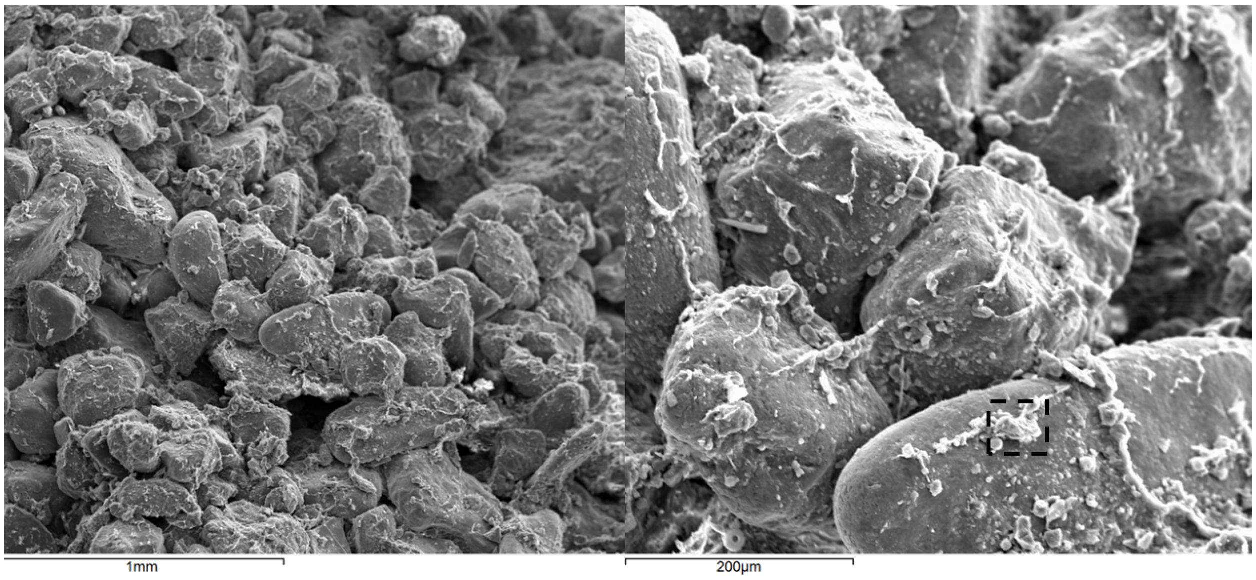

Figure 1.1: SEM image of the depositional fabric in a sand shoal in San Francisco Bay, California (courtesy ENGEO).

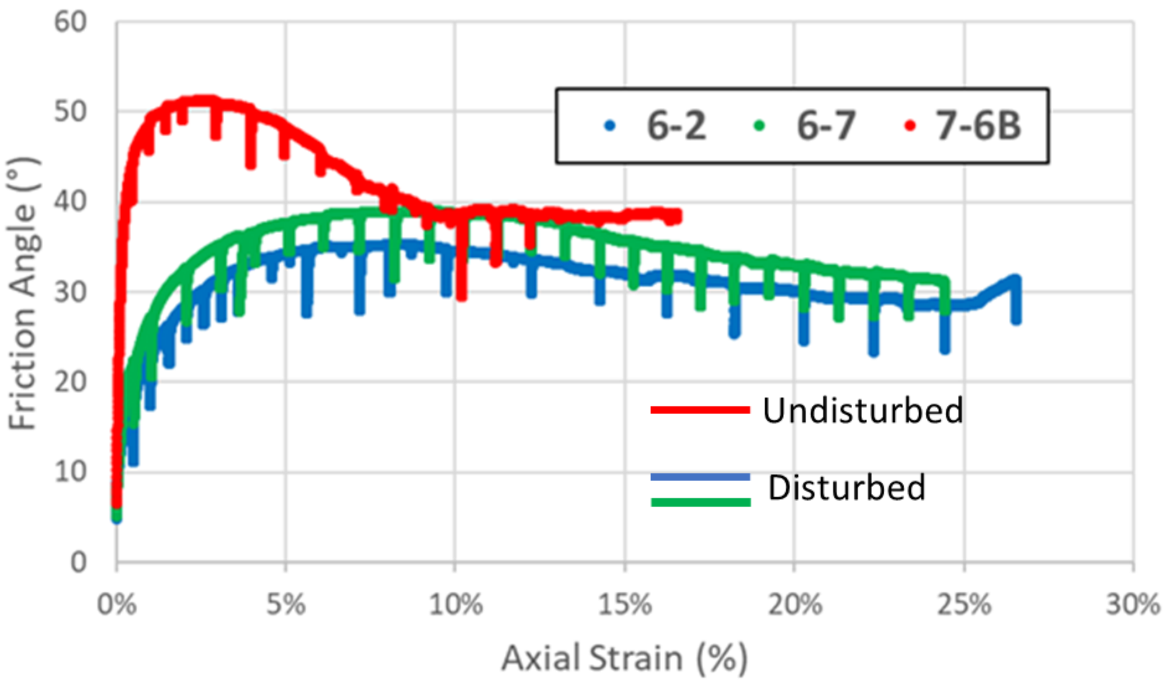

Figure 1.2: Mobilized friction angle and stiffness in triaxial tests on undisturbed and disturbed samples of shoal sand from the San Francisco Bay (after Garcia et al. (2022).

a Level-Set Discrete Element Method, LS-DEM. They used the code to demonstrate the significance of accurately modeling sand grain morphology and modeling its micromechanical behavior in a simulated triaxial test. The high fidelity of the reconstructed sand samples and the ability to faithfully model the micromechanical properties of the sand made it an attractive candidate for the work presented here. The challenge, however, was the large computational cost of the code developed by Kawamoto et al. (2018), which made it prohibitively expensive for analysis of data sets with large numbers of particles. Thus, the objectives of this study were to develop a parallel LS-DEM code capable of modeling large data sets, and to use the newly developed code to model the mechanical response of the naturally deposited shoal sand.

1.2 REPORT STRUCTURE

Chapter 2 focuses on the development of a LS-DEM code using a variant of the binning algorithm and a spatial domain decomposition strategy to model arbitrarily complex-shaped grains with a history-dependent contact model. The algorithm is optimized for the existing LS-DEM code, and majority of the parallel algorithms and the implementation details can also be ported to other disciplines. The performance-critical elements of the code are optimized for high performance and scalability, and the performance of the new parallel LS-DEM code is benchmarked in terms of speedup, efficiency, scalability, and granularity for problems of various sizes and using different amounts of computing resources.

Numerical triaxial compression simulations of naturally deposited sands reconstructed from XRCT images with different resolutions are presented in Chapter 3. A calibrated linear elastic model with the Coulomb friction yield criterion is used to simulate the frictional response. A significant conclusion reached in this study is that the primary source of mobilized strength is the mechanical interlocking between irregularly shaped grains. The results show that even the simplest contact model gives results consistent with the micro- and macromechanical responses observed in the experimental data, provided the microstructure of the grains and the grain-to-grain contacts are accurately captured. Triaxial test simulations with flexible membrane give results that capture the observed stress-strain and volumetric response, as well as the onset and growth of strain localization. The simulation results also match experimentally observed relationships between deviatoric stress and mobilized friction angle as axial shortening increases.

In Chapter 4 we present parallel LS-DEM algorithm and code reformulated in an impulse-velocity framework. The impulse-based method is attractive in large part because it circumvents the need for small time steps and saves significant amounts of computational effort for complex shaped objects. To improve convergence of the collision resolution, we modify an existing collision resolution algorithm and propose a novel time integration scheme. We then demonstrate the modeling of protection nets on arbitrarily shaped topography, showing that the impulse-based method can also be used to simulate the dynamics of deformable structures. The code is especially well suited for realistic modeling highly dynamic problems such as granular flows and rock avalanches containing thousands of particles, which we demonstrate with several example simulations. However, a limitation of the impulse-based method remains its inability to model a system of irregular, non-convex, non-uniform objects, in a highly confined, quasi-static environment where objects of very different shapes and sizes interact at numerous contact points and travel at low speeds.

Chapter 5 then provides a summary of the major findings and includes recommendation for future extensions of the numerical modeling techniques presented herein.

2 Parallel Implementation of LS-DEM with Hybrid MPI+OpenMP

2.1 INTRODUCTION

Grain morphology plays an essential role in determining the macroscopic properties of granular assemblies. Recent developments in the characterization of granular systems from X-ray computed tomographic (XRCT) images enable the study of grain morphology, soil fabric, and grain interactions through the numerical reconstruction of three-dimensional, complex-shaped avatars. The grain-level morphology information can be integrated into a numerical method such as the DEM to understand the link between granular material's macroscopic properties and its engineering behavior. Therefore, having a method that can model arbitrary grain shape is of paramount importance. However, large scale modeling requires significant computational effort and a DEM simulation on a single CPU machine might take several weeks to months to complete. To alleviate the computational cost of DEM, grain shapes are often simplified and non-physical parameters such as rolling resistance are introduced in contact models. Existing DEM approaches account for grain morphology mostly by clustering or clumping spheres (Garcia et al., 2009; Tamadondar et al., 2019; Wu et al., 2021), using a simplex or a polygon as a base geometry (Zhao et al., 2006; Zhao and Zhao, 2021), or by generating realistic grains based on the concept of Fourier descriptors or spherical harmonic functions (Garboczi, 2002; Taylor et al., 2006; Mollon and Zhao, 2012; Zhou et al., 2015). The first category is less appealing due to the lack of continuity in the curvatures and tangents. The other two approaches entail high computational expenses with narrow-phase contact detection and force calculations. The idea of using LS to encode object shape, as proposed by Osher and Sethian (1988) has drawn substantial attention in the realm of image segmentation, as it is capable to fully capture the complex morphology of natural granular material with high-fidelity. The strength of the LS-based algorithm is that it can track motion on a fixed Eulerian grid, which is handy when dealing with topological changes as the curve evolves. The fundamental idea ofthe LS-based image segmentation is to implicitly represent the boundaries of objects through grid-based interpolation from a space which is one dimension higher.

Coupling the LS representation of grain morphology with a DEM simulation is an attractive approach because the grain interactions are straightforward in the in the LS framework, as shown by Vlahinić et al. (2013). LS Discrete Element Method (LS-DEM) was introduced by Kawamoto (2018) who used a high-fidelity LS grain reconstruction to investigate the kinematic and mechanical behavior of a system of discrete sand grains Kawamoto (2018). However, modeling of arbitrarily complex grain shapes significantly increases the problem size thus making three-dimensional DEM impractical for many applications. To overcome this barrier, it is desirable to optimize the code to perform the large-scale computational work using modern supercomputers. In this section, we present a parallel, the three-dimensional LS-DEM algorithm for simulating complex-shaped granular grains.

Tan (2022) parallelized the LS-DEM code[1] with MPI (Message Passing Interface) using a variant of the binning algorithm and a spatial domain decomposition strategy to model arbitrarily complex-shaped grains with history-dependent contact model. Performance-critical implementation details are managed optimally to achieve high performance and scalability. Although these algorithms are tailored to fit into the existing LS-DEM code, most of the algorithms and implementation details can be migrated to applications in other disciplines. This chapter is organized as follows, the binning algorithm is first introduced in the context of DEM and the contact model and numerical integration scheme used in LS-DEM (Kawamoto et al., 2016; Kawamoto, 2018) are described for completeness. The MPI considerations and their implementation details are then presented. Finally, performance analyses of the modified code are presented, including speedup, efficiency, scalability, and granularity for different problem sizes (numbers of grains).

2.2 IMPLEMENTATION STEPS

2.2.1 Contact Detection Complexity and Binning Algorithm

A conventional DEM formulation involves an enormous amount of contact detection, and an efficient solution of large-scale discrete element problems relies upon a fast and efficient contact detection algorithm should be employed for significant scale problem. In terms of the types of neighbor search algorithm, there are three typical neighbor search algorithms with various time complexities: $O(n^2)$, coming from a n-by-n complete mapping of an assembly of grains; $O(n \log n)$, multi-level grids rooted from a tree-based algorithm (Jagadish et al., 2005; Muja and Lowe, 2009); $O(n)$, binning algorithm (Munjiza and Andrews, 1998; Williams et al., 2004) or the link cell algorithm (Grest et al., 1989). Yan and Regueiro (2018a) believed that the algorithms with the three different complexities $O(n^2)$, $O(n \log n)$, and $O(n)$ only affect the performance of neighbor search and have no influences on the contact resolution if non-trivial shaped grains are studied, and the overall performance is highly limited for complex-shaped grains. The most straightforward and commonly used approach is the binning algorithm or the link-cell algorithm.

2.2.2 Domain Decomposition Strategy

The idea of the binning algorithm is to place each grain into a bin using a hash on the grain's coordinates. Once the grains are sorted into bins, one can determine their proximity based on the fixed relationships between the bins. The bins are created through a domain decomposition strategy: the computational domain is first divided into sub-domains. Each MPI processor carries out calculations on its respective portions of the data domain. Then every sub-domain is further

[1] https://github.com/tanpeng1995/LS-DEM_Parallel_Benchmark.git

partitioned into many bins to hold the objects of interest, grains in the case considered herein. The size of bin is larger than the largest diameter of an assembly of grains such that any pair of grains that are separated by more than one bins are not considered to be in contact. An illustration of domain decomposition strategy can be found in Figure 2.1 which also denotes the notions of bin, block, and border layer. The main advantage of using the domain decomposition strategy is its high scalability even for a large number of processors, and its usability for both shared and distributed architecture machines (e.g. Gopalakrishnan and Tafti, 2013). The domain decomposition strategy is not limited to discrete element modeling. For example, Zohdi and Wriggers (1999) developed a similar technique for reducing the computational complexity of boundary value problems associated with structural analysis of bodies with an arbitrary external geometry and heterogeneous micro-structure. They partition and decouple the heterogeneous body into more computationally tractable, non-overlapping subdomains and the subdomain boundary conditions are approximated. Similarly, (McCallen et al., 2021) used domain decomposition for phsysics-based ground motion simulations, particularly for the finite-difference solution of the viscoelastic wave equation in both space and time. Overall, there has been a considerable interest in developing parallel DEM codes in recent years (Henty, 2000; Baugh Jr and Konduri, 2001; Washington and Meegoda, 2003; Maknickas et al., 2006; Walther and Sbalzarini, 2009; Chorley and Walker, 2010; Kačianauskas et al., 2010; Gopalakrishnan and Tafti, 2013; Amritkar et al., 2014), a comprehensive review is given in Yan and Regueiro (2018b) and Yan and Regueiro (2019).

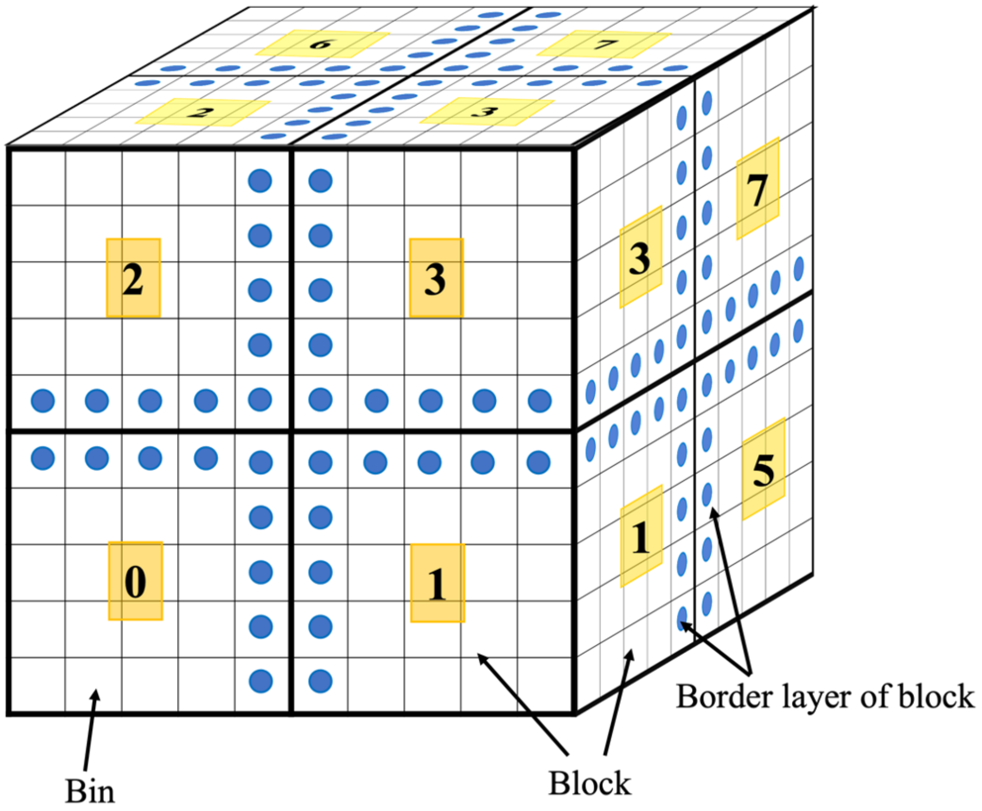

Figure 2.1: Domain decomposition strategy with eight sub-domains, blue dots are boundary bins that require communication between processors.

In general, most parallel implementations of DEM only deal with spheres rather than complex-shaped grains (Yan and Regueiro, 2018b). Although it is acceptable to sacrifice grain size and shape in the trade-off for a larger problem size in the prototype-scale simulations for engineering studies, overly simplified grains cannot reproduce the detailed micromechanical behavior of the grain assembly without capturing grain-level details as pointed out by Peters et al. (2009). However, modeling arbitrarily shaped particles typically demands several orders of magnitude greater computational effort. For example, the CPU demand for simulating 122 million spheres is equivalent to simulating 488k poly-ellipsoids when using the same inter-grain contact model (Yan and Regueiro, 2018b). This is also a major computational bottleneck in LS-DEM. The geometric basis of a grain is embedded in a LS table and discretized into hundreds of nodes as seeded on the surface. Each node on the surface of the particle is checked for contact in the force resolution step, which results in significantly greater computational cost than for spherical grains. Another complexity arises when considering history-dependent tangential behavior for granular materials. Modeling of arbitrarily complex grains and using a more sophisticated inter-grain contact model is not difficult in a sequential code. However, it becomes a lot more cumbersome in a parallel code, as tracking the loading-unloading-reloading path and contact histories is not straightforward. Consequently, the parallel domain decomposition algorithm has to be modified for different inter-grain contact models, grain shape complexity, boundary conditions, and computational granularity.

2.2.3 Contact Detection and Resolution Algorithm

The LS reconstructed avatars are represented by hundreds of nodes, and the volume, mass, and moments of inertia are computed by counting the number of voxels inside the avatars. In terms of computational cost, the signed distance function can be calculated efficiently using a marching method (Sethian, 1996). This is a one-time cost in constructing a rigid body model. For efficiency, the object is stored with the center of mass at the origin and the axes aligned with the principal axes of inertia resulting in a diagonal inertia tensor to simplify many calculations. The interactions between two implicit surfaces are found by checking the sign of nodes in the signed-distance function of the other. However, this is not sufficient to detect all collisions, as edge-face collisions are missed when both edge vertices are outside the implicit surface. Nevertheless, since these errors are proportional to the edge length, they can be ignored in a well resolved mesh with sufficient node density.

The particle overlap model is the most commonly used model which determines the contact force from the separation distance between particles and material properties, the assumption is that the particles are spheres (Zohdi, 2017). Typically, the contact detection and resolution algorithm phases are the primary computational bottlenecks, especially when simulating complex-shaped grains with hundreds of nodes per grain. If nonlinear history-dependent mechanical models are used, this stage becomes even more computationally demanding. The contact identification and resolution algorithm phases in most DEM codes programs are divided into two sub-phases: nearest neighbor search (or spatial resolution) and contact resolution. A neighbor search phase identifies or estimates objects that are close to the target object using an easy-to-model approximate geometry, such as a bounding box or a bounding sphere. The LS-DEM code adjusts to the following:

$$\|\mathbf{c}^i - \mathbf{c}^j\| > r^i + r^j \tag{2.1}$$

Where $\mathbf{c}^i$ is the position of the mass center of grain i and r^i is the equivalent radius of grain i. Following that, the contact resolution phase employs a geometric representation of each body. At the resolution phase, the LS reconstructed avatars comprise hundreds of discretized nodes, each of which must be checked to the surface of a neighboring avatar. This stage is performed sequentially and has a minimal vectorization potential due to the explicit nature of DEM modeling. This is the cost of a three-dimensional DEM code capable of simulating grains of any shape. In this aspect, resolving the contact between two arbitrarily shaped grains is significantly more computationally expensive than resolving the contact between two grains having basic geometrical representations, such as spheres. Due to the necessity for numerical precision and resilience, it frequently increases the floating-point operations by many orders of magnitude. We used the binning algorithm to limit the scope of the neighbor search; the binning algorithm's fundamental idea is to improve spatial locality, which means that a discrete grain does not need to check all other grains but only those in its immediate vicinity. However, the total computing benefit of neighbor search may be negligible for complex-shaped grains, as neighbor search accounts for a small fraction of floating-point operations. As a result, the most computationally intensive portion of the LS-DEM is the force resolution phase, rather than the neighbor detection phase which is the major bottleneck for conventional DEM simulating disks or spheres. We later show that the idea of binning algorithm is well compatible with domain decomposition since many parallel implementations consider bin as a base unit.

2.2.4 Search Complexity of the Binning Algorithm

The binning method assumes that grain interactions occur only when two grains come into contact. The node-to-surface contact is solved explicitly and is not vectorized. The computational cost of an n-by-n naive search algorithm is well-known to be $O(n^2)$ for a generalized N-body problem (Gray and Moore, 2000). When the non-contact pairs of objects are excluded, the computational complexity is reduced from $O(n^2)$ to $O(n)$ for the binning algorithm or to $O(n \log n)$ for a tree algorithm. In this particular problem, the idea of binning is to place each grain into a bin of prescribed size hashed on the grain's coordinates. The cutoff distance for DEM is chosen to be at least the largest equivalent diameter of grains so that any two grains which are at least one bin distance away will not interact. Once the grains are sorted into bins, one can evaluate the spatial proximity based solely on the fixed relationships of the bins. As shown in Figure 2.2, the grain filled in yellow and marked in orange line represent the one of interest, and blue grains represent those that require check for detection. As can be seen, the computational effort for neighbor detection substantially decreases through binning, especially for spherical particles (Zohdi, 2004a, 2010, 2012). Due to the 3D shape of grains, i.e., grains cannot be clustered arbitrarily dense, we can assume the average number of grains in each bin is b, then the computational complexity is reduced to $O(n)$ with a small constant b. However, it can be shown that the $O(n)$ neighbor search algorithm might be inferior to $O(n^2)$ for computing a very small number of grains for two reasons. 1) Both shared-memory and distributed-memory encounter synchronization and communication overheads, which are inevitable and always retards the performance, primarily when the performance is governed by communication bandwidth and latency. 2) The overall performance improvement resulting from a parallel algorithm might be highly limited for complex-shaped grains, because the algorithm only affects the performance of the neighbor estimate rather than the contact resolution. The contact resolution step takes up a large fraction of floating-point operations in the computation. For these reasons, it is acceptable and advisable to use the serial approach when the number of grains is relatively small. Of course, serial implementation becomes extremely inefficient as the number of grains increases.

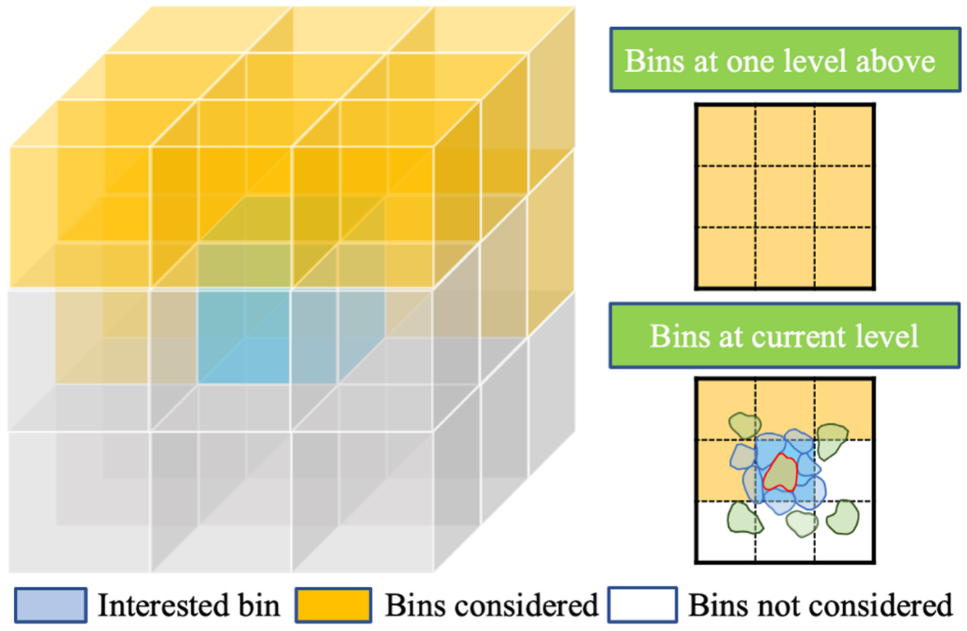

Figure 2.2: Illustration of the neighbor search in binning algorithm with the computational effort reduced by half due to symmetry

2.2.5 Normal Contact Force Resolution

The LS based shape representation and contact algorithms unique to LS-DEM were developed by Kawamoto et al. (2016) and are followed herein. Contact in LS-DEM is handled through iterating node-to-surface contact algorithm, whereby nodes are seeded onto the surface of each grain and grain shape is implicitly embedded in its LS grid value table. This representation is similar to a triangulated surface mesh except that LS-DEM does not store connectivity information between nodes and therefore does not consider edge-surface collision. The density of nodes on a given grain is a matter of choice and is entirely up to the designer. The number of nodes seeded onto a grain has no effect on the underlying geometry but does have an effect on the computational complexity associated with force resolution. With grain refining, extremely high-fidelity reconstruction frequently requires unaffordable computation time. Lim et al. (2014) demonstrate that seeding with a maximum node-to-node spacing less than $d/10$, where d is the grain diameter, is sufficient to capture grain morphology and a further increase in nodal densities has a minor impact on behavior.

Contact is determined by comparing each node of a master grain to a slave grain for penetration. By embedding the grain in a three-dimensional Cartesian grid with a value indicating the signed distance to the nearest grain surface, the grain surface is implicitly defined by a set of nodes with zero LS value. This framework is quite convenient as it is amenable to calculating the forces between grains with the commonly used penalty-based method. As shown in Figure 2.3, $d_k^{j,i}$ denotes the scalar penetration of the k-th node on grain i to the geometry of grain j; ϕ^j is the LS function of grain j; $\mathbf{p}_k^j$ is the position of k-th node on grain i considered in grain j's coordinate; $\hat{\mathbf{n}}_k^{j,i}$ is the unit normal direction of penetration $d_k^{j,i}$. The amount of penetration can be computed through interpolation from grid values near $\mathbf{p}$; any order of interpolation can be used, and linear interpolation was used here for simplicity and speed.

Note that one property of LS accommodated with signed distance function is that the gradient of a point of LS is unity at that point. However, due to the LS function's discrete nature, the magnitude of $\nabla\phi^j(\mathbf{p}_k^j)$ is very close but not equal to unity and therefore it is normalized. If at least one node $\mathbf{p}_k^j$ of master grain i is penetrating a slave grain j, then the two grains are considered to be in contact and inter-grain forces are computed. This process is detailed in Algorithm 1:

The current code adopts the linear elastic contact model. Thus, the normal contact force contributed from the node $\mathbf{p}_k^i$ on grain i is:

$$\mathbf{F}_{n,k}^i = \begin{cases} -k_n d_k^{j,i}\hat{\mathbf{n}}_k^{j,i} & d_k^{j,i} < 0 \\ 0 \quad \text{else} \end{cases} \tag{2.2}$$

Where K_n is the normal contact stiffness. By action and reaction, the contribution of contact normal force $\mathbf{F}_{n,k}^j$ from the node $\mathbf{p}_k^i$ on grain j is:

$$\mathbf{F}_{n,k}^j = -\mathbf{F}_{n,k}^i \tag{2.3}$$

The moment $\mathbf{M}_{n,k}^i$ contributed by the normal contact force $\mathbf{F}_{n,k}^i$ at the node $\mathbf{p}_k^i$ on grain i is:

$$\mathbf{M}_{n,k}^i = (\mathbf{p}_k^i - \mathbf{c}^i) \times \mathbf{F}_{n,k}^i \tag{2.4}$$

Where $\mathbf{c}^i$ is the centroid of grain i. Similarly, the moment $\mathbf{M}_{n,k}^j$ contributed by the normal contact force $\mathbf{F}_{n,k}^j$ at the node $\mathbf{p}_k^i$ on grain j is:

$$\mathbf{M}_{n,k}^j = (\mathbf{p}_k^i - \mathbf{c}^j) \times \mathbf{F}_{n,k}^j \tag{2.5}$$

It is important to keep in mind that the contact forces between two grains vary slightly depending on which grain is selected as the master grain. This is because the k-th node on the master grain i might penetrate the slave grain j, while there does not exist a corresponding node on the slave grain j penetrating master grain i due to the discrete nature of LS geometry representation. This does not influence the serial implementation, as the force resolution phase is always iterated from a small index to a large index. Nevertheless, the index order will change after adding or deleting the migrated grains from the bins, thus we always consider the grain with the smaller index as the master grain in the force resolution in the parallel implementation.

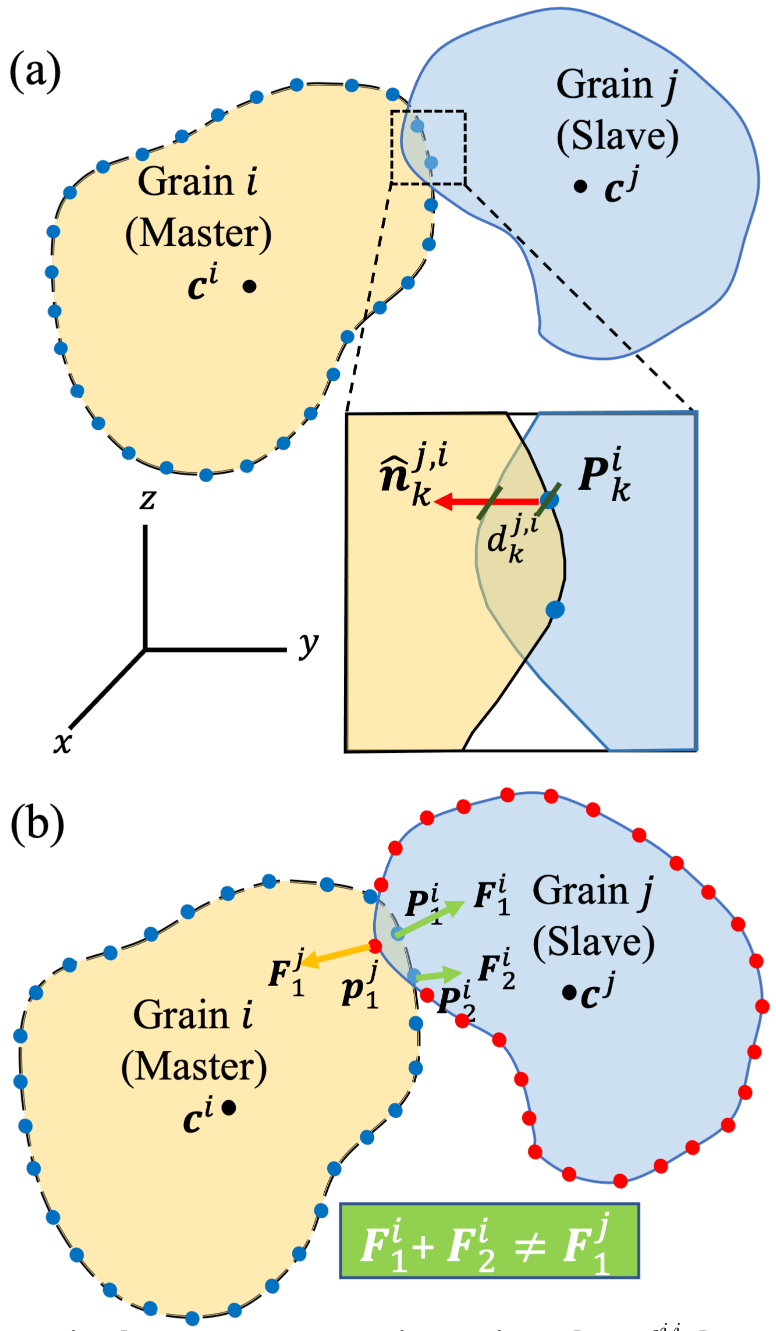

Figure 2.3: (a) Interaction between two contacting grains, where $d_k^{j,i}$ denotes the scalar penetration of the k-th node on grain i to the geometry of grain j, $\hat{\mathrm{n}}_k^{j,i}$ is the unit normal of penetration $d_k^{j,i}$. (b) Contact forces between two grains are different.

Algorithm 1 findPenetrationDirection

```
INPUT: grain, point $\mathbf{P}$
OUTPUT: flag, penetration depth $d$, contact normal in principal frame $\tilde{\mathbf{n}}$
/* extract LS values nearby $\mathbf{P}$ */
/* $\mathbf{P}_x$, $\mathbf{P}_y$, $\mathbf{P}_z$ are coordinates of $\mathbf{P}$ */
$x_0 = \textbf{floor}(\mathbf{P}_x)$, $y_0 = \textbf{floor}(\mathbf{P}_y)$, $z_0 = \textbf{floor}(\mathbf{P}_z)$
$x_1 = \textbf{ceil}(\mathbf{P}_x)$, $y_1 = \textbf{ceil}(\mathbf{P}_y)$, $z_1 = \textbf{ceil}(\mathbf{P}_z)$
/* function getGridValue looks up the LS table to extract value */
$P_{000} = \textbf{getGridValue}(x_0, y_0, z_0)$
$P_{001} = \textbf{getGridValue}(x_0, y_0, z_1)$
$P_{010} = \textbf{getGridValue}(x_0, y_1, z_0)$
$P_{011} = \textbf{getGridValue}(x_0, y_1, z_1)$
$P_{101} = \textbf{getGridValue}(x_1, y_0, z_1)$
$P_{100} = \textbf{getGridValue}(x_1, y_0, z_0)$
$P_{110} = \textbf{getGridValue}(x_1, y_1, z_0)$
$P_{111} = \textbf{getGridValue}(x_1, y_1, z_1)$
/* find penetration $d$ via linear interpolation */
$P_x = P_{100} - P_{000}$
$P_y = P_{010} - P_{000}$
$P_z = P_{001} - P_{000}$
$P_{xy} = -P_x - P_{010} + P_{110}$
$P_{xz} = -P_x - P_{001} + P_{101}$
$P_{yz} = -P_y - P_{001} + P_{011}$
$P_{xyz} = P_{xy} - P_{001} - P_{101} - P_{011} + P_{111}$
$\Delta x = \mathbf{P}_x - x_0$
$\Delta y = \mathbf{P}_y - y_0$
$\Delta z = \mathbf{P}_z - z_0$
$d = P_{000} + P_x \cdot \Delta x + P_y \cdot \Delta y + P_z \cdot \Delta z + P_{xy} \cdot \Delta x \cdot \Delta y + P_{xz} \cdot \Delta x \cdot \Delta z + P_{yz} \cdot \Delta y \cdot \Delta z + P_{xyz} \cdot \Delta x \cdot \Delta y \cdot \Delta z$
if $d < 0$ then
    flag = True
/* take derivative respect to $d$ obtain contact normal $\tilde{\mathbf{n}}$ */
$\tilde{\mathbf{n}} = P_x + P_{xy} \cdot \Delta y + P_{xz} \cdot \Delta z + P_{yz} \cdot \Delta y \cdot \Delta z + P_{xyz} \cdot \Delta y \cdot \Delta z$
return $d$, $\tilde{\mathbf{n}}$
```

To maximize efficiency, the grain is stored with the center of its mass at the origin and the axes aligned with the inertia primary axes, resulting in a diagonal inertia tensor that simplifies many calculations. As a result of the rigid body assumption, the grain's LS function is never altered. When a contact is computed, the nodes $\mathbf{p}_k^i$ of grain i are temporarily relocated into the reference configuration of the grain j's LS function. The contact forces and moments are then determined (in the reference configuration of grain j) and translated back to the global frame.

2.2.6 Tangential/Traction Force Resolution

Unlike the normal forces, the tangential forces at grain contacts can be either history dependent or independent. A history-independent tangential contact model is one in which the tangential force is always proportional to its normal equivalents. The original LS-DEM code (Kawamoto et al., 2016) uses a history-dependent Coulomb friction model similar to that in Cundall and Strack (1979). This model requires that contact histories accompany migrating grains because the tangential displacements are computed in increments until the two objects are separated. Consequently, the grains must be considered individually for cross-block migration. Typically, a history-dependent tangential model is required to simulate physical experiments, since highly simplified contact models are incapable of accurately capturing the physical properties of frictional granular material because they do not account for shear history and do not simulate non-linearity. While the Coulomb friction model is the simplest, more sophisticated models incorporate the rate of shearing, and incorporating such models may result in improved results.

For a given node $\mathbf{p}_k^i$, frictional forces and the related moments only exist if $\mathbf{F}_{n,k}^i \neq 0$. The relative velocity $\mathbf{v}_k$ of node $\mathbf{p}_k^i$ to grain j is:

$$\mathbf{v}_k = \mathbf{v}^i + \boldsymbol{\omega}^i \times (\mathbf{p}_k^i - \mathbf{c}^i) - \mathbf{v}^i - \boldsymbol{\omega}^j \times (\mathbf{p}_k^i - \mathbf{c}^j) \tag{2.6}$$

Where $\mathbf{v}^i$, $\mathbf{v}^j$, $\boldsymbol{\omega}^i$, $\boldsymbol{\omega}^j$ are translational and angular velocities of grain i and grain j. The incremental shear displacement $\Delta\mathbf{s}_k$ is then:

$$\Delta\mathbf{s}_k = [\mathbf{v}_k - (\mathbf{v}_k \cdot \hat{\mathbf{n}}_k^{j,i})\hat{\mathbf{n}}_k^{j,i}] \cdot \Delta t \tag{2.7}$$

The shear force $\mathbf{F}_{s,k}^i$ on grain i contributed by node $\mathbf{p}_k^i$ is updated as such:

$$\mathbf{F}_{s,k}^i = \mathbf{Z}\mathbf{F}_{s,k}^i - k_s \Delta\mathbf{s}_k \tag{2.8}$$

Where $\mathbf{Z}$ is the rotation operation that rotates the normal vector $\hat{\mathbf{n}}_k^{j,i}$ at the previous time step to the normal vector at the current time step and k_s is the shear contact stiffness. This step is necessary because the relative orientation of the two grains would change between time steps. In the code presented herein, we use Rodrigues's rotation formula (Murray et al., 1994):

$$\mathbf{v}_{rot} = \cos\theta + (1 - \cos\theta)(\mathbf{k} \cdot \mathbf{v})\mathbf{k} + \sin\theta \mathbf{k} \times \mathbf{v} \tag{2.9}$$

Where θ is the angle between the interested vector in two time steps, and $\mathbf{k}$ is cross product between normal vector at the current and previous time step. The Coulomb friction law dictates $\mathbf{F}_{s,k}^i$ be capped at a fraction of the normal force $\mathbf{F}_{n,k}^i$:

$$\mathbf{F}_{s,k}^i = \frac{\mathbf{F}_{s,k}^i}{\|\mathbf{F}_{s,k}^i\|} \min\left(\|\mathbf{F}_{s,k}^i\|, \mu\|\mathbf{F}_{s,n}^i\|\right) \tag{2.10}$$

Where μ is the inter-grain friction coefficient. By action and reaction:

$$\mathbf{F}^{j}_{s,k} = -\mathbf{F}^{i}_{s,k} \tag{2.11}$$

The moment $\mathbf{M}^{i}_{s,k}$ contributed by node $\mathbf{p}^{i}_{k}$'s shear force on grain i is:

$$\mathbf{M}^{i}_{s,k} = (\mathbf{m}^{i}_{k} - \mathbf{c}^{i}) \times \mathbf{F}^{i}_{s,k} \tag{2.12}$$

Similarly, the $\mathbf{M}^{j}_{s,k}$ contributed by node $\mathbf{p}^{i}_{k}$'s shear force on grain j is:

$$\mathbf{M}^{j}_{s,k} = (\mathbf{m}^{i}_{k} - \mathbf{c}^{j}) \times \mathbf{F}^{j}_{s,k} \tag{2.13}$$

In the end, the total contact force on grain i is found by summing all nodal contact forces:

$$\mathbf{F}^{i}_{rot} = \sum_{k=1}^{N} (\mathbf{F}^{i}_{n,k} + \mathbf{F}^{i}_{s,k}) \tag{2.14}$$

Where N is the number of nodes on grain i. By action and reaction:

$$\mathbf{F}^{j}_{rot} = -\mathbf{F}^{i}_{rot} \tag{2.15}$$

The total contact moment on each grain is found by summing all nodal contact moments:

$$\begin{aligned} \mathbf{M}^{i}_{rot} &= \sum_{k=1}^{N} (\mathbf{M}^{i}_{n,k} + \mathbf{M}^{i}_{s,k}) \\ \mathbf{M}^{j}_{rot} &= \sum_{k=1}^{N} (\mathbf{M}^{j}_{n,k} + \mathbf{M}^{j}_{s,k}) \end{aligned} \tag{2.16}$$

The implementation of the grain interaction model is shown in Algorithm 2 below.

2.2.7 Discrete Equations of Motion

The scheme to update the center of mass and nodes of each grain implemented by Kawamoto et al. (2016) was adapted from work by Lim and Andrade (2014). The locations, forces, and velocities of the grains are known at the end of each time step, allowing the grain motion to be explicitly updated via Newton's law:

$$ma_i + Cv_i = F_i \tag{2.17}$$

Algorithm 2 findInterGrainForceMoment

```
INPUT:grain $A$, grain $B$
OUTPUT: grainForce $\mathbf{F}$, grainMoment M
for int $i = 0$; $i <$ num_nodes; $i = i + 1$ do
    $\mathbf{u^{AB}} = \mathbf{u_A} - \mathbf{u_B}$, where $\mathbf{u_A}$ is the mass center of grain $A$
    if $|\mathbf{u^{AB}}| < r_A$, $r_A = \max_{k\in\mathcal{N}}\{|\mathbf{u^k_A}|\}$, where $\mathbf{u^k_A}$ is the vector from $k$-th node of grain $A$ to mass center then
        /* rotate $\mathbf{u^i_A}$ to the principal frame of grain $B$'s LS grid */
        $\tilde{\mathbf{u}}^i_A = \mathbf{R}^T\mathbf{u^i_A} + \mathbf{c_B}$, where $R$ is the operator rotate vector from principal frame to global frame
        /* check if $\tilde{\mathbf{u}}^i_A$ penetrates into grain $B$ */
        flag, $d^i$, $\tilde{\mathbf{n}}^i$ = findPenetration($\tilde{\mathbf{u}}^i_A$, grain B)
        if flag then
            /* rotate contact normal $\tilde{\mathbf{n}}^i$ to global frame */
            $\mathbf{n^i} = \mathbf{R}\tilde{\mathbf{n}}^i$
            /* Compute spring force at normal direction */
            $\mathbf{f^i_n} = k_n \cdot d^i \cdot \mathbf{n^i}$
            $\mathbf{F} = \mathbf{F} + \mathbf{f^i_n}, \quad \mathbf{M} = \mathbf{M} + \mathbf{u^i_A} \times \mathbf{f^i_n}$
            /* Compute relative velocity of two grains at $\mathbf{u^i_A}$ */
            $\mathbf{v^i_{AB}} = \mathbf{v_A} - \mathbf{v_B} + \omega^\mathbf{i}_\mathbf{A}\mathbf{u^i_A} - \omega^\mathbf{i}_\mathbf{B}\mathbf{u^i_B}$
            /* Compute friction increment in tangential direction */
            $\mathbf{f^{t+\Delta t,i}_t} = \mathbf{Z}\mathbf{f^{t,i}_t} - \mathbf{v^i_{AB}} \cdot k_t \cdot \Delta t$, where $\mathbf{Z}$ is operator rotate past shear force direction to current direction.
            /* Check Coulomb's friction criterion is satisfied */
            $f^i = \min\{|\mathbf{f^i_t}|, \mu|\mathbf{f^i_n}|\}$
            $\mathbf{f^i_t} := f^i \cdot \mathbf{f^i_t}/|\mathbf{f^i_t}|$
            $\mathbf{F} = \mathbf{F} + \mathbf{f^i_t}, \quad \mathbf{M} = \mathbf{M} + \mathbf{u^i_A} \times \mathbf{f^i_t}$
    return F, M
```

Where $i = 1, 2, 3$ in three dimensions, m is the mass of the grain, $C = \xi m$ is damping that proportionally scales the linear velocity v_i, with ξ $[1/T]$ being the global damping parameter (Walton and Braun, 1993; Lim and Andrade, 2014). The linear acceleration is given by a_i and is related to the resultant force F_i. Centered finite-difference integration scheme is used to integrate the translational components of motion:

$$v_i^{n+\frac{1}{2}} = \frac{1}{1 + \frac{\xi\Delta t}{2}}[(1 - \frac{\xi\Delta t}{2})v_i^{n-\frac{1}{2}} + \frac{\Delta t}{m}F_i] \tag{2.18}$$

$$x_i^{n+1} = x_i^n + \Delta t v_i^{n+\frac{1}{2}} \tag{2.19}$$

This scheme is second-order explicit, which is conditionally stable, and there is a family of trapezoidal integration schemes in various forms. Based on the variable metric, the scheme becomes implicit, unconditionally stable, and the coupled system of equations is solved using an adaptive iterative scheme. (Zohdi, 2003, 2004b, 2007, 2013). For complex-shaped objects, the rotational components of motion must also be integrated, and the time derivatives of the angular accelerations in the principal frame are given by Euler's equations of motion.

$$\dot{\boldsymbol{\omega}} = (M - \boldsymbol{\omega} \times (I\boldsymbol{\omega}) - \xi I\boldsymbol{\omega})/I \tag{2.20}$$

Where $\dot{\boldsymbol{\omega}}$ is the angular acceleration, $\boldsymbol{\omega}$ is the angular velocity, I is the (diagonal) moment of inertial tensor in the principal body-fixed frame, and M is the torque vector in the principal body-fixed frame. The Euler equations are nonlinear due to the presence of angular velocities products on both sides. Therefore, to appropriately integrate the rotational components of motion, a predictor-corrector procedure is recommended:

(1) Estimate the angular velocities at the current time step by assuming constant angular acceleration for an additional half step.

$$\omega_i'^n = \omega_i^{n-\frac{1}{2}} + \frac{1}{2}\Delta\omega_i^{n-1} \tag{2.21}$$

where $\Delta\omega_i^{n-1} = \alpha_i^{n-1}\Delta t$.

(2) Calculate angular velocity predictor by using the estimates as mentioned earlier.

$$\begin{aligned}
\Delta\omega_1'^n &= \Delta t[M_1^n + \omega_2'^n\omega_3'^n(I_2 - I_3) - \xi I_1\omega_1'^n]/I_1 \\
\Delta\omega_2'^n &= \Delta t[M_2^n + \omega_3'^n\omega_1'^n(I_3 - I_1) - \xi I_2\omega_2'^n]/I_2 \\
\Delta\omega_3'^n &= \Delta t[M_3^n + \omega_1'^n\omega_2'^n(I_1 - I_2) - \xi I_3\omega_3'^n]/I_3
\end{aligned} \tag{2.22}$$

(3) Predict angular velocities at the current time step.

$$\omega_i^n = \omega_i^{n-\frac{1}{2}} + \frac{1}{2}\Delta\omega_i'^n \tag{2.23}$$

(4) Calculate angular velocity correctors.

$$\begin{aligned}
\Delta\omega_1^n &= \Delta t[M_1^n + \omega_2^n\omega_3^n(I_2 - I_3) - \xi I_1\omega_1^n]/I_1 \\
\Delta\omega_2^n &= \Delta t[M_2^n + \omega_3^n\omega_1^n(I_3 - I_1) - \xi I_2\omega_2^n]/I_2 \\
\Delta\omega_3^n &= \Delta t[M_3^n + \omega_1^n\omega_2^n(I_1 - I_2) - \xi I_3\omega_3^n]/I_3
\end{aligned} \tag{2.24}$$

(5) Update angular velocities by using the correctors.

$$\omega_i^{n+\frac{1}{2}} = \omega_i^{n-\frac{1}{2}} + \frac{1}{2}\Delta\omega_i^n \tag{2.25}$$

For small time steps used to resolve the grain contacts and for quasi-static conditions, in which the angular velocities are small, the number of iterations is typically small. Usually, between three and five iterations are required to achieve machine precision tolerance. Orientations for each grain are updated using Evans' singularity with free quaternion approach (Evans and Murad, 1977). For Euler's equations of motion and the integration of the quaternions, the torques are specified in the body or principal frame, while the contact detection and force calculations are performed in a space or global frame. Therefore, the rotation matrix from space to body frame is given by:

$$R = \begin{pmatrix} -q_1^2+q_2^2-q_3^2+q_4^2 & -2(q_1q_2-q_3q_4) & 2(q_2q_3+q_1q_4) \\ -2(q_1q_2+q_3q_4) & q_1^2-q_2^2-q_3^2+q_4^2 & -2(q_1q_3-q_2q_4) \\ 2(q_2q_3-q_1q_4) & -2(q_1q_3+q_2q_4) & -q_1^2-q_2^2+q_3^2+q_4^2 \end{pmatrix} \tag{2.26}$$

Where the q's are the quaternions of Evans and Murad (1977).

$$\begin{aligned} q_1 &= \sin\frac{\theta}{2}\sin\frac{\psi-\phi}{2} \\ q_2 &= \sin\frac{\theta}{2}\cos\frac{\psi-\phi}{2} \\ q_3 &= \cos\frac{\theta}{2}\sin\frac{\psi+\phi}{2} \\ q_4 &= \cos\frac{\theta}{2}\cos\frac{\psi+\phi}{2} \end{aligned} \tag{2.27}$$

And θ,ψ,ϕ are Euler's angles representing successive rotations about the z, x' and z' axes. It turns out the time derivatives of the orientation parameters (q_1, q_2, q_3, q_4) can be expressed in terms of the quaternions and the angular velocities.

$$\begin{aligned} \dot{q}_1 &= \frac{1}{2}(-q_3\omega_x - q_4\omega_y + q_2\omega_z) \\ \dot{q}_2 &= \frac{1}{2}(q_4\omega_x - q_3\omega_y - q_1\omega_z) \\ \dot{q}_3 &= \frac{1}{2}(q_1\omega_x + q_2\omega_y + q_4\omega_z) \\ \dot{q}_4 &= \frac{1}{2}(-q_2\omega_x + q_1\omega_y - q_3\omega_z) \end{aligned} \tag{2.28}$$

$$\sum_{i=1}^{4} q_i^2 = 1 \tag{2.29}$$

Above equations can be solved explicitly for the quaternion values at the new time step in terms of the old values, and for the angular velocities at the midpoint of the time step using time-centered finite difference scheme.

2.3 HYBRID MPI+OPENMP DESIGN CONSIDERATIONS

Message Passing Interface (MPI) is the industry-standard Application Programming Interface (API) for message passing across distributed-memory machines and often incorporates with the idea of domain decomposition. Open Multi-Processing (OpenMP) is a standard API for parallel programming on shared-memory architectures that parallelizes serial code by adding directives to instruct the compiler how to distribute workload at the data level. A hybrid HMP+OpenMP model is considered in this section to fully leverage the potential of mutliprocessing clusters. This combination of models makes use of data distribution and explicit message passing between cluster nodes, as well as shared memory and multithreading within nodes.

2.3.1 Data Abstraction and Motivation for the Use of Data Structure

Three levels of abstraction are used in the binning algorithm: blocks, bins, and grains. To begin, the computation domain is partitioned into blocks, each of which is ”owned” by a computing processor. The block is then separated into equal-sized bins with a length equal to or greater than the diameter of the largest grain in the assembly. The number of processors is chosen to minimize the total area of communication between the computational sub-domains. The bin length can be expressed mathematically as $R + \Delta R$, where R is the largest diameter of the grain in assembly and ΔR, is determined empirically to strike a balance between the size of bins and the maximum number of grains that can reside in a bin, as increasing the bin size has the same effect as decreasing the total number of bins required to partition the domain. As a primitive task unit, each bin contains grains and communicates with 26 neighbors to detect and resolve contact and force. Finally, grains are assigned to a given bin based on the coordinates of their mass center in relation to the bin and block sizes.

In the realm of clustered system, it is prudent to define a fixed number of grains in a bin, as conveying all data at once is far more efficient than sending a fraction several times due to data arrival latency. As a result, the bin size should be fixed to minimize border/ghost layer communication overheads. The simplest method is to calculate the size of the bin in advance and estimate the maximum number of grains that can fit within. This technique grows linearly for large numbers of grains of roughly the same size and simple shape, but it is less efficient for assembling arbitrary shaped avatars with a range of grain sizes. As a result, a binning algorithm that explicitly models all bins and allocates each grain to its proper bin is memory intensive and prone to memory management problems. Furthermore, from a practical standpoint, the number of bins may exceed the number of grains if the majority of bins are empty, resulting in resource waste. A more elegant solution is to use a linked-list abstraction to map grain-bin associations, so that grains are sequentially indexed by a unique number and only the exact number of total grains is maintained. This technique efficiently implements the belonging relationship between bins and grains, the neighbor list for each grain, and the ghost bins using four lists in $O(1)$ operations.

2.3.2 Border/Ghost Layers Communications

In a binning algorithm, each grain is uniquely labelled using a hash on their coordinates and based on the bin-size. A carefully chosen cutoff distance guarantees that every grain can only interact with grains in 26 neighbor bins, as even the largest grain cannot extent across more than two bins. However, a boundary grain may extend into neighbor sub-domains, and this requires each sub-domain to maintain a copy of remote grains to correctly account for the interactions of boundary grains. This creates a halo region which is an extended layer of bins outside the boundary bins and records all updates from neighboring sub-domains. The process of exchanging border information is termed border/halo communication. The purpose of border/halo communication is to update boundary information for sub-domains because each processor only understands its own space and the associated grains. Therefore, before proceeding to a new time step, boundary grains in the border/halo area must be updated with the latest translational velocity, angular velocity, rotation in the global frame, and center mass location. The new code packs information beforehand and minimizes communication overheads effectively. The amount of data required to update remote data from halo regions is constant and small; hence it can be packed to send collectively. Note that grains' shear history in border/halo layers does not need to be transmitted because those grains are computed on their host processors.

2.3.3 Across-Block Migration

Across-block migration refers to a circumstance when a grain enters or leaves its host block. If a grain migrates across the block border, one sub-domain needs to delete it while another needs to add it. In contrast to the border/ghost communication, where data can be packed and communicated together, the across-block migration shall be considered individually for different grains because the size of the message being transferred is different. A naive algorithm analogous to border/ghost migration is to construct a spatially outward extension from each processor's border (Yan and Regueiro, 2018b). The size of the extended layers is independent of the size of virtual bins, and it is determined by the velocities of the grains and the time step used in the current time increment. Each processor should check its extended layers to see if any of its grains move into the computational domain. If yes, the corresponding processor should send such grains to the interested processor and delete them from its own space. However, this is not the best solution for across-block migration because a grain can essentially move to any other processor, and the grain velocity is difficult to estimate for some dynamic problems.

In general, three-dimensional DEM simulations fall into two main categories: static and quasi-static, and dynamic. While the velocity range in problems falling into the first category can be reasonably estimated, it is more difficult to determine the position that a grain could reach in the second category problems. The idea of an extension layer also creates technical difficulties in its implementation. In the binning algorithm, blocks that are mapped on sub-domains might have different spatial locations and hence create many edge cases. Moreover, if a history-dependent tangential contact model is chosen, the shear history ought to be sent as well. It is less efficient to send and receive complex information of varying length from specific senders to specific receivers using point-to-point communication APIs (primitive send and receive functions) in MPI implementation. Moreover, this is very likely to cause communication deadlocks that can be addressed using BoostMPI, which runs on top of MPI implementation, or using one side communication features of MPI3. In the new code we adapt highly tuned collective functions, MPI_Alltoall() and MPI_Alltoallv(). This makes the code more readable because all processors call the collective functions and, consequently, the same code is applicable for all processors.

2.3.4 Dynamic Workload Distribution

The workload distribution is almost uniform among processors in simulating semi-static load problems, such as the triaxial compression test, even though grains may consist of various numbers of nodes on their reconstructed surface because the grain assembly is arranged in a relatively uniform and compact manners. Therefore, although the specific number of grains residing in a bin varies, the total amount of workload in a sub-task is similar, and the approximate cost can be evaluated in advance. Typically, static domain decomposition works well for homogeneous compact materials, but it is not as effective for inhomogeneous grain distributions due to performance-limiting work imbalances. In highly dynamic modeling of debris flows or earthquake simulations, characterized by large grain displacement between time steps, grains can move and enter arbitrary sub-domains frequently, causing the number of grains residing inside a fixed-size bin to fluctuate as well.

The parallel scheme adopted herein aims to be applicable for wide ranges of simulations and adopts an adaptive re-binning scheme for highly dynamic problems. This scheme decomposes the latest computational domain and redistributes resources to best balance the current geometrical configuration. The subroutine can be either called at each time step or after specific numbers of time steps depending on the problems. The re-binning step has the same time complexity as the standard step. It does not affect the performance because dynamic adjusting and re-assigning grains at the beginning of a time step essentially calls the same subroutine as across-block migration, it also skips border/halo communication. The fundamental idea behind adaptive re-binning is to treat the boundary sub-domains as infinitely large temporarily. Any grain about to leave the computational domain will still behold in the boundary sub-domains. After a certain amount of time steps, the whole computation grid is regenerated. An alternative of adaptive dynamic binning is the quad-tree algorithm (a recursive coordinate bisection algorithm), which divides the domain into flexible bin sizes, counts the number of grains within a bin, and emphasizes the load balance of each bin. While the dynamic decomposition approach can improve load balancing of MPI-implementing simulations, it is limited by its focus on balancing the number of grains rather than the actual workload among processors (Yan and Regueiro, 2019). This can shift computational effort to individual processors, which leads to additional undesirable work imbalance.

2.3.5 Modeling a Flexible Membrane in DEM

Both fixed and movable rigid boundaries are readily modeled in the parallel implementation because the geometries of such boundaries are simple and can be modeled as a single entity. However, a deformable boundary is required to initiate and develop strain localization, and such boundaries, such as a flexible membrane in a triaxial simulation, are challenging to parallelize. In a conventional triaxial test, a cylindrical specimen is enclosed by a flexible latex rubber membrane that can stretch or contract in response to local deformation. To replicate the latex rubber in DEM, packed spheres were arranged in a hexagonal pattern and deform with relation to their surrounding spheres, as illustrated in Figure 2.4. Contact bonds were allocated to the membrane spheres to keep them connected while allowing relative motions between spheres to be unrestricted. The contact bond strength between membrane spheres was set to a value that was sufficiently weak to allow for complete development of the shear band and volumetric change while remaining sufficiently strong to enclose the specimen. Unlike the temporary springs used in the soft-contact model of inter-grain interaction, membrane spheres are permanently connected via normal and tangential springs. The normal spring secures the spheres in place as the membrane stretches or contracts, while the tangential spring maintains the hexagonal pattern of the spheres. The stiffness of these springs is much smaller than the grain-membrane contact stiffness, allowing the membrane to deform freely mimicking the latex rubber.

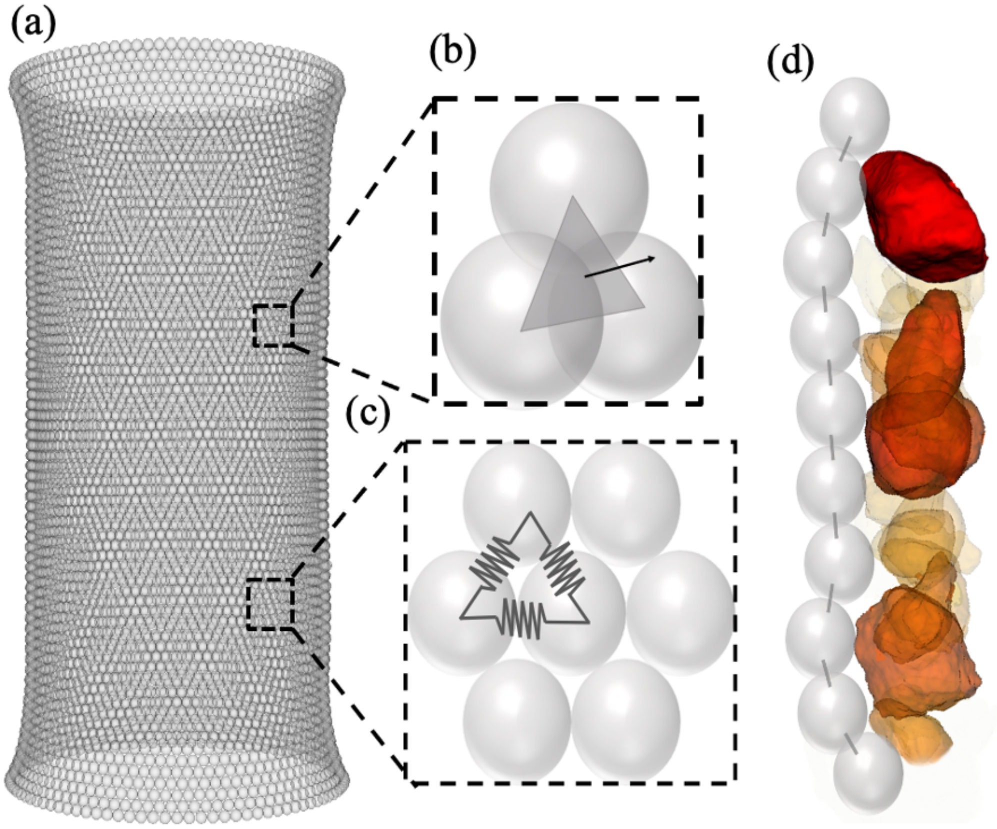

Figure 2.4: (a) Flexible membrane representation, $\sim 15,000$ balls. (b) Confining pressure normal to faces. (c) Hexagonal patterns of bonded spheres. (d) Membrane-grain interaction.

Numerically, an external force is applied to each sphere in accordance with the confining pressure specified. In essence, the magnitude of the applied force on the membrane spheres is equal to the product of the confining pressure and the triangular region's area. Moreover, the applied force is perpendicular to the region and pointed inward to the specimen. For a triangular mesh element T, we can calculate the outward normal $\mathbf{n}$ and the directional area $\mathbf{S}$. Thus, according to the divergence theorem of Gauss, the volume of the specimen can be calculated:

$$V = \iiint_{\Omega} dV = \oint_{\partial\Omega} \mathbf{S} \cdot \mathbf{n} d\Gamma \approx \sum_{T \in \partial\Omega} \mathbf{S}_T \cdot \mathbf{n}_T \tag{2.30}$$

Due to the simple geometry of the membrane elements, interactions between grains and the membrane are straightforward and as follows: an avatar node determines whether the distance between the node and the sphere center is less than the radius of the sphere and determines the amount of penetration as well as the normal direction. However, this procedure has two limitations. To begin, the grain of interest should iterate on all membrane spheres and perform a bounded circle check to identify nearby spheres before refining the force resolution phase on individual nodes. While sphere-grain interaction is considered to be notably less computationally intensive than inter-grain interaction, this process is still $O(n^3)$ and requires a significant amount of computing power. Second, the flexible membrane is designed to conform to the geometric configuration of the specimen and strike a balance between confining pressure and resistance forces, which is proportional to penetration depth and is dependent on contact stiffness. However, once the membrane sphere enters the grain completely, all distances between the nodes and the sphere center exceed the radius of the sphere, and the sphere is no longer considered to be interacting with grains. Consequently, the spheres will continue to move inward and rapidly distort the entire membrane. This issue can be resolved by introducing a set of sophisticated criteria that classify the position of the membrane sphere relative position of the grains. However, this is far from an efficient or elegant measure. As a solution, we implemented grain-membrane interaction similarly to inter-grain interaction by temporarily moving spheres in the principal frame of the grain of interest and reading penetration from its LS grid. This modification increases efficiency because only one lookup of the LS table is required to complete a pair of grain-membrane interactions. In addition, this process can be applied in a variety of different scenarios without modifying the code or establishing arbitrary rules because decoding LS values from a fixed underlying grid is reliable and accurate.

Directly partitioning and parallelizing a flexible membrane according to the hash of coordinate is challenging for the following reasons: 1) assignment of a membrane sphere to one sub-domain hashed on its spatial coordinate is complicated as boundary deforms locally, and it is tedious to track and update spheres that are about to migrate across sub-domains. In addition, the hexagonal connectivity of a sphere should also be maintained before and after the migration; 2) it is possible that a membrane sphere and its contacting soil grains belong to different sub-domains. As a result, to correctly capture this relationship, the implementation must memorize the coordinate of a membrane sphere and its spatial relationship with contacting grains; and 3) although the information exchange between membrane spheres and contacting grains involves border communication and cross-domain migration and has a great similarity to the information exchange between grains, this kind of message passing occurs much more frequently and retards the computation.

2.3.6 Memory Management

Our initial implementation of the parallel code duplicated one copy of all grains for each processor, which turns out to be an acceptable design when the problem size is modest. The critical factor in this design is to optimize simulation speed, as each processor can handle all updates to grain velocities and rotations without reloading the morphology data, and it is simple to implement. Because reading data from main memory is approximately hundred times slower than reading data from cache in modern systems, preserving data locality is critical for performance. To simulate large problems, the increased number of processors similarly increases the size of the overall data file, producing four distinct issues: 1) the combined effect puts a strain on the memory, leaving little space for calculation; 2) in a clustered computing system, multiple processors share a finite amount of memory, and we may be unable to allocate sufficient memory to each processor, resulting in under utilization of the computing resources available; 3) during the initialization phase, the code must read all data files many times, which turns out to be a significant expense, and 4) this approach defies the goal of parallelism because another common reason to parallelize serial code, aside from speeding up execution, is to partition the data between processors when it is too large to fit on a single processor. Clearly, retaining a copy of data regardless of its size violates this design concept. With these constraints in mind, the solution is to read morphological files only once and assign grains to one of the processors based on their coordinates. When a grain is about to migrate to another computational sub-domain, the new host processor consults the database and regenerates the avatars using the most recent velocity, rotation, and friction information. Rather than releasing the memory associated with a grain that migrates to another sub-domain, we mark it as unused and retain it in memory in case it migrates back later, thus reducing the number of times we look up the database. This procedure was not invoked frequently during the simulation of quasi-static type problems, resulting in a negligible amount of additional overhead.

2.3.7 Limitations of the Parallel Implementation

In our implementation we treat the whole computational domain as a big box and cut it into identical cuboid sub-domains. Therefore, the underlying connectivity pattern for message passing is determined. As a result, the implementation details such as linked list-like data structure, halo communication, and grain migration are heavily reliant on this specific yet sophisticated sub-domains configuration. Fortunately, this implementation is tuned to utilize an arbitrary number of available processors and to suggest the best domain granularity. The major drawback is that it cannot properly handle problems where the distribution of the grains across the domain is inhomogeneous, i.e. domain sparsity. For instance, when dealing with a cone-shaped container we must still create the computational domain as a box large enough to encompass the cone. As a result, some sub-domains are left empty, resulting in resource waste.

The second drawback of this implementation is that it cannot use as many processors as possible because the cutoff size of a bin must be larger than the maximum equivalent diameter of a grain assembly. In addition, the cutoff size is chosen so that the average number of grains assigned to one processor is larger than a prescribed value. Although this implementation can guarantee that a processor is saturated by at least a certain amount of workload, they are also reasons why some of the cores being used are halted at times. Furthermore, the domain decomposition strategy induces work imbalance among processors due to different amounts of additional interactions with ghost bins in border layer. The center sub-domain is totally encircled by others and is influenced by ghost bins from all six faces, whereas a corner sub-domain has just three ghost bins. Thus, a cubic sub-domain consists of n^3 bins, and the number of bins involved in force resolution is $(n+2)^3$ for a center sub-domain, but only $n^3 + 3n^2 + 3n + 1 = (n+1)^3$ for a corner sub-domain, with $n = \text{Domain Size}/(\text{Bin Size} \times \text{Processors})$. The work ratio $(n+2)^3/(n+1)^3$ decreases as the number of processors increases and increases as the number of bins per sub-domain increases. Finally, the implementation is not universally applicable and ought to be modified for different contact models, boundary conditions, and sorting algorithms, although the code can handle many types of problems. As previously stated, parallel implementation treats various boundary conditions quite differently, and the implementation primarily relies on the current linked-list-like data structure to index and sort grain and to map the relationship between grains, bins, and blocks. Hence, both the halo layer communication and the grain movement are intrinsically linked to the specific sorting algorithm.

2.3.8 Programming Environment

The principles of Object-Oriented Programming (OOP) were followed in the design and programming of the code. Specifically, various classes are designed to model the practical concepts and objects that exist in a DEM simulation system: grains, bins, and exchange information packages. The Standard Template Library (STL) is heavily used, such as vector and set, to ensure code robustness and performance. The Eigen library is used to facilitate matrix-matrix or matrix-vector multiplication and transfer mathematical operations naturally. In developing MPI functions of three-dimensional LS-DEM, both border/ghost communication and across-block migration are accomplished using highly tuned API functions.

2.4 IMPLEMENTATION DETAILS AND EXPERIENCE

2.4.1 Parallel Implementation of the Membrane

Parallel implementation of a flexible membrane is more complex than the parallel implementation of a rigid boundary that does not deform locally. In this work we explored two approaches to membrane simulation in a parallel environment based on the concept of addressing local grain-membrane interaction in distinct computational subdomains specific to the corresponding host processors. We note that, the concept of directly decomposing flexible membranes between different processors was quickly abandoned due to the inefficiency of tracking grain-membrane interactions, updating spheres, and maintaining hexagonal connectivity across processors. For example, a single large grain may interact concurrently with multiple membrane spheres, but each sphere may belong to a different processor; additionally, the hexagonal pattern should be preserved whenever a sphere migrates to an adjacent processor.

Our first parallel implementation (Figure 2.5) utilized an additional processor to exclusively handle grain-membrane interactions. Before the contact detection and force resolution phases, all normal processors communicate most recent positions of the grains to the membrane processor, so that the grains in the membrane processor are up to date. In membrane processors, grain-membrane interactions are resolved concurrently with inter-grain interactions, and grain forces and moments are communicated back to normal processors. The most advantageous aspect of this strategy is that the time spent on computing grain-membrane interaction is separated from the inter-grain interaction, thereby reducing the serial portion of code. However, this implementation is complicated and doubles memory consumption due to the additional copies of grains required by membrane processors. Alternatively, the second strategy (Figure 2.6) does not employ an additional membrane processor but rather operates on grain-membrane interaction immediately

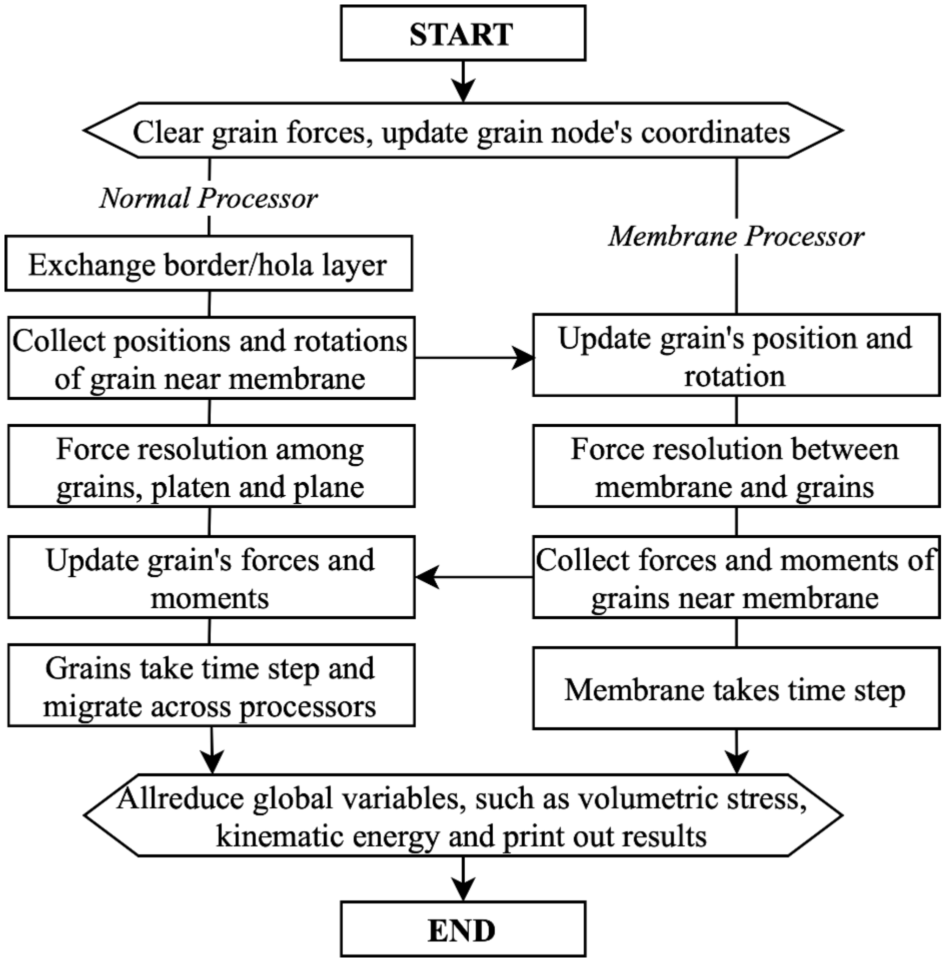
START
Clear grain forces, update grain node's coordinates
Normal Processor
Membrane Processor
Exchange border/hola layer
Collect positions and rotations of grain near membrane
Update grain's position and rotation
Force resolution among grains, platen and plane
Force resolution between membrane and grains
Update grain's forces and moments
Collect forces and moments of grains near membrane
Grains take time step and migrate across processors
Membrane takes time step
Allreduce global variables, such as volumetric stress, kinematic energy and print out results
END

Figure 2.5: Flowchart of parallel strategy for using an extra membrane processor to handle the membrane-grain interactions.

following force resolution; each processor is equipped with its own membrane, and only those spheres interacting with grains are activated. At each time step, processors broadcast and gather the forces of activated spheres from different sub-domains prior to advancing a time step. This strategy appears to be more elegant and simpler to implement; it also results in a more qualified parallel code because all processors execute the same task. However, interaction between grains and the membrane becomes a necessary component of all processors. We prioritized speed when choosing between the two strategies and discovered that the performance of the two versions is dependent on the number of grains being modeled; when the problem size is small, the membrane processor can complete grain-membrane interactions faster than normal processors and vice versa, when the problem size becomes larger.

2.4.2 Memory Consumption Minimization and Memory Leak Prevention

The C++ class to represent a grain used in LS-DEM consists of a large LS table containing the data for the nodes seeded on the reconstructed grain surface along with their shear histories. Most of this data remains unchanged and only small amount of data has to be communicated among

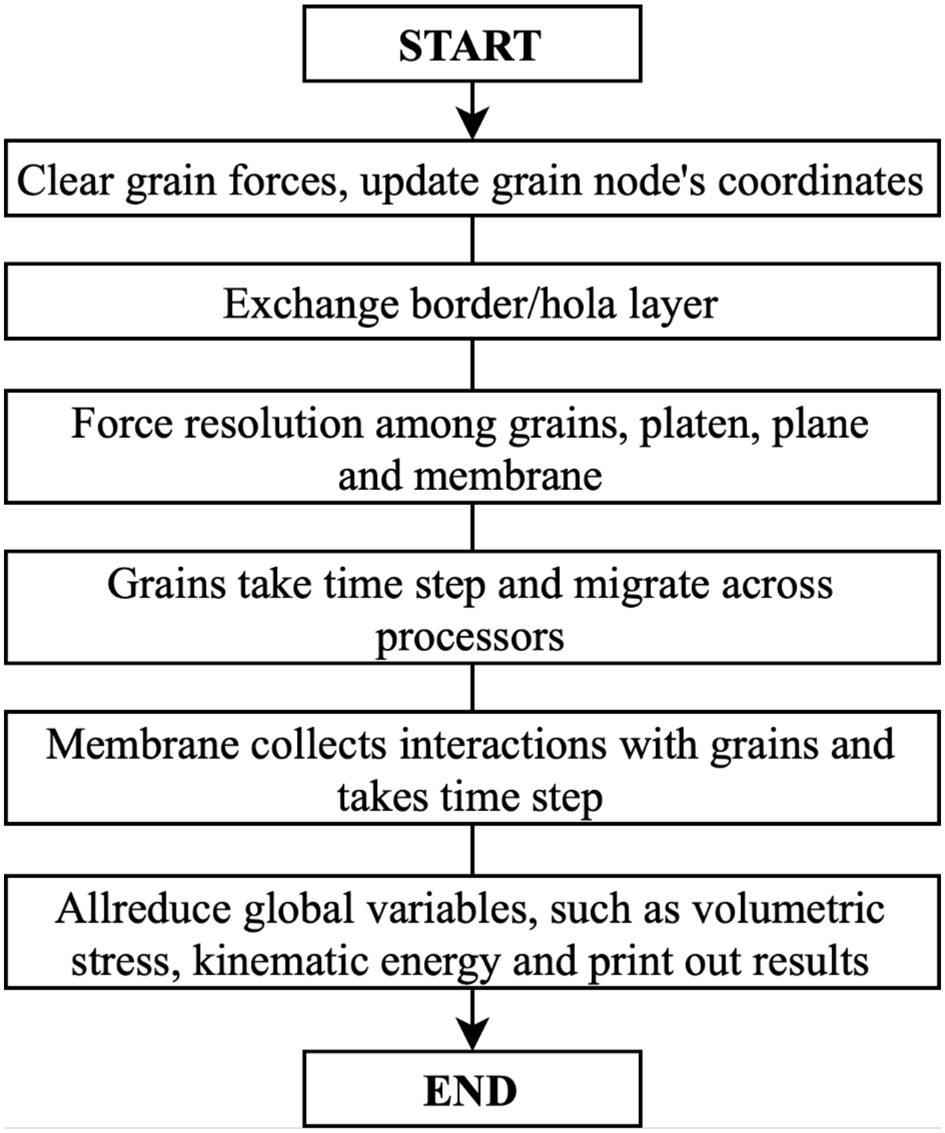
START
Clear grain forces, update grain node's coordinates
Exchange border/hola layer
Force resolution among grains, platen, plane and membrane
Grains take time step and migrate across processors
Membrane collects interactions with grains and takes time step
Allreduce global variables, such as volumetric stress, kinematic energy and print out results
END

Figure 2.6: Flowchart of parallel strategy to assure that the membrane-grain interactions are handled the same way by all processors.

sub-domains. As a result, it is wiser not to store a whole grain object in the bins, and only index it instead.

One of the crucial aspects is that the number of grains can be smaller than the number of bins, which is somewhat counter-intuitive since bins are conceptually more extensive abstraction than grains. For a mini grain assembly in our test, 74 grains were distributed across a $100 \times 100 \times 100$ computational domain. The largest diameter of the assembly is 20, hence the cutoff distance is chosen as 20; this creates 125 bins partitions. Therefore, instead of building 125 bins and assigning grains hashed on their mass centers, it is more efficient to maintain an array of 74 grains and keep track of their bin indices. Although contact detection, resolution algorithm, and time step work directly on the grain array, the bin structure is still required to assist message packing in the MPI communication.

The rigid body assumption significantly reduces the number of messages to exchange. Under this assumption, the information required to update grain status for the next time step only consists of the positions of mass center, rotations with respect to the global frame, plus the translational and angular velocities. Those are quantities needed to update boundary grains in the outermost bins of a sub-domain via border/halo communication. Since both LS table and the node positions relative to the center are unchanged during simulation, we have another implementation option to update grains: regenerating the grains from their morphology files with updated quantities. That is: after border/ghost communication, a new grain can be generated from the corresponding morphology file with freshly computed location, quaternion, and velocities. This alternative requires less memory for a very large problem but needs to access disk memory more frequently. Compared to the border communication, across-block migration is much more complicated due to the use of the history-dependent tangential contact model, where the shear history has to be brought together with the migrating grains because moving to other processors does not imply grains lost contact with their neighbors.

To avoid memory leakage, containers in Standard Template Library (STL), such as vector and set, are heavily relied on. The smart pointer feature is used as much as possible whenever it applies. While raw pointers cannot be completely avoided due to MPI's incompatibility with smart pointers, they are produced and discarded with special care. Every processor keeps a copy of the full grain list to map bins and grains, which is a memory-efficient strategy. In addition, the copy constructor and assignment operator are overwritten to avoid memory issues. Eigen and STL also inherently implement deep copy and provide memory protection.

2.4.3 Code Simplification and Edge Case Avoidance

One of the essential rules of MPI implementation is that as many processors as feasible accomplish the same task. To simplify implementation and avoid edge cases, each block/sub-domain is wrapped in a layer of padding bins (the number of bins along each dimension is increased by two, so that all bins within the sub-domain are regarded spatially similar, that is, each has a full set of 26 nearby bins. At the domain decomposition level, our code treats each processor identically by adding additional computational domains in such a way that each processor is associated with sub-domains of equal size. Furthermore, the number of processors used is determined to ensure that the communication area between MPI ranks is kept to a minimum. Then the code may dynamically select the optimal domain granularity based on both domain topology and resource availability. Moreover, the code establishes a limit value to ensure that each processor works on a minimum number of grains to conceal communication overhead; this value is established based on the prototype test results and is dependent on the average number of nodes used to discretize avatars. The code also takes advantage of MPI built-in functions like MPI_Dims_create(), MPI_Cart_create() and MPI_Cart_shift() to arrange processors into a grid, which allows MPI to automatically handle edge cases when communicating with neighbors, such as when each processor is instructed to send data to their left neighbor, whereas there is no left neighbor for the left-most processor.

2.4.4 Computational Effort Reduction

To further minimize computation time, we take advantage of Newton's third law and successfully reduce the computation time by a factor of two. In an ideal scenario, where the grain surface is continuous and integrable at any location, the interaction force between master and slave grains is the opposite of the interaction force between slave and master grains while maintaining the same magnitude. However, due to the discrete representation employed in the LS-DEM, the magnitudes of forces and reaction forces are not same. As a result, the code always applies the force exerted by the grain with the lower index to the grain with the higher value. Even though this measure can only guarantee that the resulting forces are identical within a sub-domain, it is inadequate at the boundaries where forces are always exerted from the halo layer to the inner grains. However, it reduces randomness due to thread-level parallelism and domain decomposition, which keeps the differences between tests within an acceptable range. Newton's third law requires that only half of the adjoining bins be iterated. As illustrated in Figure 2.2, the code takes into account 13 neighboring bins, including all nine bins above the target bins and four bins at the same level as the target bins. Any 13 bins can be chosen, and the algorithm chooses these 13 bins since the additional interactions between grains and boundary grains are reasonably straightforward. The code avoids handling edge cases as a result of padding. In a shared-memory system, the grain-grain interaction is entirely symmetrical because the computational domain and message exchange are not partitioned. This is not true for parallelism in distributed memory machines, because only 'real' grains are taken into account and updated when a real grain interacts with grains in the halo region. For instance, a bottom grain in the upper sub-domain should interact with ghost grains in the lower sub-domain, but only forces exerted by ghost grains on non-ghost grains are taken into account.

2.4.5 Position Reasoning and Linked-List Data Structure

A naive approach to binning iterates over bins to detect contacts and resolve forces. A more elegant and efficient technique used in our algorithm uses linked lists to map relationships between grains and bins and runs in time complexity $O(1)$. This approach enables a simple encoding and decoding of a bin ID associated with a certain grain with $O(1)$ complexity, and vice versa. The following example introduces four connected lists and is illustrated in Figure 2.7.

(1) The first array:

```
int* bins = new int[num_of_bins];
```

has a length of the number of bins and is initialized as all -1. The value that is stored in each element of bins is the first grain number in that bin. For instance, bins[23]=13 means the first grain in the 23-rd bins is #13, bins[4]=-1 means there is currently no grain stored in the 4-th bins.

(2) The second array

```
int* binList = new int[num_of_grains];
```

has a length of the number of grains and is initialized as all -1. The value stored in each element of binList is the bin ID that the grain belongs to. For instance, binList[13]=8 means grain $\#13$ is stored in the 8-th bins.

(3) The third array

```
int* grainList = new int[num_of_grains];
```

has a length of the number of grains and is initialized as all -1. The value stored in each entry of grainList is the next grain ID stored in the same bin as the current grain. For instance, grainList[13]=17 means the next grain which was stored in the same bins as grain $\#13$ has index $\#17$; similarly, grainList[17]=-1 means there are no more grains stored after grain $\#17$. As a more comprehensive example, instructions bins[23]=13; grainList[13]=17; grainList[17]=9; and grainList[9]=-1 describe a search path if one is interested which grains are stored in bins $\#23$, the first grain is $\#13$ which followed by $\#17$ and $\#9$. There is no sequential order among grains, the one is added merely to assist neighbor searching and interaction computation.

(4) The fourth array

```
int* belongToRank = new int[num_of_particles];
```

has the length of the number of grains and is initialized as all 0. The value that is stored in each element of belongToRank is in the enumerator $\{0, 1, 2\}$, 0 means the grain is not associated with the current rank, 1 means the grain is residing inside the associated sub-domain, 2 means the grain is a ghost grain related to current rank. For instance, belongToRank[2]=0 implies the 2-nd grain does not belong to the current rank.

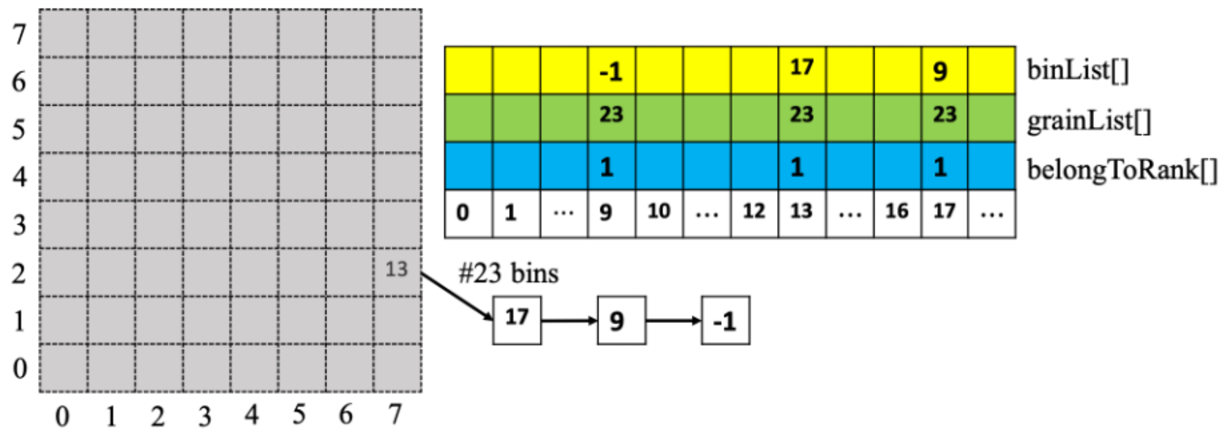

Figure 2.7: Grain search using linked list data structure

2.5 COMPARISON BETWEEN DIFFERENT IMPLEMENTATIONS

A numerical experiment was performed to examine the efficiency of the different code versions using the GNU C compiler on Lenovo NeXtScale nx360m5 nodes of the Savio system at UC Berkeley, which has 2 Intel Xeon Haswell processors with 12 cores per processor @2.3GHz. Each core has two 512-bit-wide vector processing units, four hardware threads, 256KB L1 cache, and every four cores share an 8MB L2 cache. The goal of this experiment was to measure both the strong and weak scaling of MPI implementations and to observe the performance improvement by integrating $O(n)$ algorithm and using parallel techniques.

2.5.1 Original Implementation of LS-DEM

The code developed by Kawamoto et al. (2016) uses a hybrid MPI+OpenMP implementation, which does not employ binning algorithm or tree algorithm to reduce the computation complexity. It loops over all the grains and relies on OpenMP directives. The load balance is improved by specifying a dynamic loop schedule and each processor receives grain updates through the collective function MPI_Allreduce(), which gathers information from all computing units and distributes it back. Although collective communication and identifier MPI_IN_PLACE could reduce some memory demands and the border/halo communication phase could also be removed because all processors have a copy of grains, there is still a significant amount of additional computational effort because this algorithm requires $O(n^2)$ computational complexity, and all grains are participating in the global all-to-all communications.

Overall, this LS-DEM code implementation is fast enough for most applications, as it can model a triaxial compression test on roughly $60,000$ grains within one day. The main limitation is that the code requires each processor to own one copy of all of the data and hence occupies a substantial amount of memory. Even though LS-DEM's contact algorithm has constant time complexity with respect to grid resolution, a large memory footprint nonetheless may lead to increased computation time due to cache misses and it limits the number of grains that can be simulated. For example, a $40 \times 40 \times 40$ reconstructed avatar requires 64,000 LS values to be stored. Moreover, the code formulation does not protect against false sharing or data race, i.e. two grains may simultaneously interact with the same grain and compete to write on the memory of the third grain.

2.5.2 Two Binning Algorithm Implementations

The flowcharts of the two parallel implementation introduced here are depicted in Figure 2.8 and Figure 2.9. They show four major flow components for both naïve binning algorithm implementation and the binning algorithm implemented in this work. The two implementations differ from each other in terms of the memory efficiency, the amount of interaction computation, the border/halo communication, and the across-block migrations. These four steps are not equally weighted and can take place at different times during the simulation. The computational complexity is effectively reduced from $O(n^2)$ to $O(n)$ for both implementations. Bins from each sub-domain are isolated from other processors and updated with information available in the host sub-domain. Therefore, they do not participate in the message passing phases. The false sharing or data race issue is also avoided in both implementations because every grain is only associated with and managed by one processor at a time.

2.5.3 Border/Ghost Communication Algorithm

At the beginning of a time step, the border/halo layers communicate with neighbors. Usually, a sub-domain needs to exchange information with its neighbors through six surfaces, twelve edges, and eight vertexes. In the new code, this step is simplified and reduced to three sequential steps. Taking advantage of the blocked and synchronized MPI_Sendrecv() function, the message passing procedure is guaranteed to happen in the right order. MPI_Sendrecv() is well-suited to send and receive a message simultaneously because it is specifically designed to circumvent deadlocks. The border/ghost communications algorithm is depicted with a two-dimensional illustration in Figure 2.10 and Figure 2.11, and the 3D algorithm is analogous. Two steps are sufficient for a two-dimensional border/ghost communication.

Consider a two-dimensional computational domain divided into six sub-domains. Each sub-domain is further divided into a 3×3 bins matrix (grey areas in the plot). Every sub-domain also maintains a copy of boundary bins of remote processors so that a layer of ghost bins is used to wrap the original 3×3 bins matrix and form a 5×5 matrix. The adopted algorithm shows that two sequential calls of MPI_Sendrecv() are enough to update all ghost bins, and only three sequential calls are required for a three-dimensional simulation.

As the first border/halo communication step (Figure 2.10), each processor sends its rightmost non-boundary bins (marked as type-1 bins) to update the left boundary bins of its right neighbor. Simultaneously, the current processor accepts a message from its left neighbor and updates the left boundary bins. MPI_Sendrecv() synchronizes this processor and completes this procedure at the same time. Similarly, each processor sends its leftmost non-boundary bins (marked as type-2 bins) to update its left neighbor's right boundary bins. The current processor then accepts a message from its right neighbor and updates the right boundary bins. The edge cases where the left or right neighbors do not exist are automatically handled because the new code organizes processors and constructs a Cartesian grid using MPI_Dims_create(), MPI_Cart_create(), and MPI_Cart_shift().

In the second border/ghost communication step (Figure 2.11), each processor sends its upmost non-boundary bins (marked as type-3 bins) to update top neighbor's bottom boundary bins. Simultaneously, the current processor accepts a message from its bottom neighbor and updates the bottom boundary bins. After these two sequential steps, the corner bins in the ghost area are all updated. One could consider the procedure to update corner bins a two-step process: first, send corner bins to the left and right neighbors, and the left and right neighbors, after receiving the corner bins, re-direct the corner bins to top and bottom. More complicated three-dimensional border/ghost communications are implemented using precisely the same logic and three steps. The left and right ghost bins are updated first, then the back and front ghost bins, finally the top and bottom ghost bins.

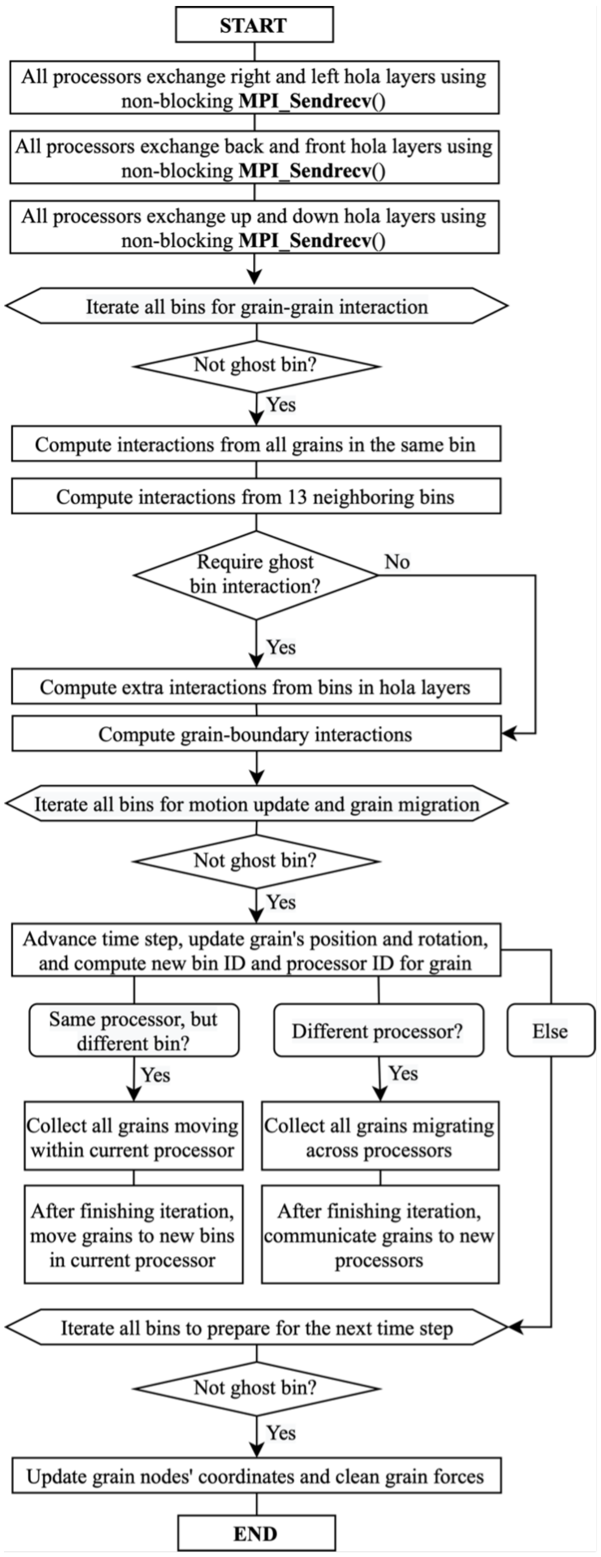
START
All processors exchange right and left hola layers using non-blocking MPI_Sendrecv()
All processors exchange back and front hola layers using non-blocking MPI_Sendrecv()
All processors exchange up and down hola layers using non-blocking MPI_Sendrecv()
Iterate all bins for grain-grain interaction
Not ghost bin?
Yes
Compute interactions from all grains in the same bin
Compute interactions from 13 neighboring bins
Require ghost bin interaction?
No
Yes
Compute extra interactions from bins in hola layers
Compute grain-boundary interactions
Iterate all bins for motion update and grain migration
Not ghost bin?
Yes
Advance time step, update grain's position and rotation, and compute new bin ID and processor ID for grain
Same processor, but different bin?
Different processor?
Else
Yes
Yes
Collect all grains moving within current processor
Collect all grains migrating across processors
After finishing iteration, move grains to new bins in current processor
After finishing iteration, communicate grains to new processors
Iterate all bins to prepare for the next time step
Not ghost bin?
Yes
Update grain nodes' coordinates and clean grain forces
END

Figure 2.8: Flowchart of the naive binning algorithm where bins are basic elements in iteration.

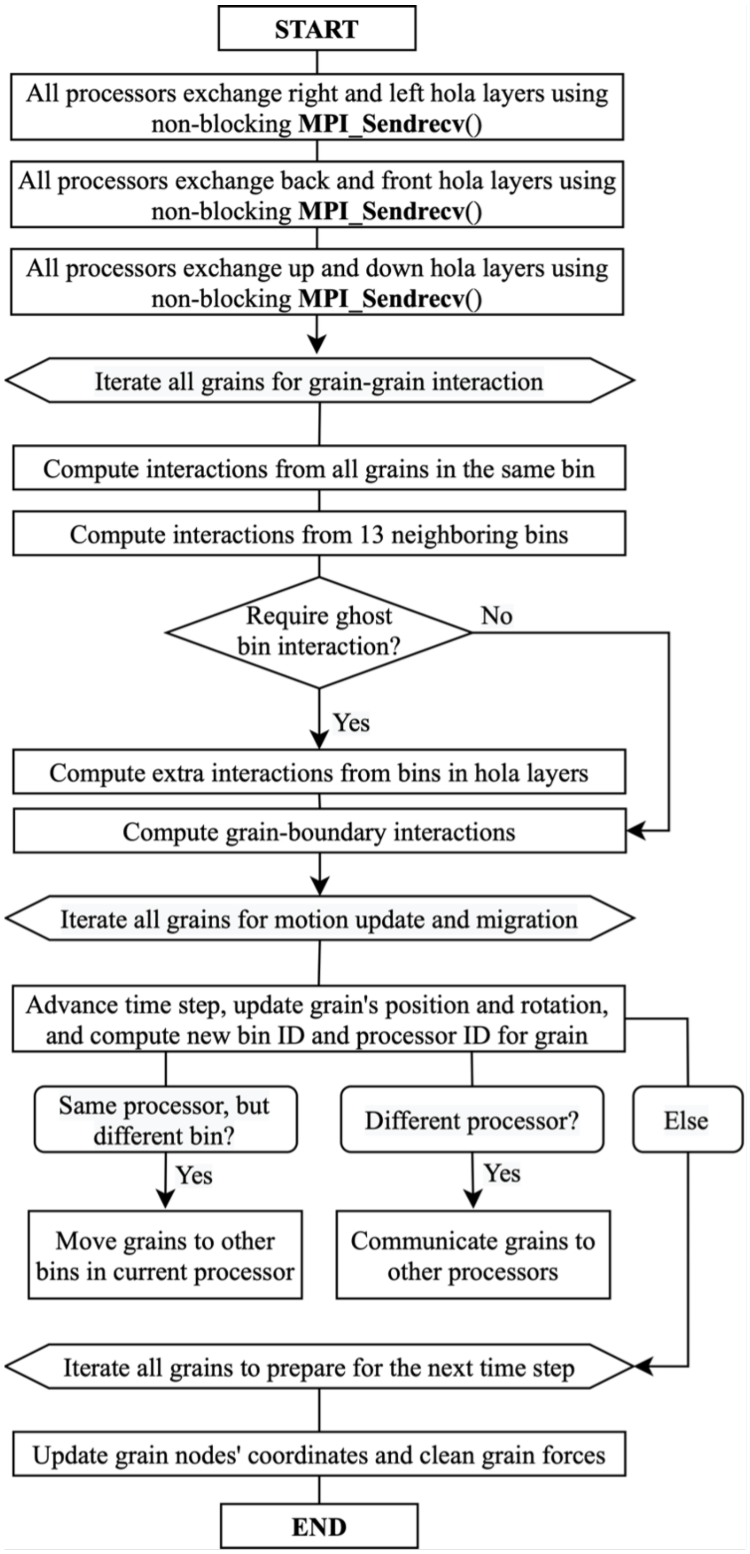
START
All processors exchange right and left hola layers using non-blocking MPI_Sendrecv()
All processors exchange back and front hola layers using non-blocking MPI_Sendrecv()
All processors exchange up and down hola layers using non-blocking MPI_Sendrecv()
Iterate all grains for grain-grain interaction
Compute interactions from all grains in the same bin
Compute interactions from 13 neighboring bins
Require ghost bin interaction?
No
Yes
Compute extra interactions from bins in hola layers
Compute grain-boundary interactions
Iterate all grains for motion update and migration
Advance time step, update grain's position and rotation, and compute new bin ID and processor ID for grain
Same processor, but different bin?
Different processor?
Else
Yes
Yes
Move grains to other bins in current processor
Communicate grains to other processors
Iterate all grains to prepare for the next time step
Update grain nodes' coordinates and clean grain forces
END

Figure 2.9: Flowchart of the binning algorithm where the grains are basic elements in iteration.

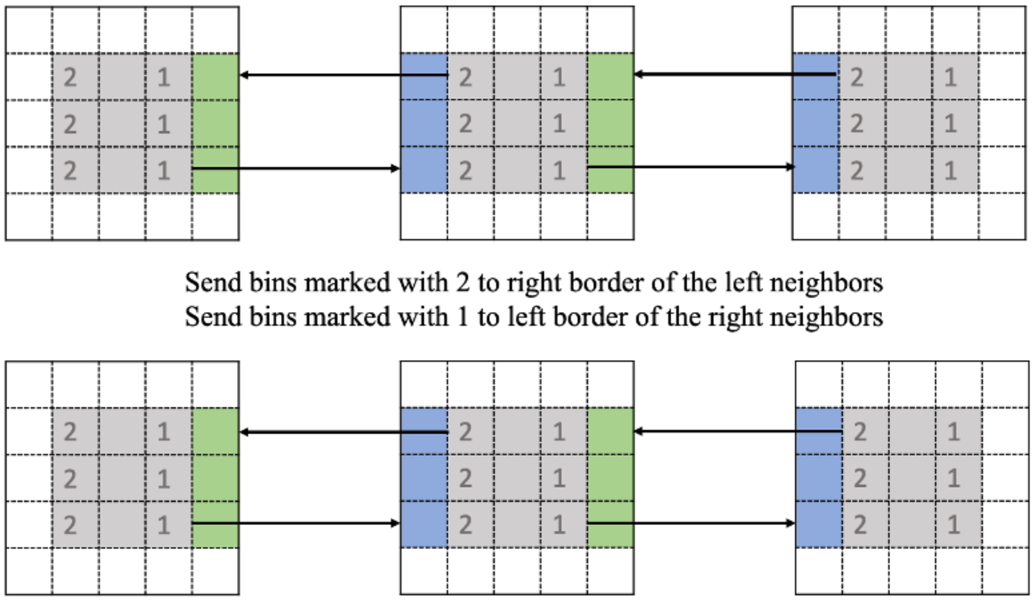
Send bins marked with 2 to right border of the left neighbors
Send bins marked with 1 to left border of the right neighbors

Figure 2.10: The first border/ghost communication step; exchange left and right layers.

2.5.4 Iterate Over Bins vs. Iterate Over Grains

This computation stage involves computing the interactions between grain pairs. The straightforward implementation iterates over bins and computes inter-grain interactions for grains that reside there, starting with interactions between grains in the same bin and progressing to interactions between grains in nearby bins. This is a simple yet inefficient technique, as bins may be empty. Therefore, we introduce a new force resolution strategy which iterates over grains and does optimum amount of work. The grain-to-grain relationship and the grain-bin relationship are mapped using linked-lists with time complexity $O(1)$. Carrying an envelope calculation, a strategy that iterates grains instead of bins could reduce the amount of analysis by an order of magnitude. The resulting code scales linearly and computations are directly proportional to the number of grains. In contrast, whenever a grain is queried, the naive implementation code requires to iterate over all bins to find the target. Therefore, the naive implementation is computationally more expensive and brings about redundancy. Also, consider a situation in which one grain is about to migrate, it has not already left the host sub-domain, but it obtains a new bin ID that differs from the current one. However, this operation cannot be immediately performed until all bins are checked because it is possible that the grain could move to unchecked bins and be recalculated. In the end, both implementations using data structure abstraction should consider extra interactions from ghost grains since a sub-domain does not understand the ambient information of the clustered system.

2.5.5 Across-Block Migration

The motion of a grain is updated at the third step, and across-block migration may occur at the fourth step. Both naive binning algorithm implementation and the new implementation using data structure abstraction adopt the same algorithm. This is a relatively expensive step because

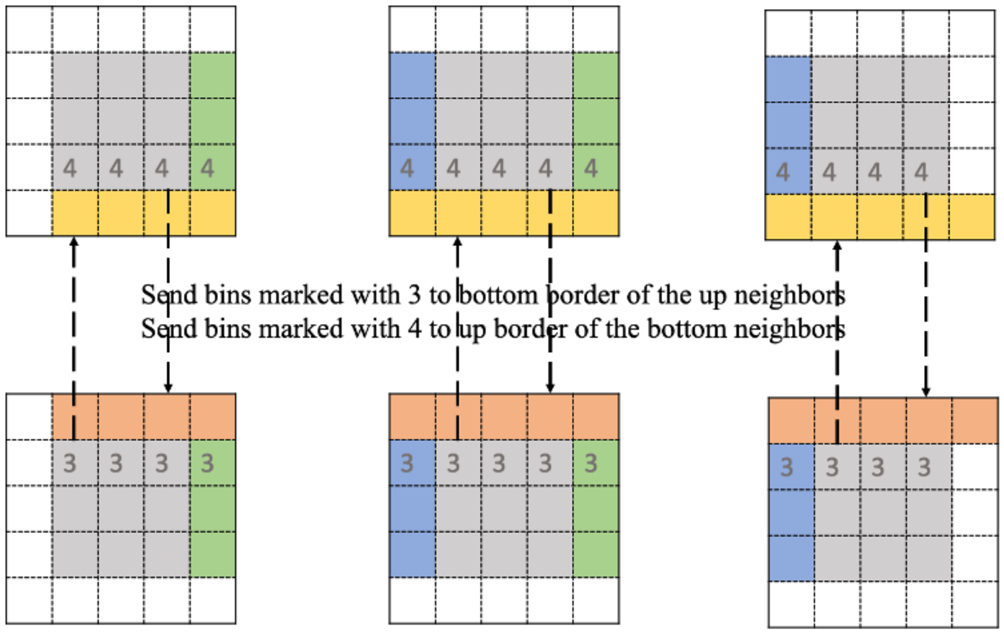
4 4 4 4
4 4 4 4 4
4 4 4 4
Send bins marked with 3 to bottom border of the up neighbors
Send bins marked with 4 to up border of the bottom neighbors
3 3 3 3
3 3 3 3 3
3 3 3 3

Figure 2.11: Border/ghost communication, step two; exchange up and bottom layers.

the contact history is also brought along with the migrated grain if the history-dependent tangential contact model were applied. Although each migrated grain is considered individually, necessary information such as its translational velocity, angular velocity, mass center location, and grain rotation can still be packed collectively to exploit overlap and reduce communication latency. To scatter migrated grains to other processors, collective operations like MPI_Alltoall() and MPI_Alltoallv() are heavily relied upon. It is also possible to only implement fundamental MPI_Send(), and MPI_Receive() functions with neighboring 26 processors if the time step is small enough so that grains will not move across an entire processor. However, this approach cannot be generalized, and using fundamental MPI_Send() and MPI_Receive() in nested loops could easily cause deadlocks. The across-block migration approach is illustrated below. We pack and communicate the common quantities first, and send the grain specific information (shear forces, normal shear direction, and contact grain ID) individually. This procedure is illustrated in Figure 2.12 and Figure 2.13.

In this process a packer is the container storing the necessary grain information and the underlying data structure of the packer is:

```
vector<vector<basic_grain_property>> packer(num_of_proc);
```

The basic grain properties have six components: grain ID, number of nodes on the grain, position, quaternion, translational velocity, and angular velocity. When a grain leaves the current sub-domain and enters another, the associated grain information is pushed back into the related packer[new_proc_id]. The contact history resides in three containers, which are defined as :

```
vector<vector<Eigen::Vector3d>> packerNodeShears(num_of_proc);
```

This tracks the vector of tangential forces on each node, while the length of the element is changing as the number of nodes seeded on a grain is not constant. The second quantity of a contact history

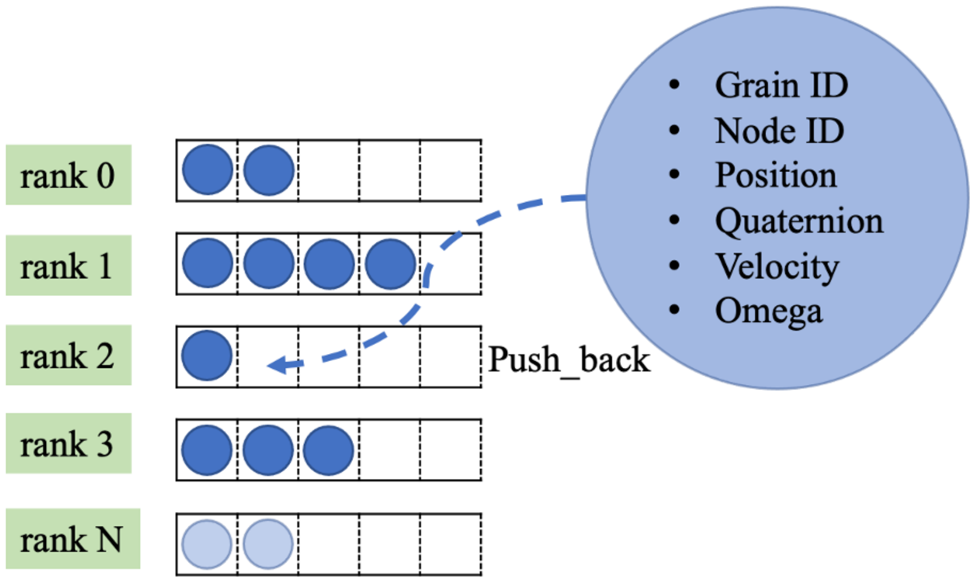

Figure 2.12: Schematic of data containers for basic grain information in across-block migration.

is the grain ID that a node is in contact with and is stored separately:

```
vector<vector<int>> packerNodeContact(num_of_proc);
```

The third quantity is the unit normal direction of a node in the principal body frame of the last time step. This is stored as:

```
vector<vector<Eigen::Vector3d>> packerNodeNormals(num_of_proc);
```

The data communication is achieved using the MPI collective functions MPI_Alltoall() and MPI_Alltoallv(). MPI_Alltoall() is a collective operation for the case where each processor sends distinct data to all other processors. The j-th block sent from processor i is received and placed in the i-th block of processor j's buffers. MPI_Alltoallv() adds flexibility to MPI_Alltoall() in that the location of data to be sent is specified by send_buff_displacement and placed in the location specified by recv_buff_displacement. The effect of all-to-all operation, MPI_Alltoall(), is equivalent to a matrix transpose operation as illustrated below Figure 2.14.

2.6 NUMERICAL EXPERIMENTS

2.6.1 Numerical Experiments on a Small Dataset

A series of small numerical experiments was performed to study communication overheads occurring in small size problems. The problem sizes were multiples of 74 obtained by duplicating existing avatars with positions shifted to avoid overlap. The computational domain in each numerical test was cubic, for instance, 592 grains are constructed using 8 copies of existing 74 grains and shaped into a $200 \times 200 \times 200$ domain. The problem setting was simple: domain boundaries

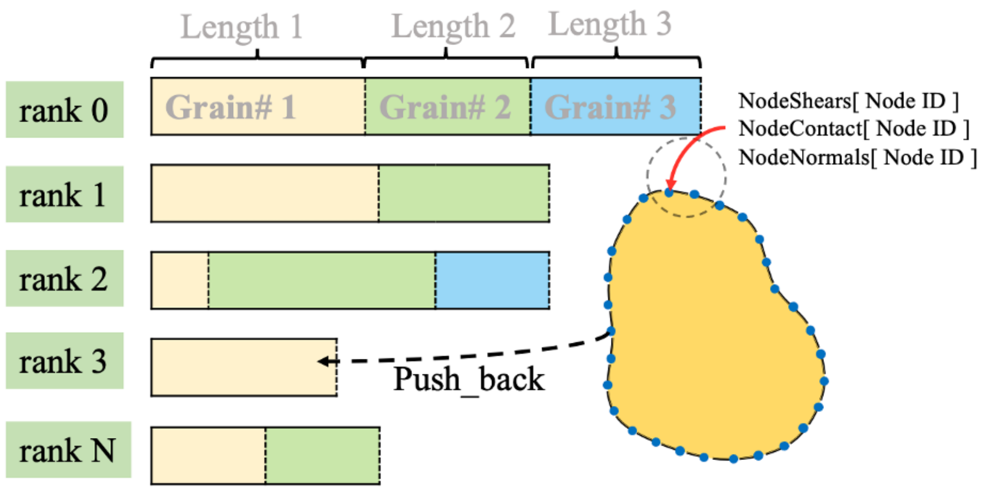

Figure 2.13: Arrangement of the data containers for contact history in the across-block migration.

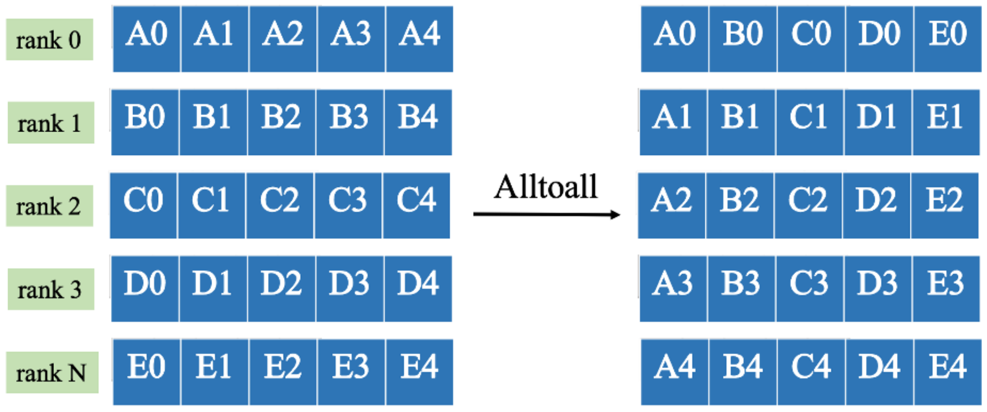

Figure 2.14: Illustration of the mechanism of MPI_Alltoall().

were modeled as undeformed planes; a grain was not allowed to leave the domain and would be bounced back if it intended to do so. For simplicity and for work balance, grains were subjected to a random force at each time step. The bookkeeping and adaptive binning were switched off when the performance was benchmarked.

Figure 2.15 shows the performance speed-ups by varying the number of processors. All speed-ups were measured relative to a serial run, which is equivalent to an MPI simulation with a $1 \times 1 \times 1$ decomposition. Each processor works on its own smaller region and maps to the closest physical memory to maximize memory bandwidth usage and avoid bandwidth limitations. Consequently, a well-balanced MPI implementation has high efficiency and keeping the communication cost low. However, there is a point beyond which more MPI processors lead to an increased MPI traffic due to the communication overheads and stalls or deteriorates scalability. In addition, increasing domain granularity also increases the area of inter-processor interfaces through which border layers are exchanged, and more ghost grains are cloned to create extra workload. An evenly distributed workload and suitable domain granularity help to hide communication overhead, i.e., if the major tasks were finished later than the communication, then a good linear speed-up is expected as the cost of communication latency is amortized. In general, increasing the number of grains should bring the scalability closer to the idealized speed up. This is because the CPU is saturated and the portion of simulation time spent on contact detection, force resolution and grain update are more significant compared with those used for communication. This also explains why the overall performance of larger test sample is better than that of its smaller counterparts. However, with increasing number of processors, MPI traffic due to data migration and increased synchronization times could add up and dominate. One may consider increasing the number of processors which is analogous to reducing the problem size. In this sense, a single parameter defined as the number of grains associated with one processor is more appropriate for optimizing the parallel execution.

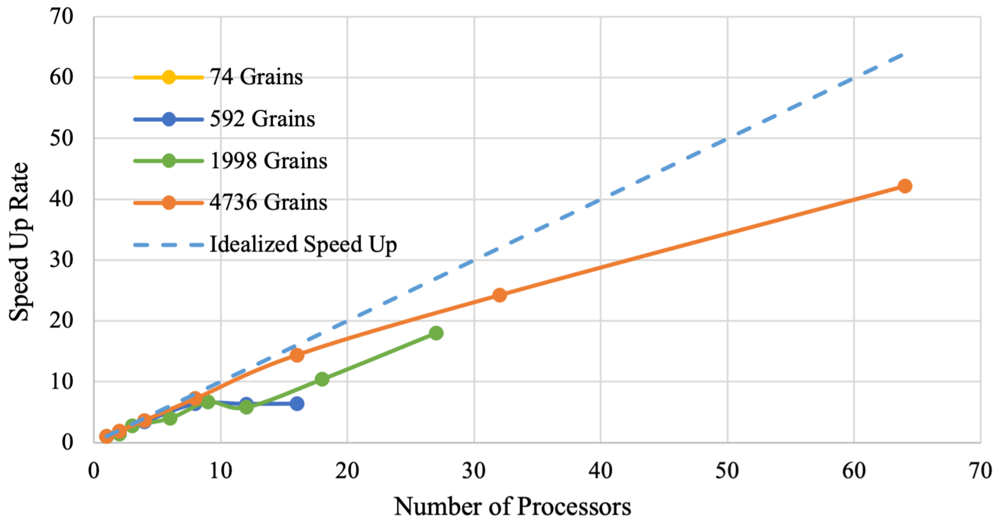

Figure 2.15: Comparison between the actual speed-up and the theoretical speed-up for a series of small numerical experiments to examine strong scalability.

Figure 2.16 shows that the running time decreases as the number of processors increases. The new code displays considerable speed-up by evenly distributing work to more processors. Reading from the plot, it takes 852 seconds to finish $1,000$ time steps with a single processor, while this number drops dramatically to 20 seconds if the parallel techniques are used. However, the running time is not a purely decreasing function of the number of processors, this is because domain granularity also considers bin size apart from the availability of computational resource. For example, to simulate 1998 grains residing in a $300 \times 300 \times 300$ domain with 2 processors and consider bin size 100. Then we can at best divide the computational domain into $100 \times 300 \times 300$ and $200 \times 300 \times 300$, and as a result, instead of experiencing 2 times speed-up, we could only save one third of computation time (256 seconds using 2 processors vs. 356 seconds using single processor). In our application, the bin size was chosen to be at least the largest equivalent grain diameter of specimen, this implies that the bin size is the minimum scale to which the computational domain could be divided, and we cannot gain parallel performance indefinitely just by using more computational resource, for instance, we cannot parallelize 74 grains in $100 \times 100 \times 100$ domain with bin size 100, and we cannot do better than 8 processors to speed up 592 grains in $200 \times 200 \times 200$ domain (the running time using 8, 12 and 16 processors should be same).

Figure 2.15 and Figure 2.16 demonstrate the $O(n)$ computational complexity of domain

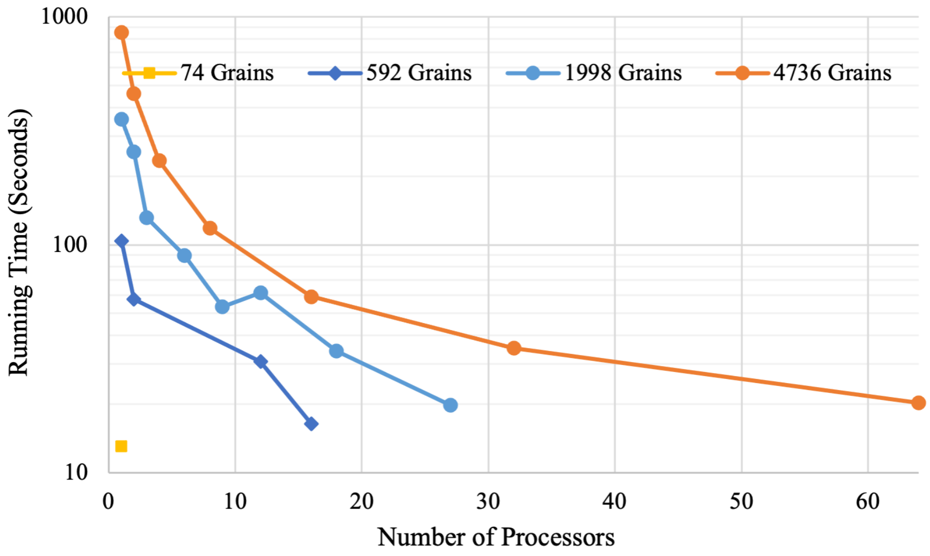

Figure 2.16: Relationship between the run-time and the number of processors for a series of small numerical experiments to examine weak scalability.

decomposition. The general pattern is almost but not perfectly linear and there are several reasons account for this. Binning algorithm is based on the idea that grain interactions would not occur if two grains are far enough apart, and DEM further assumes that the contact forces exist only between grain pairs. As a result, the contact detection step in the new code acts merely as a filter to eliminate needless calculation and this process is nonlinear. For instance, the number of grains in a bin and its neighbors is not proportional to the number of grains that come into direct contact with the grain of interest. The computation cost depends on both grain geometry and spatial distribution of the grain assembly, which may vary substantially across the assembly. The force interaction phase is distributed into tiered steps, only those pairs having passed the contact detection check are considered for force resolution, in which surface nodes are checked against the LS grid of the slave grain and the process can be further refined if a node penetrates into slave grain's surface. Those steps minimize the amount of calculation, but also introduce sophisticated nonlinearity.

The running time breakdowns are displayed in Figure 2.17. All run times are decomposed into four categories showing the amount of time spent in each phase. These statistics include: T_{force}: Time spent for contact detection and force resolution; T_{border}: Time spent in the explicit MPI border/halo exchange portion; T_{migrate}: Time spent in the grain moves to another bin or the grain migrates from host processor to new processor; and T_{update}: Time spent in parts of the code which manipulates the state of grains, meshes, and other computations such as numerical integration, and grain-wall interactions. Note that, the timing of each category is the maximum value over all MPI processors. Simulations of 4736 grains were chosen for analysis as this was a sufficient problem size to observe whether a processor was saturated or dominated by communication overheads. The grain interaction accounts for more than 90% of run time, implying that most of the computational resources were used for contact resolution. The border/halo exchange component is a function of the number of ghost grains, which is in turn controlled by the bin size and domain granularity. The grain migration is influenced by the magnitude of external forces, boundary conditions and grain densities. As can be seen, the run time for both MPI communication phases are trivial compared with force interaction computations because in this example the external forces were too small to produce significant grain movements.

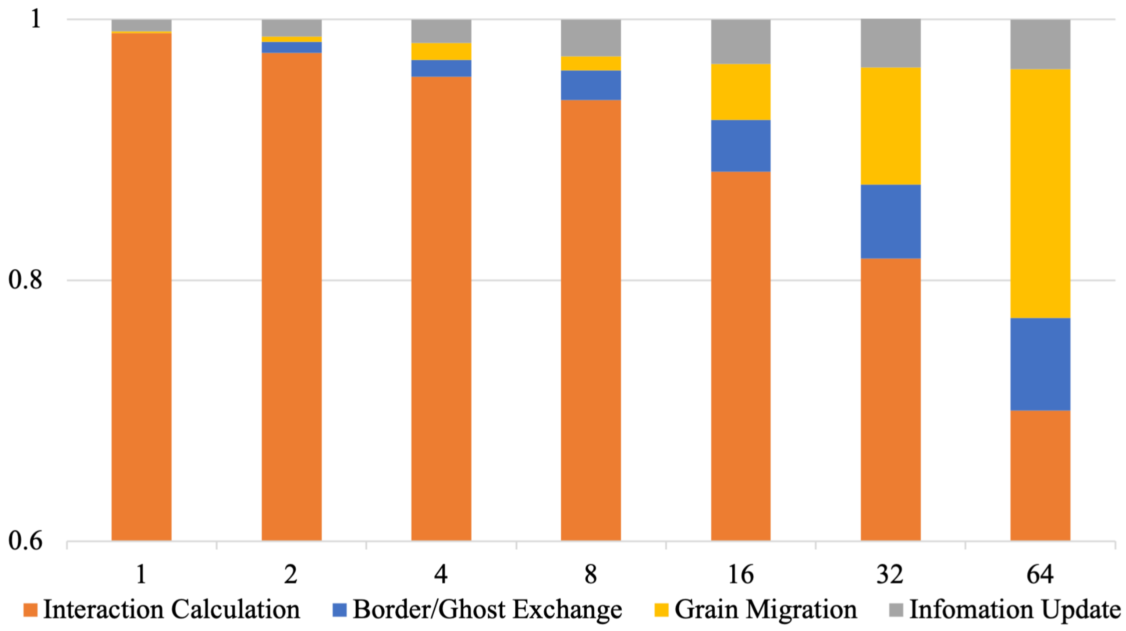

Figure 2.17: Run time decomposition for a simulation with 4763 **grains.**

The time for synchronization has been implicitly included. Consider two portions of the code where MPI communication takes place. The border/halo exchange phase is synchronized since all processors are synchronized at beginning of a new time step. Synchronization for the across-block migration is accomplished by setting barriers. It appears that work imbalance arises during force resolution, althhough MPI_Alltoall() and MPI_Alltoallv() functions are highly optimized and are capable of handling peculiar situation, e.g, transmit zero-length message. The main reason appears to be that the MPI collective operations are blocked, i.e. faster processors must wait until all processors have completed previous phase to proceed to the next. To relieve this issue, OpenMP directives are integrated in the new code, because work imbalance issue can be relaxed in the shared-memory implementation. Therefore, a hybrid type parallelism has the potential to be the best of both worlds: fully utilize the available computing resources and minimize the work imbalance via a dynamic load scheme.

The run time breakdown may be deceptive because the time for force calculation is exaggerated due to work imbalance, such that the percentage of force calculation is raised. The widely used Amdahl's law (Rodgers, 1985) does not consider work imbalance and communication synchronization. Specifically, Amdahl's law assumes that all processors compute for the same length of time, which implies that the work is perfectly load balanced. If some processors take longer than others, the speed-up declines and results in an equivalently larger serial fraction. Second, there is a missing term representing the overhead of synchronizing processors in Amdahl's law, which is a monotonically increasing function of the number of processors. A better metric is the serial fraction, f, proposed by Karp and Flatt (1990) and it is used herein to probe the effect of work imbalance due to increasing overhead. The increasing overhead decreases the speed-up and increasing, f, is a warning indicator that the granularity is too fine.

The speed up measured by running the same program on a varying number of processors is defined as:

$$s = \frac{T(1)}{T(p)} \tag{2.31}$$

where $T(1)$ is elapsed time with 1 processor. The issue of efficiency is related to price/performance, and is usually defined as:

$$e = \frac{T(1)}{pT(p)} = \frac{s}{p} \tag{2.32}$$

Consider the Amdahl's law which in the simplest form states:

$$T(p) = T_s + \frac{T_p}{p} \tag{2.33}$$

Where T_s is the time taken by the portion that must be run serially and T_p is the time in the parallel part. Then:

$$T(1) = T_s + T_p \tag{2.34}$$

If the fraction serial is defined:

$$f = \frac{T_s}{T(1)} \tag{2.35}$$

and the Amdahl's law can be re-written as:

$$T(p) = T(1)f + \frac{T(1)(1-f)}{p} \tag{2.36}$$

or in terms of speed up s:

$$\frac{1}{s} = f + \frac{1-f}{p} \tag{2.37}$$

The serial fraction can be solved as:

$$f = \frac{\frac{1}{s} - \frac{1}{p}}{1 - \frac{1}{p}} \tag{2.38}$$

The effect of work imbalance and synchronization is illustrated in Table 2.1. The results show that the computational effort was insufficient to saturate the CPU when more than 16 processors were used, thus a large portion of the run time was devoted to communication overhead.

The efficiency is interpreted in a similar way, as it experiences a rapid drop when more than 16 processors are used. The performance gain is most noticeable when all CPUs have a sufficient computational demand and are arranged in a balanced domain granularity. In addition, the hardware configuration, particularly the cache and the memory bandwidth also play a role in determining the overhead.

Table 2.1: Serial fraction as a function of the number of processors for simulations with $4,763$ grains.

Processors	Time (sec)	Speed-up (s)	Efficiency (e)	Serial fraction (f)
1	852.359	1.000	1.000	-
2	459.581	1.855	0.927	0.078
4	234.757	3.631	0.908	0.034
8	118.234	7.209	0.901	0.016
16	59.269	14.381	0.899	0.008
32	35.175	24.232	0.757	0.010
64	20.198	42.200	0.659	0.008

2.6.2 Limitations (Strong Scalability) of Domain Decomposition Strategy

High performance computing has two common notions of scalability: strong scalability and weak scalability. Strong scalability is defined as the solution time variation as a function of the number of processors for a fixed total problem size. In this regard, we conducted numerical tests to explore the code scalability by simulating a large number of grains. 42684 grains were constructed from relatively low-resolution images ($13 - 15\mu m$/pixel) resulting in substantially more avatars and smaller morphology files for each. The total amount of computer memory used for this test was 2.5GB and the surface of each grain was discretized into 279 nodes on the average. The entire computational domain was $600 \times 600 \times 600$, the largest possible grain radius was 40, the bin size was 50. The grains were subjected to random forces with mean magnitude about 10 times gravity to induce grain movement. The computational domain was partitioned in a way to minimize the communication area between adjacent processors. Even though the communication time tends to increase as the number of processors increases, it still took less than 1% in all runs. Different sub-domains interact with different number of ghost bins depending on their location, e.g., a sub-domain in corners has less interactions than a sub-domain on sides, and both of them have less interactions than the one in the center. This causes work imbalance, and the ratio is associated with the number of processors and bin size. Intuitively, more processors and larger bin size should yield more severe load imbalance. Table 2.2 lists the worst cases for each run. There are several layers of intricacies that determine the exact amount of work a processor undertakes. First, the total workload is not proportional to the number of bins because contact detection and force resolution are extremely convoluted and nonlinear processes, however, we can still assume that this relationship is linear because the specimen is large and dense. Second, not all ghost bins are visited equally frequently because we avoid half computation redundancy by accepting that the forces between a contacting pairs of grains matched in magnitude and opposed each other. The accurate estimation of visited times for a ghost bin depends on both location of sub-domain and the symmetric pattern used for inter-grain interactions. Again, we skip the computation and assume half of ghost bins are visited, $\alpha = 1/2$. Moreover, it is possible that no sub-domain is completely wrapped by neighbors as this requires at least 27 processors to achieve, and in such case determination of α is challenging. Finally, apart from the interactions from inner bins to ghost bins, there are also the opposite interactions from ghost bins to inner bins. This is because we only consider 13 out of 26 neighbors of an inner bin for force resolution, i.e., if we consider interactions between inner bins and their upper neighbors, we then need to incorporate interactions from bottom ghost bins for completeness. The maximum number of bins that a ghost bin could influence is 9 out of 13 (only consider half neighbors due to symmetry), if the ghost bin is below the bottom and close to the center, and the minimum number is 1 out of 13 if the ghost bin is at the corner. Therefore, another multiplier $\beta = 1/13 \sim 9/13$ should apply. The estimated workload is computed accordingly and the true code speed up is tabulated in Table 2.3.

Table 2.2: Runtime breakdown for simulations of $42,684$ grains in a $600 \times 600 \times 600$ domain, with bin size 50, simulations ran for $1,000$ steps, (time is in seconds).

Processors	T_{force}	T_{border}	T_{migrate}	T_{update}	Theoretical #Bins	Actual #Bins	Difference #Bins
1	67083.43	0.00	0.00	82.30	1728	1728	0
2	32727.49	23.74	0.95	51.41	864	1008	144
4	17836.61	22.54	1.14	33.39	432	588	156
6	14422.88	21.09	0.94	41.72	288	504	216
8	12661.43	20.17	0.36	36.25	216	343	127
12	9261.44	16.47	1.13	37.68	144	294	150
16	6525.09	13.68	1.00	22.58	108	245	137
18	8814.76	15.02	1.21	27.10	96	252	156
24	7118.26	15.11	1.33	26.13	72	210	138
27	4896.77	18.87	1.54	20.67	64	216	152
32	4940.85	20.04	2.68	22.46	54	175	121
36	5545.15	13.70	2.35	19.05	48	180	132
48	4495.77	12.45	2.67	19.29	36	150	114
54	4185.16	16.18	1.17	17.90	32	144	112
64	3900.41	17.82	1.71	17.66	27	125	98
72	3431.75	11.82	3.34	27.36	24	120	96
96	3218.85	11.96	4.36	14.36	18	100	82
108	3486.49	11.32	2.36	13.04	16	96	80
144	3135.79	15.74	4.41	12.06	12	80	68
216	2758.50	13.19	6.29	10.02	8	64	56

As shown in Figure 2.18, the measured speed-up is close to the approximate mean speed-up. Super linearity is observed such that parallel efficiency is greater than one, which is not uncommon in benchmarking a parallel code because performance gain from augmenting total cache size outweighs communication overhead, especially when the number of processors used is small. We should point out that the domain decomposition is a natural and recognized parallel strategy

Table 2.3: Estimated workload for different numbers of processors, and the corresponding theoretical speed-up. N_{Lower}, N_{Upper}, N_{Middle}, N_{True}: lower bound, upper bound, mean, and measured speed-up.

Processors	Upper ($\beta = 9/13$)	Upper ($\beta = 1/13$)	Upper ($\beta = 5/13$)	N_{Lower}	N_{Upper}	N_{Mean}	N_{True}
1	1728	1728	1728	1.00	1.00	1.00	1.00
2	1036	947	991	1.67	1.82	1.74	2.05
4	618	522	570	2.80	3.31	3.03	3.76
6	546	413	479	3.17	4.19	3.61	4.65
8	367	289	328	4.70	5.97	5.26	5.30
12	323	231	277	5.35	7.50	6.25	7.24
16	271	187	229	6.37	9.24	7.54	10.28
18	282	186	234	6.13	9.29	7.38	7.61
24	237	152	194	7.31	11.40	8.90	9.43
27	245	152	198	7.05	11.39	8.71	13.70
32	198	124	161	8.72	13.96	10.73	13.58
36	205	124	165	8.41	13.92	10.49	12.10
48	172	102	137	10.05	16.98	12.63	14.92
54	166	97	131	10.44	17.89	13.18	16.03
64	144	84	114	12.01	20.69	15.20	17.20
72	138	79	109	12.48	21.77	15.86	19.55
96	116	65	91	14.93	26.46	19.09	20.84
108	111	62	87	15.51	27.80	19.91	19.24
144	93	51	72	18.57	33.73	23.95	21.40
216	75	40	58	23.11	42.87	30.03	24.32

for many problems beyond the scope of DEM modeling. Our application encourages us to probe why the performance gain would cease at some point if we were to further increase computing resources, even though the communication cost remained at low levels. This will no longer be an issue if the bin size is much smaller than the size of sub-domain and a better linearity is expected.

2.6.3 Weak Scalability of Domain Decomposition Strategy

Our series of tests implies that strong scalability does not hold for domain decomposition strategy if the bin size is significant compared to the simulation geometry. Despite the inherent load imbalance nature and extra workload presented by the domain decomposition, our code nevertheless creates very small amount of communication overheads. Therefore, it we also studied the weak scalability of the code to show how the solution time varies with the number of processors with a problem size per processor fixed. We designed and conducted a series of numerical simulations (Table 2.4). The largest possible grain radius in the simulations was in the $30 \sim 40$ range, therefore, the bin size was 50 for all. All simulations ran for $1,000$ time steps.

The sub-computation domain size was identical for all test series, and the number of pro-

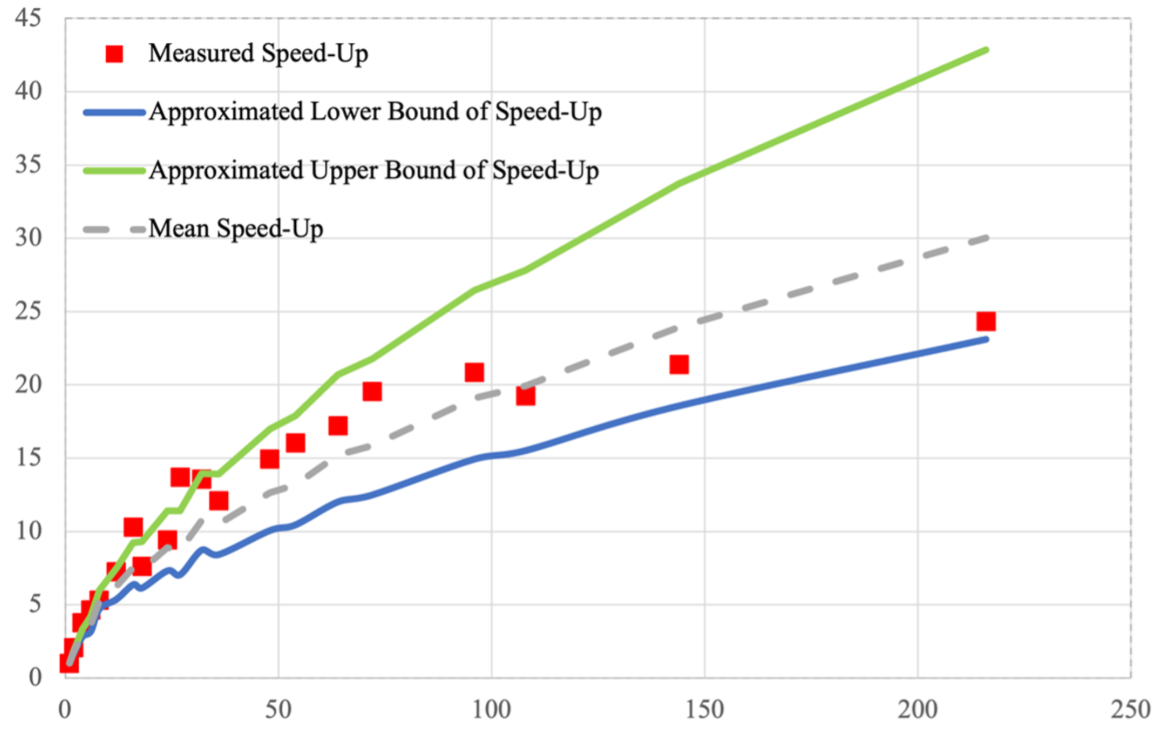

Figure 2.18: Corrected speed up relationship considering actual workload on processors, study of strong scalability.

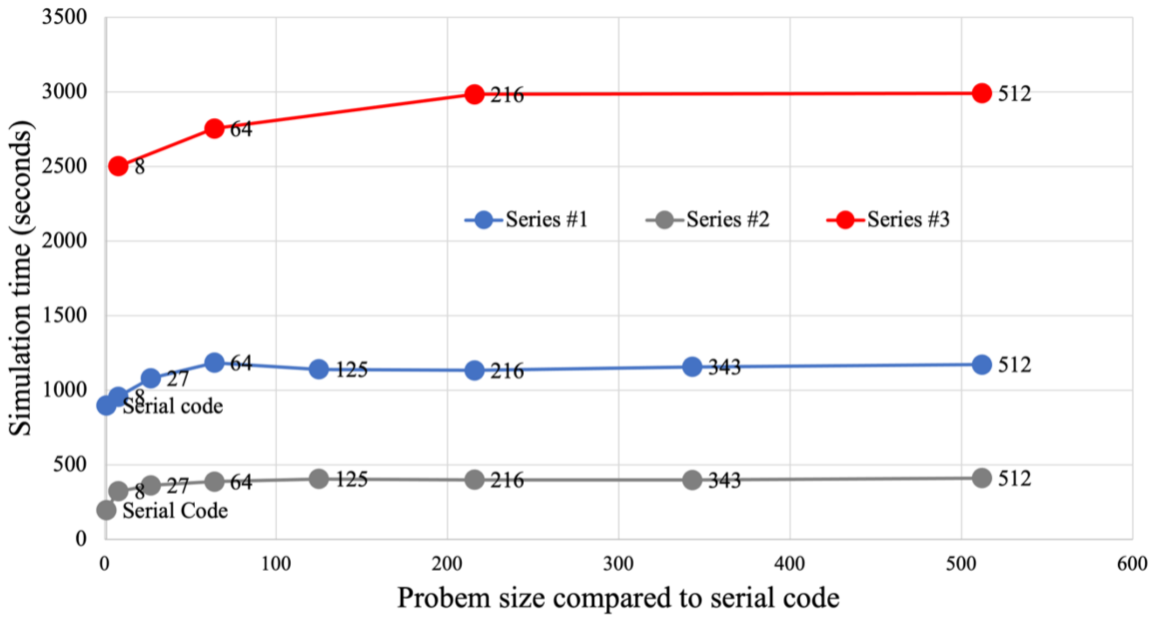

Figure 2.19: Simulation times for problems of different sizes and a fixed workload for processors, study of weak scalability.

Table 2.4: Domain angularity parameters for studies of weak scalability

Grains	Domain size	Processors	Theoretical #Bins	Actual #Bins	Simulation Times(s)
Serie #1	sub-domain size: $200 \times 200 \times 200$				
12494	$400 \times 400 \times 400$	8	64	125	2501.74
99952	$800 \times 800 \times 800$	64	64	216.	2755.65
337338	$1200 \times 1200 \times 1200$	216	64	216	2984.45
799616	$1600 \times 1600 \times 1600$	512	64	216	2991.20
Serie #2	sub-domain size: $150 \times 150 \times 150$				
481	$150 \times 150 \times 150$	1	27	27	195.95
3848	$300 \times 300 \times 300$	8	27	64	321.87
12987	$450 \times 450 \times 450$	27	27	125	360.88
30784	$600 \times 600 \times 600$	64	27	125	386.64
60125	$750 \times 750 \times 750$	125	27	125	404.79
103896	$900 \times 900 \times 900$	216	27	125	399.62
164983	$1050 \times 1050 \times 1050$	343	27	125	398.47
246272	$1200 \times 1200 \times 1200$	512	27	125	410.13
Serie #3	sub-domain size: $200 \times 200 \times 200$				
1328	$200 \times 200 \times 200$	1	64	64	896.66
10624	$400 \times 400 \times 400$	8	64	125	955.22
35856	$600 \times 600 \times 600$	27	64	216	1081.27
84992	$800 \times 800 \times 800$	64	64	216	1185.02
166000	$1000 \times 1000 \times 1000$	125	64	216	1139.87
286848	$1200 \times 1200 \times 1200$	216	64	216	1133.78
455504	$1400 \times 1400 \times 1400$	343	64	216	1155.67
679936	$1600 \times 1600 \times 1600$	512	64	216	1172.96

cessors was chosen to ensure the entire domain was decomposed into identical pieces. As illustrated in Figure 2.19, the serial code does not incur extra ghost bin interaction and therefore takes less time to finish, and simulations with 8 processors are slightly faster than others for similar reasons. Again, we are only able to count the number of extra ghost bins that are potentially involved but less sure about how intensely these bins are visited. For all other domain granularity using 27 or more processors, the computing times are close and display a slightly increasing trend with an increasing number of processors due to the communication overheads. This suggests that weak scalability still holds even though the strong scalability does not.

Finaly, we benchmarked the code with up to $700,000$ grains and investigated both strong and weak scalability for the domain decomposition strategy designed specifically for our application. Our code can simulate problems of comparable size and consumes less than 5% of the computing resources required by the original LS-DEM code. The various performance-critical subtleties are optimized so that communication overheads are kept to well below 1% of total computing wall time in favorable configurations, that is, each processor is saturated with sufficient, balanced work. Due to the low communication overhead, the weak scalability is nearly perfect, i.e., the same computing time for twice larger problems using twice the resources. By contrast, the strong scalability rapidly degrades as more processors are used. This is because the sub-domain located in the center of entire computational space requires more ghost bins from neighboring sub-domains to complete halo layer exchange, force resolution, and grain migration than its edge and corner counterparts. In other words, if the bin size is significant in comparison to the total domain size, the domain decomposition strategy is inherently load imbalanced. If we instead compute and consider the true workload of each sub-domain, the measured performance gain matches the expected value reasonably well. Again, this issue is unique to our application of simulating a densely packed grain system in a quasi-static setting, where the bin size has to be larger than the largest grain diameter in an assembly and not insignificant in comparison to the overall specimen configuration.

2.7 CONCLUSIONS

A parallel code for 3D LS-DEM to model arbitrary-shaped granular materials has been designed and implemented in C++, building on an existing LS-DEM framework developed by Kawamoto et al. (2016). A binning algorithm was introduced, which effectively reduces the computational complexity from $O(n^2)$ to $O(n)$. The newly implemented code maps relationship between bins and grains with linked-list like data structure and considers MPI communication in two major parts: border/halo exchange and across-block migration. The time complexity of execution time, communication time and parallel overhead of proposed code are analyzed with regard to the amount of computational resources and the problem size. The result shows that the proposed parallel code has an excellent weak scalability numerically and has the potential for simulating large scale DEM problems with complex-shaped grains.

The code also has high potential to have a negligible serial fraction and a low parallel overhead for a large, uniformly distributed assembly when executing on modern multiprocessing supercomputers. An important advantage of the MPI implementation is that the code is able to run on wide variety of parallel systems, including shared-memory computers, and clustered systems. As a result, the code is highly portable. Future effort to simulate large-deformation problems will require the implementation of an adaptive dynamic mesh or quad-tree algorithm to take advantage of the computational speed offered by the parallel code.

3 LS-DEM Modeling of Naturally Deposited Sand in Triaxial Compression

3.1 INTRODUCTION

The motivation for the development of a more efficient LS-DEM code was our goal to numerically explore the mechanical behavior of dense assemblies of sand particles, such as those that occur in naturally deposited sands. In this chapter, we present a parametric evaluation of the conditions necessary for successful modeling of sand subjected to triaxial loading and we present a LS-DEM simulation of a full-scale triaxial tests with a systematically calibrated contact model. The grain morphology of the naturally deposited sand is reconstructed with great fidelity from XRCT images, and a realistic confining boundary is reproduced by modeling a flexible membrane composed of bonded spheres. A numerical triaxial test was then conducted by mimicking a conventional laboratory test.

3.2 LS MODEL CONFIGURATION

3.2.1 Numerical Apparatus

A conventional triaxial test set up consisting of a cylindrical sample encased in a flexible membrane was modeled using a LS representation. The end platens were attached to the flexible membrane and the top platen was allowed to move vertically and pivot about the ram-platen contact point (Figure 3.1). We found that the rotation of the upper platen is a significant factor in determining the type of shear band formed. The upper platen rotates as a result of the uneven force generated by the complex arrangement structure and the interaction of the encased avatars. Accordingly, the specimen deforms asymmetrically and, at higher confining pressures, the samples tend to generate a single shear band. In comparison, if the upper platen does not rotate during shearing, the sample dilation is more uniform in the radial direction and an X-shaped shear bands form, (see e.g. Liang and Zhao, 2019; Wu et al., 2021).

In our simulations, the triaxial tests were performed in quasi-static conditions in two principal stages: isotropic compression and deviatoric loading. After the avatars were reconstructed in place and the material properties were assigned according to the contact-stiffness model, the grain assembly was subjected to isotropic compression with a confining pressure of $\sigma_c = 100\text{kPa}$ over

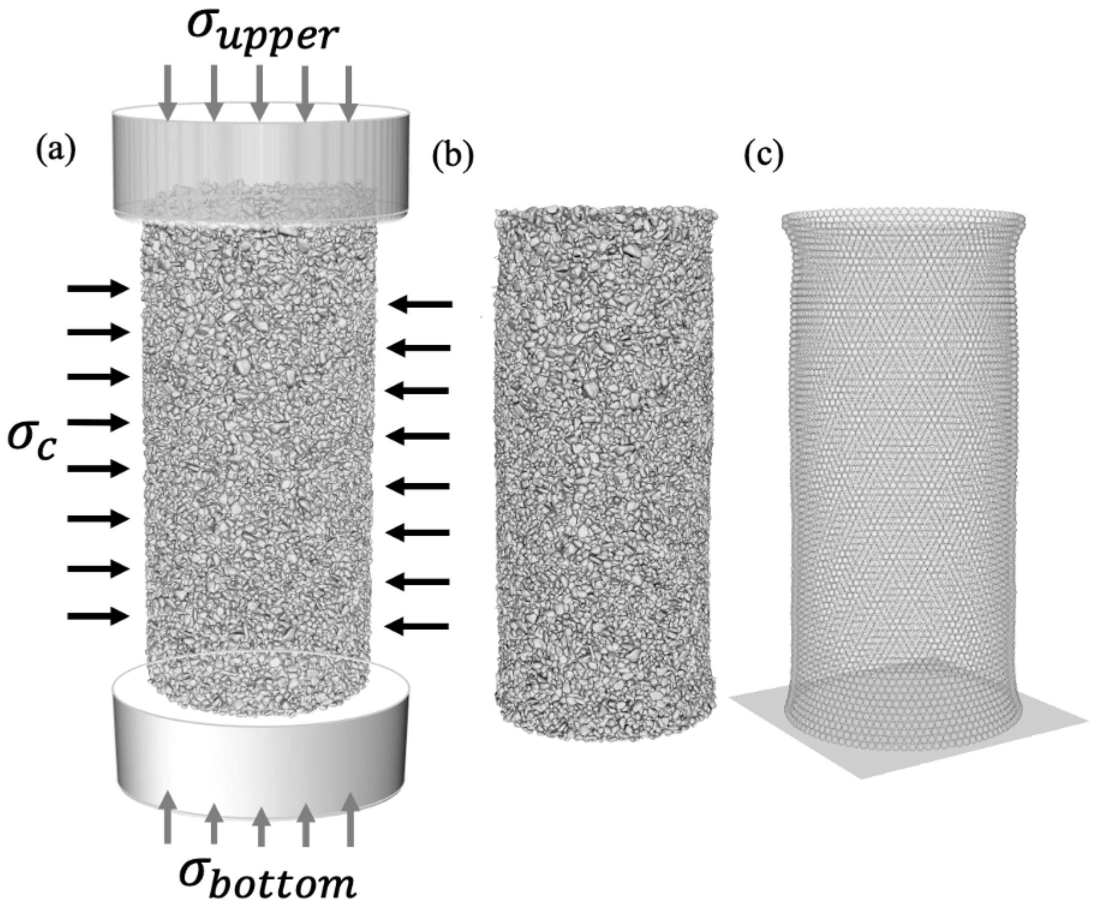

Figure 3.1: Isotropic consolidation (a) before consolidation; (b) after consolidation; (c) membrane deformation.

a large number of time steps to reach mechanical equilibrium. At the end of the consolidation, the middle portion of specimen slightly shrank, which was consistent with experimental observations. The axial loading was then applied using a strain-controlled scheme with a sufficiently low loading rate to maintain the quasi-static condition. The loading was performed in a zero gravity environment to avoid the effect of gravity-induced inhomogeneity.

3.2.2 Model Scaling

The LS avatars produced from XRCT images, numerically recreate the volume, the moment of inertia, and the mass center of the particles. Therefore, a scaling factor is necessary to link numerical models to the laboratory triaxial tests. The value of scaling factor is the XRCT image resolution and varies as per the instrument, typically between $3\mu m$/pixel and $30\mu m$/pixel. A smaller value represents a more accurate capture of the grains shapes and fabric while also inferring a higher numerical cost for both reconstruction and simulation. High-resolution avatars have more nodes to represent their geometry and to participate in force resolution. However, increasing the number of nodes on the avatar surfaces increases the computational effort because contact detection and force resolution are conducted at node level, even though parallel like domain decomposition aids in reducing search complexity and optimizing work balance. We can use either meters or pixels as the units of length. Accordingly, other parameters are converted to their equivalents and are listed in Table 3.1. While the virtual experiment appears to be independent of the scaling factor, as long as the units are consistent throughout the modeling, the confining pressure must be correctly scaled because it affects the mechanical equilibrium of the membrane spheres under confining pressure where the scaling factor does not cancel out.

Table 3.1: Unit conversion with scaling factor k ($\mu m/$pixel)

	Unit in experiments	Unit in numerical models
Density	$\rho\ (kg/m^3)$	$\tilde{\rho} = \rho \cdot k^3\ (kg/\text{pixel}^3)$
Stiffness	$s\ (N/m = kg/s^2)$	$\tilde{s} = s\ (kg/s^2)$
Pressure	$P\ (N/m^2 = kg/m \cdot s^2)$	$\tilde{P} = P \cdot k\ (kg/m \cdot \text{pixel}^2)$
Global Damping	$\xi\ (1/T)$	$\tilde{\xi} = \xi\ (1/T)$
Friction Coefficient	$\mu\ (1)$	$\tilde{\mu} = \mu\ (1)$
Length	$L\ (m)$	$\tilde{L} = L \cdot k$ (pixel)

3.2.3 Model Parameters

Two specimens were generated from XRCT images on two different samples taken from the same site but with different image resolutions of $4.3\mu m$/voxel and $9 \sim 10\mu m$/voxel, respectively. The XRCT images with the lower were less clear and produced less angular avatar shapes and eroded grain morphologies. Thus, while the high-resolution scan was able to preserve the in-situ void ratio of the sample, the low-resolution scan failed to capture significant amount of surface roughness, inter-grain contact and more importantly, small grains. As a result, the void ratio of the reconstruction from the low-resolution scan was as high as two, in and the avatars were left unsupported by neighbors. Notably, very tiny avatars are often omitted from simulation to guarantee a larger time step, but the total volume is negligible.

Figure 3.2 shows the two different samples that were reconstructed from the scans. We used the smaller, high-resolution scan for the parametric studies in majority of the simulations, with the sample reconstructed from the low-resolution scans used to illustrate the importance of accurately reconstructing the original fabric of the sand. The microscopic parameters used for the LS-DEM investigation of triaxial compression test are tabulated in Table 3.2.

3.3 NUMERICAL CONSIDERATIONS

3.3.1 Numerical Integration Scheme

The penalty-based DEM calculates the resultant force acting on objects and then updates accelerations and velocities using a numerical integrator. It is a second-order accurate finite difference

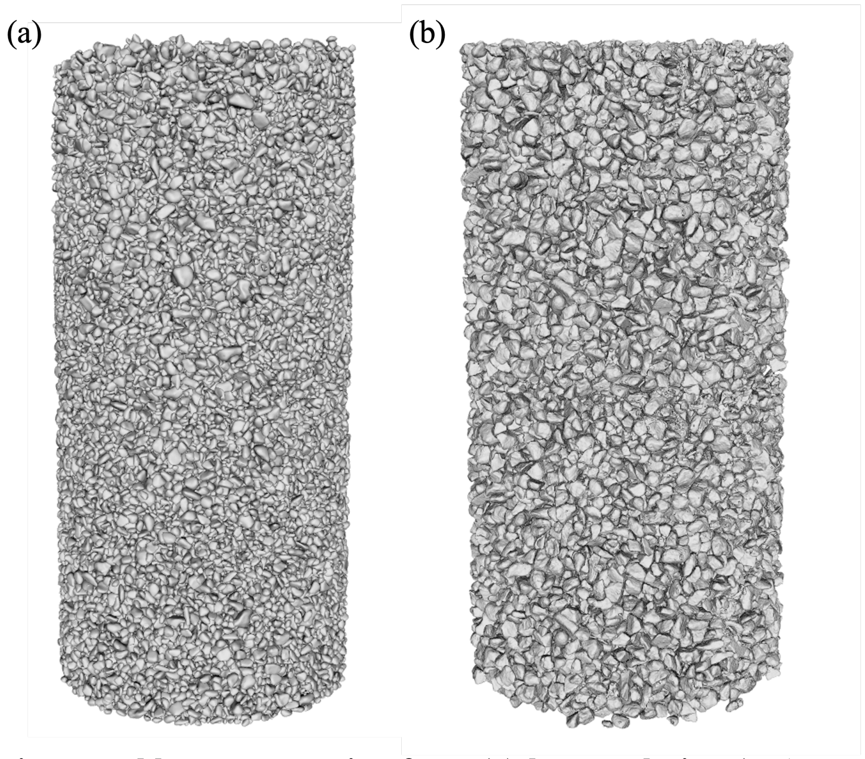

Figure 3.2: Grain assembly reconstruction from (a) low-resolution ($\sim 70,000$ grains, $9 \sim 10\mu m$/voxel); (b) high-resolution ($\sim 17,000$ grains, $4.3\mu m$/voxel).

scheme in time-centered form Walton and Braun (1993), which takes into account the 'tangent' change in velocity.

$$\mathbf{v}^{t+\frac{1}{2}} = \mathbf{v}^{t-\frac{1}{2}} + \dot{\mathbf{v}}^t \Delta t \tag{3.1}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^{t+\frac{1}{2}} \Delta t \tag{3.2}$$

Which is equivalent to Lee and Hashash (2015):

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \dot{\mathbf{x}}^t \Delta t + \frac{1}{2} \ddot{\mathbf{x}}^t \Delta t^2 \tag{3.3}$$

To guarantee the numerical stability, the central finite difference schemes adopt a time step that has to be equal to or less than a critical time step Δt_{cr} because this scheme is conditionally stable. The calculation logic of time step for explicit DEM is based on Newton's second law. To ensure the solution produced by the model remains stable, the time step in each calculation cycle should not exceed a critical time step that is related to the stiffnesses, densities, geometries of modeled objects and minimum eigenperiod of the granular assembly, and often expensive to compute. Instead, Itasca (1998) estimates the critical time step for an assembly of grains using an equivalent single degree of freedom system and concludes that the critical time step is governed by the minimum grain mass m_{min} and the maximum spring stiffness K_n in an assembly.

Table 3.2: Microparameters used in LS-DEM of triaxial compression test

Parameters	Symbol	Values	Units
Global damping	ξ	1.0	$1/s$
Platen			
Platen Radius	R_p	500	pixel
Platen Height	H_p	800	pixel
Density	ρ_p	$2,500$	kg/m^3
Normal stiffness	K_{np}	3×10^4	N/m
Shear stiffness	K_{sp}	2.7×10^4	N/m
Grain-platen friction coefficient	μ_p	0.5	
Membrane			
Sphere radius	r_b	10	pixel
Sphere-sphere normal stiffness	K_{nbb}	50	N/m
Sphere-sphere tangential stiffness	K_{sbb}	50	N/m
Grain-membrane normal stiffness	K_{nb}	3×10^4	N/m
Grain			
Density	ρ_p	$2,500$	kg/m^3
Inter-grain normal stiffness	K_n	3×10^4	N/m
Inter-grain tangential stiffness	K_s	2.7×10^4	N/m
Friction Coefficient	μ	$0.60 \sim 0.75$	

$$\Delta t_{cr} \ll \sqrt{\frac{m_{\min}}{K_n}} \tag{3.4}$$

O'Sullivan and Bray (2004) performed extensive numerical experiments on the appropriate selection of Δt and recommend that a much smaller Δt should be used than Δt_{cr} to ensure the stability for two reasons: 1) insufficiently small-time step induces significant oscillations in the numerical solution; and 2) excessive energy will quickly accumulate with simulation time because of the numerical error associated with the oscillations. For those reasons, a factor of safety of $0.1 \sim 0.2$ is often used. In many granular material simulations, Δt_{cr} may be very small because the particles are assumed to be rigid. Therefore, very high contact stiffness K_n is required to prevent particle penetration. This also leads to a large number of iterations for simulation, which can be fatal in terms of demand on computational resources.

To simulate the triaxial test, the grain assemblies were vertically compressed using a constant downward displacement of the top platen while maintaining a constant confining pressure on the flexible membrane. To achieve quasi-static shearing, the shear rate must be slow enough that the kinetic energy generated by the shear is negligible. As LS-DEM is more computationally intensive than conventional penalty-based methods, we can complete only about $20,000$ iterations per day for a full-scale reconstructed specimen containing on the order of 10^6 grains.

In this case, we wanted our simulation to finish in a reasonable amount of time, which means numerically compressing the specimen to a sufficiently large deformation to observe a fully developed shear band. As a result, the critical time step Δt_{cr} and the maximum number of iterations were related via the loading rate: the amount by which the upper platen is lowered in one step. The velocities of the loading platens are described in units of m/s and $m/$step in the literature, but seldom reported in relation to the time step for describing the loading strain rate in the compression tests. For example, the recommended loading rate for compression test in the PFC user manual is 0.02 m/s (Itasca, 2004), so that inertial effects, such as platen loads in excess of their quasi-static equilibrium values are avoided. In other studies, a variety of different loading rates, ranging from 0.0016 to 0.3 m/s, were used when modeling virtual quasi-static tests on granular material (Hazzard et al., 2002; Potyondy and Cundall, 2004; Cho et al., 2007; Fakhimi and Villegas, 2007; Park and Song, 2009). These correspond to very high loading rates in the physical world. Intuitively, if the upper platen moves much faster than the grains in direct contact with the platen, responses in the upper portion of the specimen are not adequately transmitted downward, triggering dynamic effects. Our simulation was conducted in a quasi-static conditions and there are two widely used and comparable criteria for choosing a strain rate for simulation. Zhao and Zhao (2019) recommend that a loading rate less than 1% axial strain per second should be sufficiently small to numerically reproduce a quasi-static environment. Alternatively, the inertial number I_{inertia}, as defined below is used to provide quantitative insights into the selection of loading rate (MiDi, 2004; Da Cruz et al., 2005):

$$I_{\text{inertia}} = \dot{\varepsilon}\frac{d}{\sqrt{\boldsymbol{\sigma}_c/\rho}} < 10^{-3} \tag{3.5}$$

Where d is the average equivalent grain diameter in the assembly, ρ is the density of grain ($2,650$ kg/m^3), $\boldsymbol{\sigma}_c$ is the confining pressure (100kPa), and $\dot{\varepsilon}$ is the loading strain rate. The physical meaning of inertia number is the ratio of time scale of grain rearrangement to the time scale of packing shear. MiDi (2004) and Da Cruz et al. (2005) suggested that although a real triaxial compression test requires I_{inertia} to be as small as 10^{-9}, 10^{-3} is the threshold at which further decreasing I_{inertia} has no effect on the measured macro-properties. Potyondy and Cundall (2004), and Hazzard et al. (2002) demonstrate that loading velocity has little effect on the mechanical behavior of the sample being modeled as long as the velocity is low enough to avoid the generation of transient waves.

Also, the formulation of I_{inertia}, $\frac{d}{\sqrt{\boldsymbol{\sigma}_c/\rho}}$ is independent of the scaling factor k after considering the conversion relationship tabulated in previous section, while the strain rate $\dot{\varepsilon}$ is dependent on the time step Δt which is associated with the scaling factor k. Using a larger k results in a larger time step and hence smaller I_{inertia}. As a result, using a scaling factor k larger than the actual magnitude reduces computational effort and was adopted by Thornton (2000) and Ng (2006). In our work, we explored the possibility of scaling up the grain mass for a larger time step to bring the virtual triaxial simulation as close as possible to that used in the laboratory. To achieve this, we increased the scaling factor to the point where 20% axial strain can be achieved about $40,000$ iterations and the loading strain rate satisfies both practical and quantitative criteria to guarantee a quasi-static condition. It is worth noting that Lee et al. (2012) used a slightly more complicated scheme to increase loading rate while maintaining quasi-static conditions; they discretely moved the top platen at a much higher loading rate and allowed the grains to re-equilibriate due to the top platen's excessive penetration into nearby grains. Alternatively, re-formulating the penalty-based method to an impulse form is another effective way to increase the critical time step (Mirtich and Canny, 1995). In contrast to the penalty-based method, the impulse-based method does not require the use of arbitrarily large stiffness and damping parameters as it solves for velocities directly using Newton's second law's impulse-momentum form. This enables an impulse-based method to use a time step several orders larger than a penalty-based DEM, resulting in comparable computational efficiency. We implemented and evaluated the impulse-based LS-DEM code in other studies discussed in Chapter 4. The primary limitation of the impulse-based method is its inability to model a system of irregular, non-convex, non-uniform objects, particularly in a highly confined quasi-static environment in which objects of very different shapes and sizes interact at numerous contact points and have low velocities. This makes collisions within a contact group extremely difficult to solve using a velocity-based algorithm.

As previously stated, the scaling factor k can be increased to allow for a larger time step in order to prolong the time can be modeled. This measure, however, has an effect on the maximum confining pressure applied to the membrane, as it results in excessive overlap between the membrane and the specimen. Similar issues were also reported by De Bono et al. (2012). Thus, increasing the scaling factor requires increasing both the grain-membrane and inter-grain stiffness, or decreasing the magnitude of confining pressure. We conducted a series of simulations to analyze the influence of confining pressure while maintaining a constant confining pressure to contact stiffness ratio. This allowed us to explore how the scaling factor affects the simulation results, as setting the scaling factor to its true value is computationally very intensive. The true scaling factor is on the order of 10^{-5} and the contact stiffness is on the order of 10^{5}, in order to maintain equilibrium against a 100kPa confining pressure. When the scaling factor is increased from 10^{-5} to 10^{-2}, either the contact stiffness should be increased proportionately, or the confining pressure should be decreased to retain the force balance of membrane spheres. In the simulations illustrated in Figure 3.3, the confining pressure $\boldsymbol{\sigma}_c$ of 1, 10, and 100kPa corresponds to using 100, 10, and 1 times larger time step, respectively.

The average macroscopic stress of the granular assembly is defined as:

$$\bar{\boldsymbol{\sigma}}^g = \frac{1}{\Omega_g} \sum_{c=1}^{N_c^g} \text{sym}(\mathbf{f} \otimes \mathbf{x}^c) \tag{3.6}$$

Where $\mathbf{x}^c$ is the vector defined from the origin to the point of application of contact force $\mathbf{f}^c$ at the contact point c, with a total number of contact points N_c^g in the grain g. This is the discrete form of that presented by Christoffersen et al. (1981). A more generalized average stress theorem considering body forces is given by Zohdi and Wriggers (2004). In three-dimensional setting, two often referred quantities: the mean effective stress $\mathbf{p}$ and the deviatoric stress $\mathbf{q}$, are derived from this formula:

$$\mathbf{p} = \frac{1}{3}\bar{\boldsymbol{\sigma}}^g \tag{3.7}$$

$$\mathbf{q} = \sqrt{\frac{3}{2}\bar{\mathbf{s}}^g : \bar{\mathbf{s}}^g} \tag{3.8}$$

The peak mobilized friction angle is then calculated as:

$$\phi_p = \sin^{-1}(\frac{\sigma_1 - \sigma_3}{\sigma_1 + \sigma_3})_p \tag{3.9}$$

Where the values of σ_1 and σ_3 are taken at the peak of the stress-strain curve. Similarly, the dilatancy angle ψ can be estimated from the volumetric strain versus axial strain curve of a material subjected to triaxial compression with the following expression (Schanz and Vermeer, 1996; Salgado et al., 2000):

$$\psi = \sin^{-1}(\frac{\dot{\varepsilon}_v/2\dot{\varepsilon}_a}{2 - \dot{\varepsilon}_v/2\dot{\varepsilon}_a}) \tag{3.10}$$

The axial strain ε_a and volumetric strain ε_v can be determined based on the displacement of the loading platen and volume change of the encasement according to the following equations:

$$\varepsilon_a = -\int_{L_0}^{L} \frac{1}{l} dl = \ln \frac{L_0}{L} \tag{3.11}$$

$$\varepsilon_v = -\int_{V_0}^{V} \frac{1}{v} dv = \ln \frac{V_0}{V} \tag{3.12}$$

Where L_0 and V_0 are the initial height and volume of the isotropically consolidated specimen right before shearing, and L and V are the quantities of the deformed specimen during the course of shearing.

The simulations show no obvious differences in deviatoric stress, peak mobilized friction angle, or dilatancy between the three sets of results, although there are some minor fluctuations as the specimen approaches the critical state. However, this is of less interest for our current study, and it is believed that this is partly attributable to inaccuracy in volume integration due to the severe membrane distortion at axial strains in excess of 10%. All tests conformed with the inertia number requirement ($< 10^{-3}$) to retain the quasi-static condition.

3.3.2 Flexible Membrane Modeling

For the sake of simplicity, the flexible membrane in our simulated triaxial tests is frictionless, and the degrees of freedom for each membrane sphere are reduced from six to three, obviating the need to compute moment and rotation. This assumption contributes to the model membrane stability, as numerically integrating Euler's rotation equation is a source of instability. Another problem occurs when the membrane stretches, because a few grains contained within can escape and become uncontrollable. Although this has no effect on the simulation results, it invalidates the subroutine used for dynamic domain re-decomposition. Specifically, the purpose of domain re-decomposition is to re-generate bins, blocks and sub-domains based on the current grain distribution to best balance workload across processors. When a grain escapes from the flexible membrane and flies far

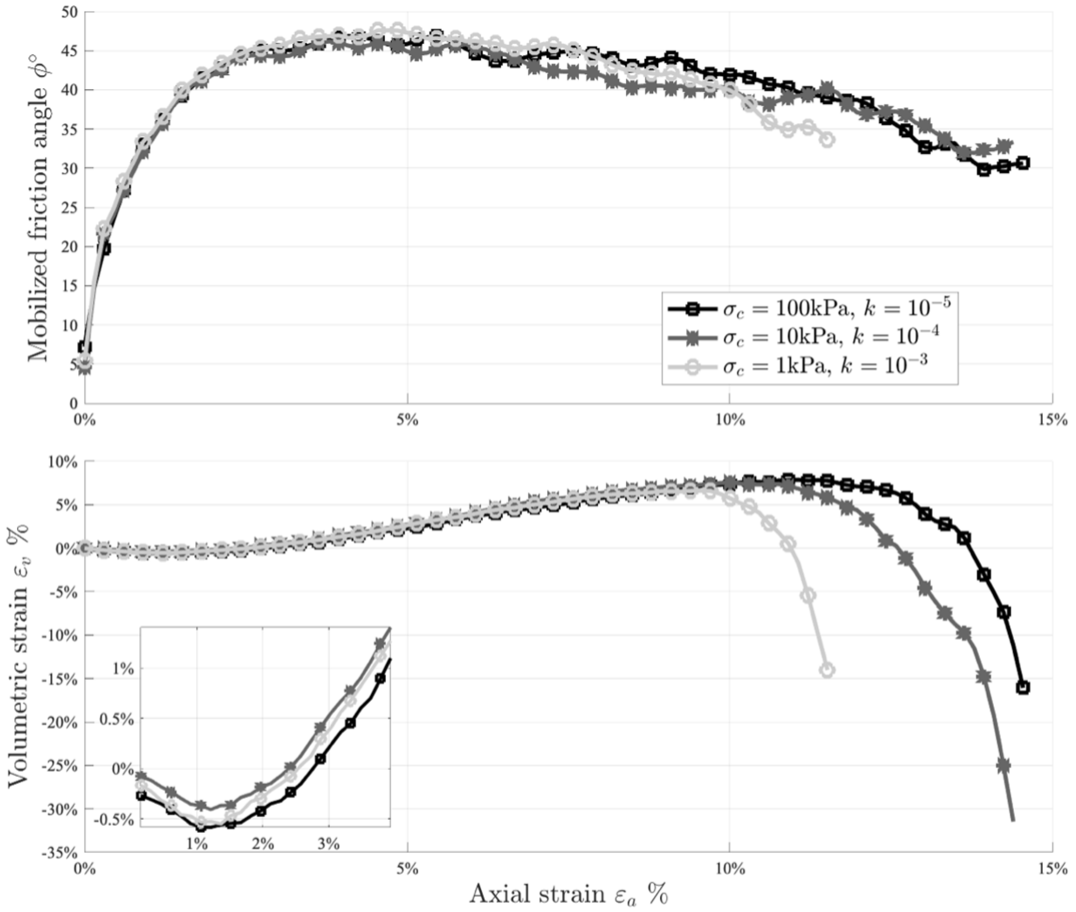

Figure 3.3: The effect of reducing confining pressure due to change of scaling factor.

away from the main body, the re-decomposition strategy will incorrectly consider a much larger domain as the current configuration and distribute grains accordingly, resulting in the majority of processors being idle and one processor performing all work. To address this issue, we used a double-layer membrane with a slightly larger sphere radius in the outer layer. This resulted in membrane spheres being displaced slightly to cover gaps in the first layer, preventing grain from escaping. The confining pressure was then applied to both layers with 80% of the pressure on the inner layer and 20% on the second layer to obtain the desired magnitude.

The computational cost of the membrane is determined by the radius ratio of the grain to the membrane sphere. Larger elements are more computationally affordable and more numerically stable, making them better able to withstand massive deformations when spheres are pushed into an incorrect geometry. However, if the ratio decreases by using larger membrane spheres, the potential for leakage of small soil grains through the membrane increases. Due to volumetric dilation caused by large deformation, the pore size (spacing between membrane spheres) can also increase. As a result, an appropriate radius ratio should be chosen to strike a balance between computational time and grain leakage. Kawamoto et al. (2018) chose the membrane sphere size as one third of the average size of the particles in the specimen, while Kim et al. (2020) selected the radius of the membrane spheres to be twice the radius of the smallest grain. The effect of membrane sphere radius was investigated for an assembly with an average equivalent radius of 18, with the size of spheres 6, 8, and 10 in the first layer of flexible membrane and 1.1 times larger in the second layer; no grain leakage was detected in all the tests. As shown in Figure 3.4, the simulation results with $r_b = 8$ and $r_b = 10$ matched each other, whereas the simulation with $r_b = 6$ suffered numerical instability at high shear strain and manifested itself by uncontrolled volumetric contraction. The stability analysis of membrane element is analogous to the grain assemblage stability in that both can be thought as a mass-spring-damper system

$$m\ddot{\mathbf{x}} + c\dot{\mathbf{x}} + k\mathbf{x} = \mathbf{0} \tag{3.13}$$

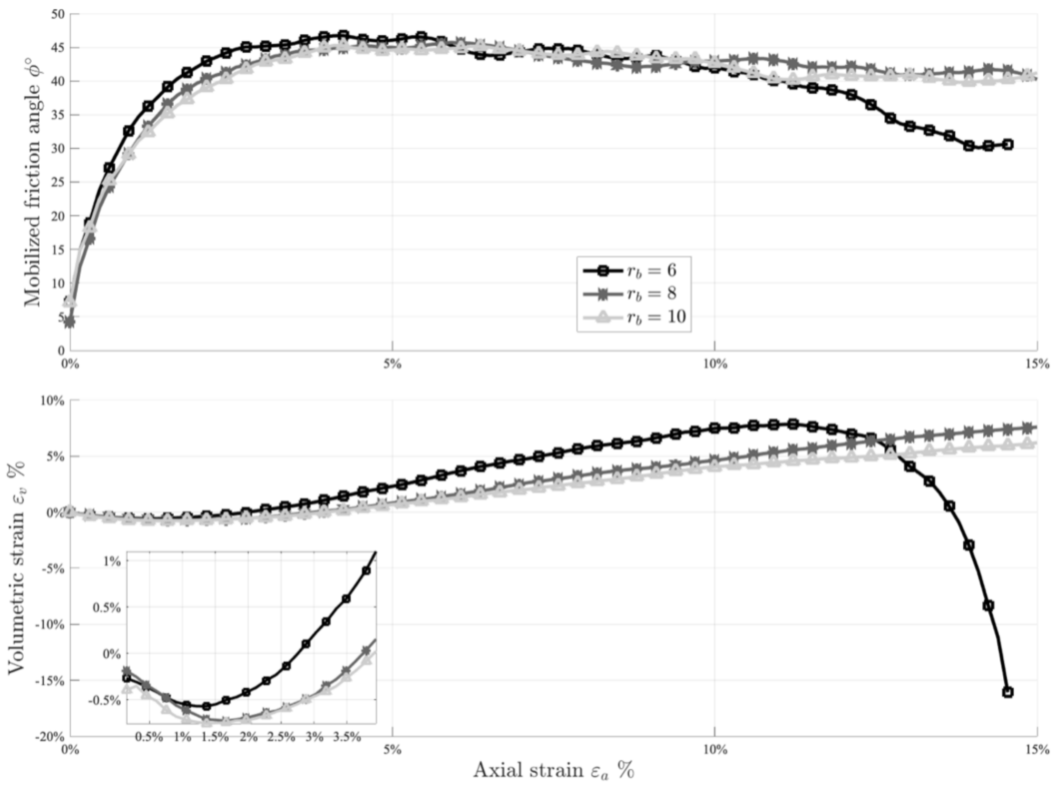

Figure 3.4: The influence of membrane sphere radius.

The contact model between membrane spheres can then be calibrated to match the behavior of the particle-based membrane to that of a real membrane at the selected radius ratio. The effect of membrane stiffness K_m on the macroscopic behavior of a granular assembly was investigated using a series of parameter studies ranging from $50N/m$ to $1,000N/m$, as shown in Figure 3.5. The results show that the membrane stiffness does not have significant influence on peak mobilized friction angle, or initial dilatancy in a triaxial simulation. However, as would be expected, a stiffer membrane provided a stronger support for the specimen, as indicated by a slightly larger elastic modulus and less volumetric contraction. By making the boundary flexible, the physical integrity of specimen is maintained while its tolerance for large deformations is increased. In addition, the system gains additional degrees of freedom to deform and more possibilities for shear band development.

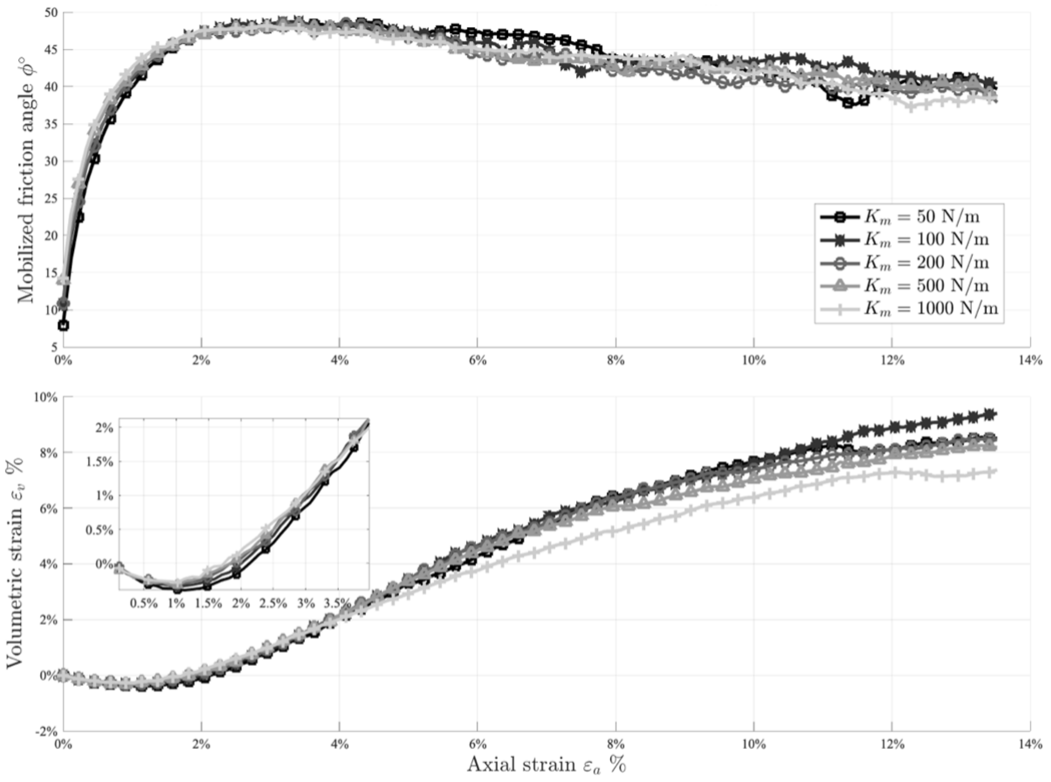

Figure 3.5: The effect of membrane stiffness K_m.

3.3.3 Numerical Stability of Angular Velocity Integration

A LS avatar integrates angular velocity and rotation using Euler's rotation equation, which is not necessary for conventional DEM simulations of spheres or discs.

$$\begin{aligned} \alpha_1 &= [M_1 + \omega_2\omega_3(I_2 - I_3) - \xi I_1\omega_1]/I_1 \\ \alpha_2 &= [M_2 + \omega_3\omega_1(I_3 - I_1) - \xi I_2\omega_2]/I_2 \\ \alpha_3 &= [M_3 + \omega_1\omega_2(I_1 - I_2) - \xi I_3\omega_1]/I_3 \end{aligned} \tag{3.14}$$

Where α_i is the angular acceleration, ω_i is the angular velocity, I_i is the (diagonal) moment of inertial tensor in the principal body-fixed frame, and M_i is the torque vector in the principal body-fixed frame.

Euler's equations are nonlinear and implicit due to the presence of angular velocities on both sides. As a result, a predictor-corrector algorithm was adopted for appropriately integrating the rotational components of motion. This method resembles a fixed-point method and is stable and convergent when the time step is sufficiently small. Since LS-DEM employs a time step smaller than the critical time step, this value also ensures the predictor-corrector algorithm's numerical stability. Additionally, global damping is used to dissipate excessive energy, and to further reduce the possibility of accumulated numerical errors. However, simulations can still occasionally fail to converge, apparently due to some grains erroneously attracting forces and moments that are too large and the predictor-corrector scheme can become unstable within a single iteration.

The principal reason for such event in our simulations is that in constructing the avatars from images of grain assemblies, grains can be accidentally reconstructed inside or cross penetrating other grains. We addressed this issue by limiting the coordination number of a grain, which is correlated with the packing density of a granular assembly. The simplest definition of the coordination number is the mean number of contacts per particles:

$$Z = \sum_{i \in N} \frac{N_c^i}{N} \tag{3.15}$$

Where N_c^i is the total number of contacts of the i-th grain and N the total number of grains in the assembly. The scalar coordination number is strongly related to the internal fabric structure of the granular material and is highly correlated with the mechanical stability (Nouguier-Lehon et al., 2003; Mitchell and Soga, 2005). In addition, the coordination number is sensitive to grain shape, which has a significant impact on deformation of a granular assembly (large drop in mean coordination number corresponding to larger dilatancy). Irregularly shaped grains result in greater coordination number. The experimental data indicates that a grain coordination in naturally deposited sands is between 8 and 16, and our parametric study shown in Figure 3.6 demonstrates that as long as the threshold value is within a reasonable range, no significant differences in the simulation results are observed. Furthermore, increasing the threshold value to 64 did not incur numerical instability. This confirms that numerical instability is primarily due to the severe overlap between avatars and hence scaling extremely large forces and moments down is a rational solution. Interestingly, reducing the coordination number to four seems to have very little impact on the simulation results. This is because the external loading consists primarily of confining pressure from the encased flexible membrane and compression forces from the downward moving platen; this leads to only a small portion of the assembly being directly subjected to those forces.

3.4 PARAMETRIC STUDY AND MODEL CALIBRATION

The objective of parameter study was to determine the suitable micro-parameters for DEM simulations. The influence of membrane properties and scaling factor were already discussed. The other model parameters were determined by trial and error until the simulation produced an initial stiffness and peak friction angle that were comparable to those obtained from a triaxial compression test on a naturally deposited sand specimen. All experiments were conducted on an assembly of $4,852$ avatars constructed from high-resolution ($4.3\mu m$/voxel), excellent-quality images such that original soil fabric was well-preserved and was found to behave consistently with real laboratory results. The number of avatars was sufficient for study of macroscopic behavior and the computational cost was affordable for parametric studies.

3.4.1 LS Avatar Node Density

Grain avatars are characterized and represented by nodes seeded on the surface and the node density is closely associated with the computational cost in inter-grain force resolution. This representation is similar to a triangulated surface mesh except that LS-DEM does not store connectivity

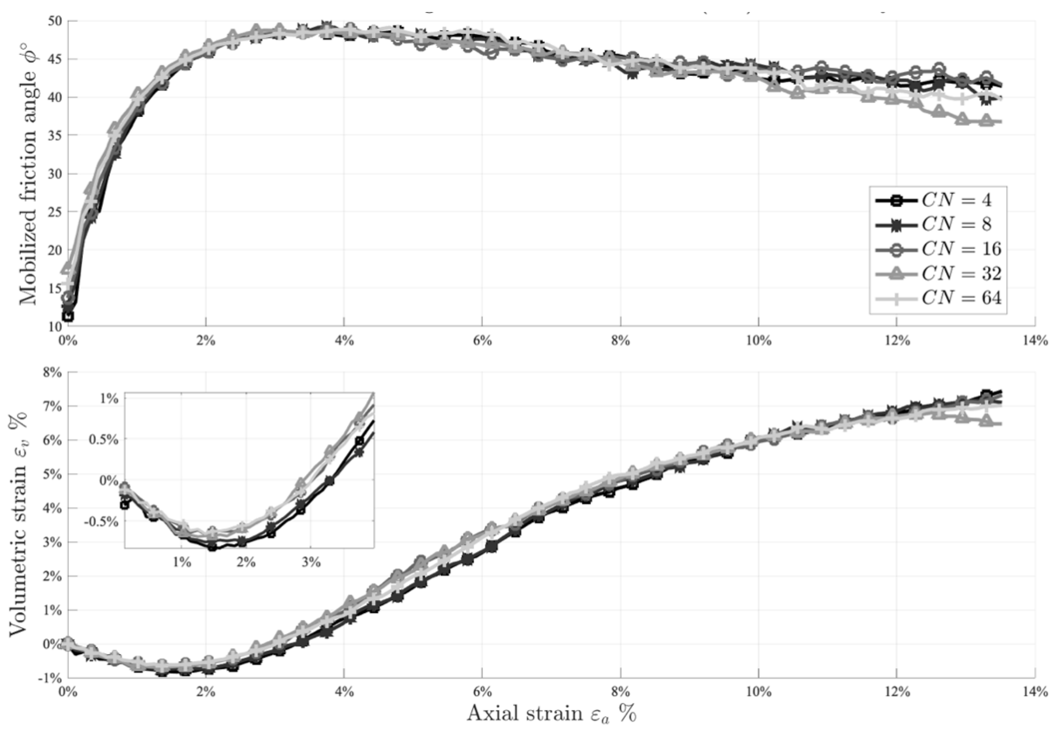

Figure 3.6: The influence of restricting coordination number to maintain numerical stability.

information between nodes and therefore does not consider edge-surface collision. The density of nodes on a grain is entirely up to the designer. It has no effect on the representing geometry but does have an effect on the computational complexity associated with force resolution.

We studied the consequences of decreasing the node density of the avatars while maintaining a balance between precise grain morphology and simulation time. To do this, we considered only the N-th nodes and skipped the rest during the force resolution step when iterating over nodes on the master avatar's surface and marked down the speed-ups. To produce similar magnitudes among different runs, we multiply the resulting forces, moments, and coordination number by N. The results in Figure 3.7 indicate that the reconstruction fidelity significantly affects the entire computation time, since decreasing node density results in linear performance benefits. The initial stiffness and maximal mobilized friction angle are reasonably consistent across simulations with $N \leq 3$. Despite the fact that the amount of dilatancy is reduced even for scenario $N = 2$, all simulations achieved the same critical condition. This data indicates that we can decrease the resolution of the avatars for a faster code execution but we do undertake the risk of missing contacts if too many nodes are skipped. While we believe that reducing node density on reconstructed avatars is a viable acceleration strategy, we are still in need of clean, high-resolution XRCT images for high-fidelity grain morphology reconstruction, as the two are fundamentally different aspects. We explored herein the implications of node density with the underlying premise that grain geometry is correctly represented and that small grains are retained to accurately duplicate the void ratio of the overall assemblage. By comparison, if the underlying data set (XRCT pictures) is significantly contaminated, we are incapable of reproducing naturally deposited soil fabric in-situ, let alone model the observed mechanical and kinetic behavior.

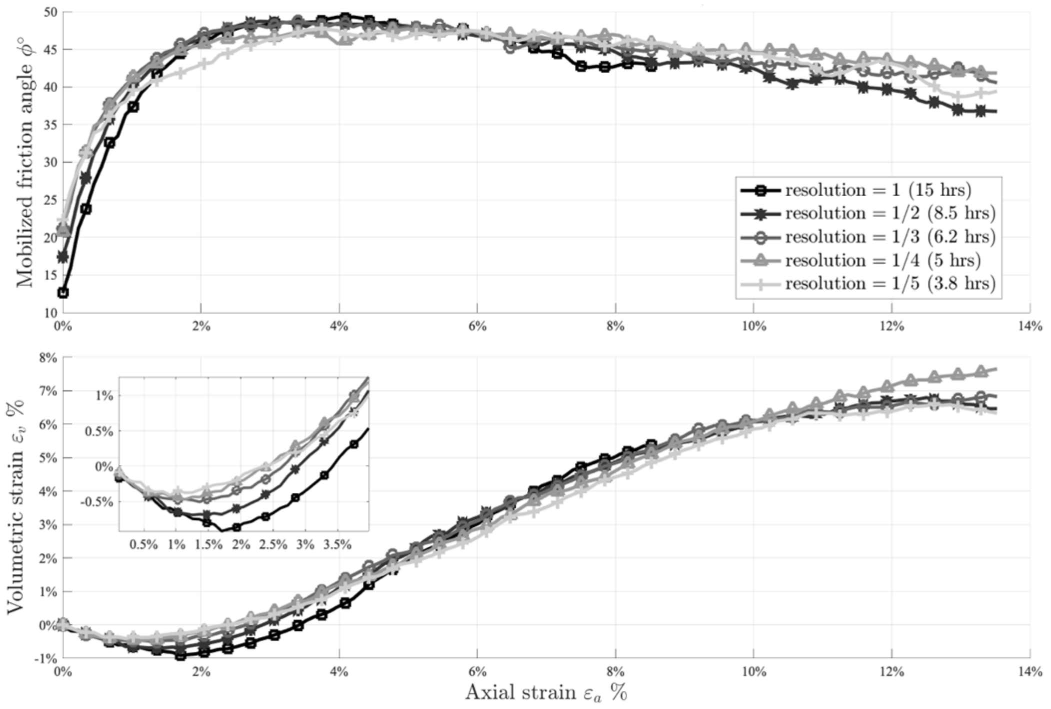

Figure 3.7: The influence of avater node density.

3.4.2 Confining Pressure

The magnitude of the confining pressure in our virtual numerical modeling depends on the membrane-grain contact stiffness, as too large a confining pressure will push membrane spheres completely into grains, leading to heavily distorted membrane and uncontrolled behavior afterwards. Thus, the ability to withstand high confining pressure indicates that the assembly is well-packed, as membrane penetration effect will result when the packing is loose and when the membrane elements are small. The range of confining pressure can be estimated from the force equilibrium relationship of membrane spheres and the maximum value was found to be about 500kPa with the membrane-grain stiffness fixed at $30,000$N/m. This was enough for our objective to investigate soil behavior under non-crushing stress environment. As would be expected the results (Figure 3.8) show that the peak mobilized friction increases only slightly, while the stiffness increases significantly. When the confining pressure $\boldsymbol{\sigma}_c$ exceeded 200kPa, the membrane elements entered into the specimens, creating an instability which halted the simulation at axial strains greater than about 7%. An alternative membrane construction could be used to model tests with higher confining pressures.

3.4.3 Normal and Shear Stiffness Ratio

Two linear springs are inserted at contact between grains, one for the normal stiffness specified by the contact K_n and another for the shear stiffness specified by K_s. Normal force is proportional to normal stiffness and the overlap of two grains, and the shear force is proportional to shear stiffness and the relative rotation of two grains. The effect of altering the ratio of normal to shear stiffness K_s/K_n was calibrated using a series of simulations. The friction coefficient has been

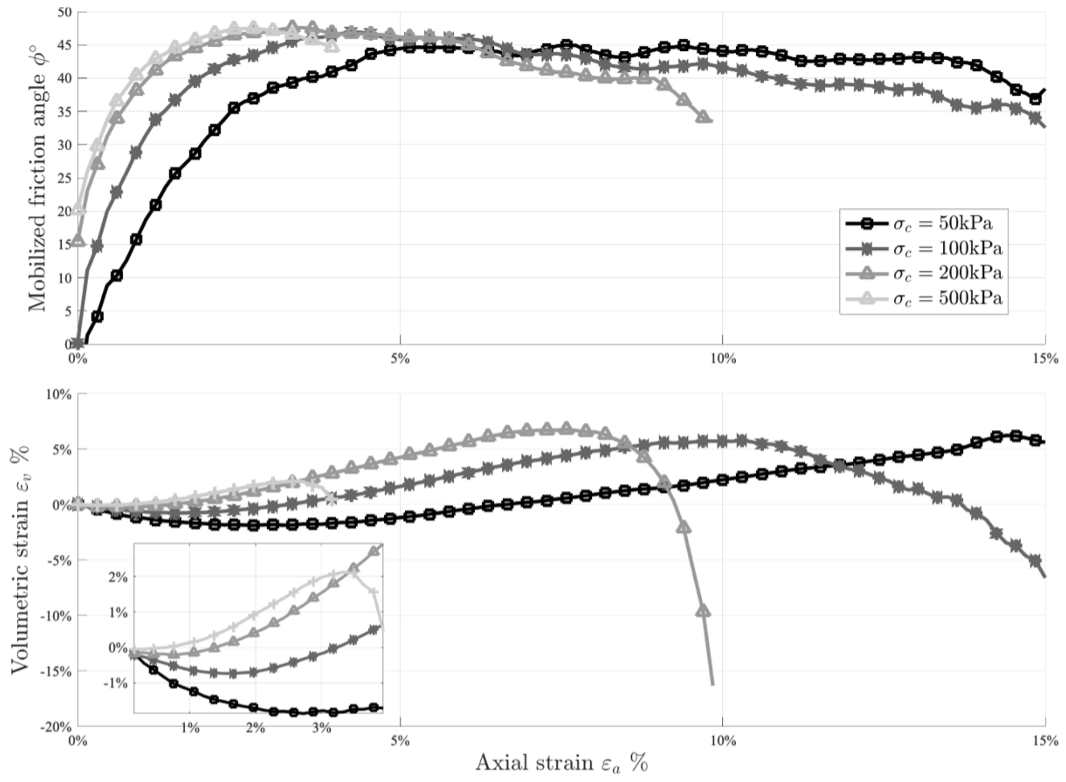

Figure 3.8: The influence of the confining pressure; the simulations with $\boldsymbol{\sigma}_c = 200$kPa and $\boldsymbol{\sigma}_c = 500$kPa ended earlier due to membrane distortion.

set to 0.75, while K_s/K_n has been set to 0.2, 0.4, 0.6, and 0.8. Figure 3.9 illustrates the effect of the normal to shear stiffness ratio on the mobilized friction angle ϕ and volumetric strain ε_v vs axial strain ε_a with a confining pressure of 100kPa. Elastic stiffness increases as the normal to shear stiffness ratio increases. Additionally, the peak strength increased somewhat, resulting in a minor rise in the angle of internal friction. On the other hand, the volumetric strain at initial dilation is nearly identical for all ratios, however it increases more distinctly with the ratio K_s/K_n as the specimens were sheared to approach the critical state. A high ratio of K_s/K_n represents high ability of tangential deformation resistance for grains in contact, and the effect is to increase the lateral deformation for the same axial deformation, generating more stable force network in the axial direction, which makes the sample stiffer. K_s is frequently smaller than K_n in DEM modeling to encourage numerical stability and prevent producing fictitiously large moments that could destabilize the simulation.

3.4.4 Grain Shape

The initial positions of the high-resolution avatars were employed to generate a spherical counterpart with equivalent radius to ensure the same initial void ratio as the LS reconstructed avatars (Figure 3.10). There may be some exaggerated or even unreasonable intersections for some spheres in the initial packing. Therefore, the packing is further subjected to more DEM cycles in the absence of gravitational force to reach a state of mechanical equilibrium. Throughout the course

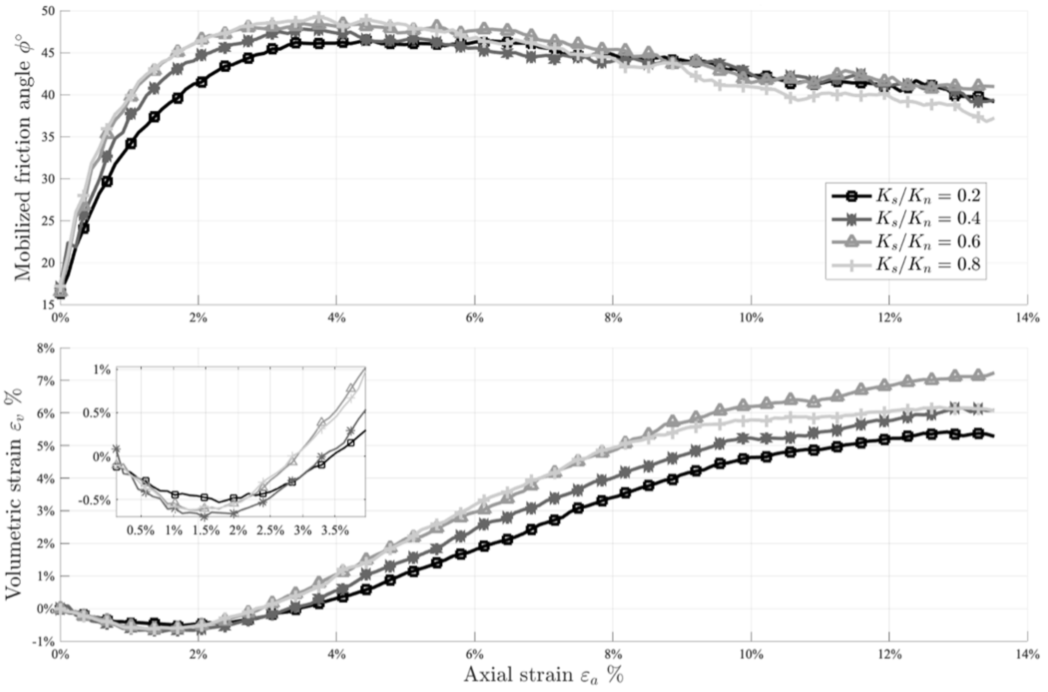

Figure 3.9: The influence of the normal to shear stiffness ratio, K_s/K_n.

of stabilization process, the velocities of the spheres were periodically reset to zero to avoid excessively large velocity due to possible significant penetration and collisions. As summarized by Mitchell and Soga (2005), even though the void ratios of an assemblage of uniform spheres are in the range of 0.35 to 0.91 depending on different packing patterns, the net void ratio may not be much different from the real granular assembly. On the one hand, smaller grains can occupy pore spaces between larger grains, while on the other hand, irregular grain shapes produce a tendency toward lower density and higher porosity. The void ratio of an assembly of reconstructed avatars is typically slightly larger than the true value as tiny grains are not captured during image processing and are also eliminated from simulation to ensure numerical stability.

The parameters for modeling idealized spherical particles were not calibrated and the rolling friction and bonding were not considered to compensate for the influence of grain shape and surface roughness in current work. The influence of internal friction coefficient is illustrated in Figure 3.11 to show that, regardless the parameter settings, spherical assemblies were not able to display dilatancy and peak mobilized friction angle which are due mainly to particle interlocking and surface roughness, even though the void ratio is kept the same as the assembly of avatars. When the internal friction coefficient μ decreases and eventually becomes zero, specimen deformation tends to be a more localized process, as interactions can only be transferred via direct normal contact, and no friction means that spheres have a higher degree of rotation and can find the optimal location to accommodate themselves without influencing others to a greater extent.

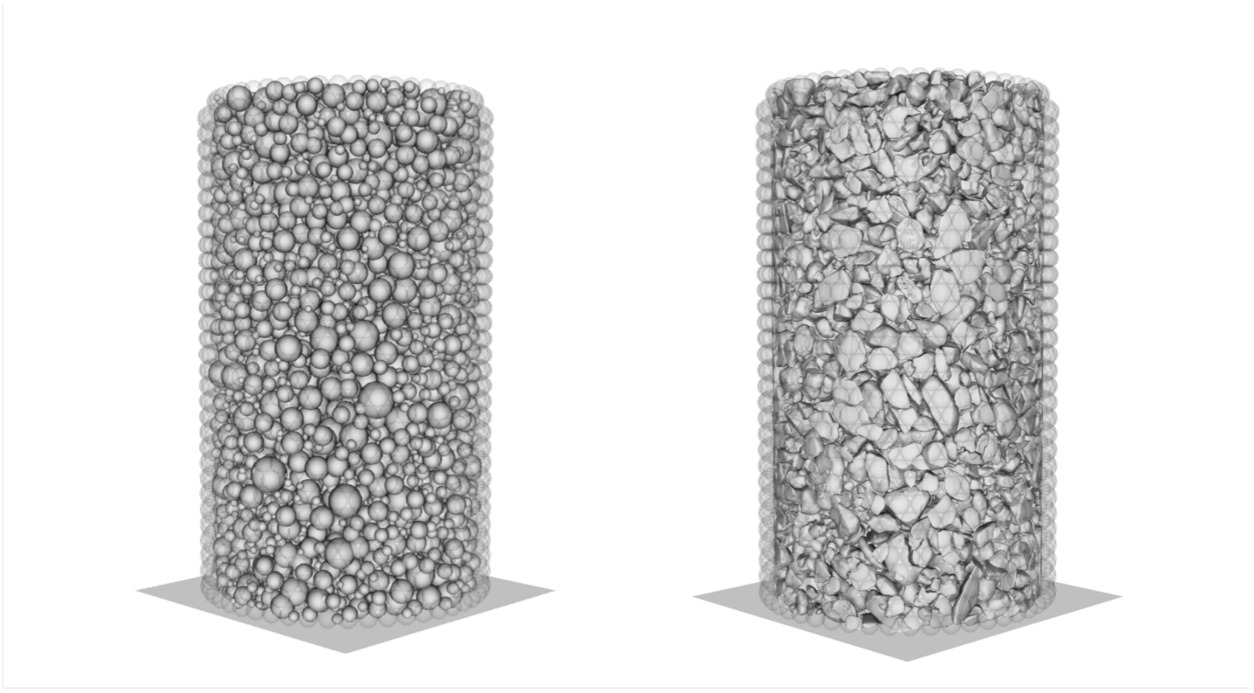

Figure 3.10: Left: An assembly of spherical particles. Right: An assembly of LS reconstructed avatars. Both specimens have identical void ratio.

3.4.5 Initial Void Ratio

The initial void ratio of a specimen is critical to the stability of the simulation because it can be unrealistically high or low and outside the normal range. For example, a numerical specimen that is reconstructed from low-resolution or polluted images has difficulty preserving small grains and capturing sharp contact edges. As a result, the numerical void ratio is unfaithfully large, individual grains are isolated and unlikely to be supported by their neighbors in order to resist confining pressure, causing the membrane to shrink significantly. To solve this issue, the assembly has to settle before performing a virtual compression test, but an erroneous choice of settlement force can result in an inhomogeneous and excessively dense specimen, which means grains may interpenetrate. In such case, very large forces are generated at the initial time step as the grains attempt to adjust their positions to avoid overlapping but are unable to do so without interfering with others.

This issue is not unique to LS-DEM; in conventional DEM, which represents grains with simpler geometries, stabilizing the specimen prior to simulation is a standard step to avoid numerical instability (Itasca, 2004). However, our issue is unique in that we want to preserve as much of the soil fabric as possible and to reconstruct avatars with the same initial positions and rotations as they were in-situ. This is one of the most significant challenges. To achieve the desired void ratio when using grain reconstruction from low resolution scans, we used a numerical pluviation scheme to control the drop height: the loose specimen was divided into several segments along its length, and each segment was activated at a time, subjected to a vertical force, and pluviated from the prescribed height. We encased the sample in a cylinder during pluviation to prevent excessive lateral

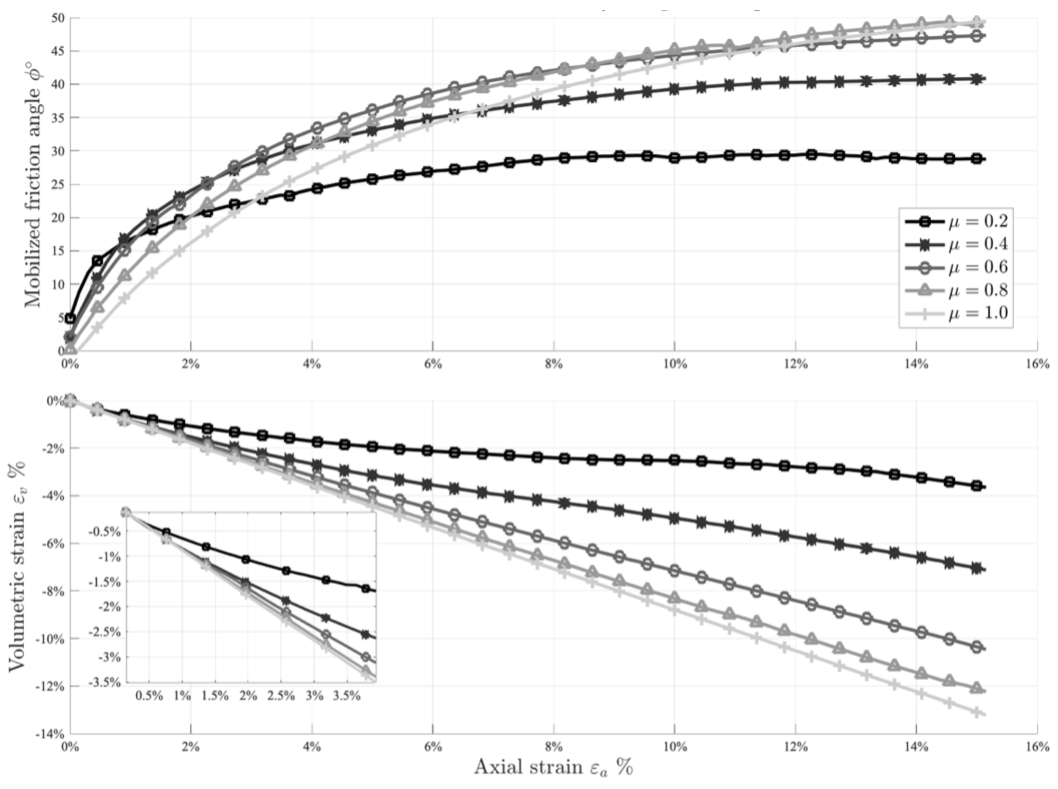

Figure 3.11: The influence of the internal friction coefficient of spherical grains on simulated stress-strain response.

movement and to avoid disturbing the original fabric. In addition the grain assembly was subjected to more gravity cycles without any external forces to reach a state of mechanical equilibrium. This procedure was stopped when the internal stresses stabilized at a value considerably smaller than the target confining pressure ($\boldsymbol{\sigma_c} = 100$kPa in current work). However, all of the above-mentioned numerical treatments alter the original fabric. As grains bounce against one another and migrate in random directions during pluviation to the desired void ratio, their initially ordered arrangement becomes random, defeating the goal of preserving as much of the original soil fabric as possible. In contrast, these steps are not required when the grains and fabric are constructed using high resolution XRCT images.

The influence of the initial void ratio and particle shape is illustrated in Figure 3.12. The fabric reconstructed from the high-resolution images could achieve almost the same void ratio as the real sample, although fair number of tiny avatars was omitted from the numerical specimen for the sake of numerical ease, as the total volume occupied by them was quite small. In contrast, the initial void ratio of low-resolution reconstruction was significantly greater than its true value for the numerical specimen reconstructed from the low-resolution scan and different void ratio can be manufactured via a numerical settlement or pluviation procedure. In addition to the high-resolution specimen and its spherical counterpart mentioned in previous discussion, we constructed another specimen containing $\sim 17,000$ avatars using medium-resolution images ($9 \sim 10\mu m$/voxel). The results show that the relatively dense specimens for both high- and medium-resolution LS avatars show strain hardening, a peak at around $\varepsilon_a = 2.0 \sim 3.5\%$, and gradual softening. Spherical grains and low resolution avatars, on the other hand, exhibit purely strain hardening and volume contraction until they achieve critical states. In comparison to perfectly spherical grains, irregularly shaped avatars have larger mobilized peak friction angles and are less likely to rotate than spherical grains. The peak mobilized friction angle increases from $\phi_p = 35°$ (spheres) up to $\phi_p = 49°$ (high resolution, high density avatars). In turn, the residual internal friction angle coincides for all simulations. The critical condition is always attained at a considerable axial strain around $\varepsilon_a = 15\%$, where the vertical normal stress remains constant with the specimen deforming at constant volume. This implies that grain shape is not significant in determining the global critical internal friction angle, as would be expected.

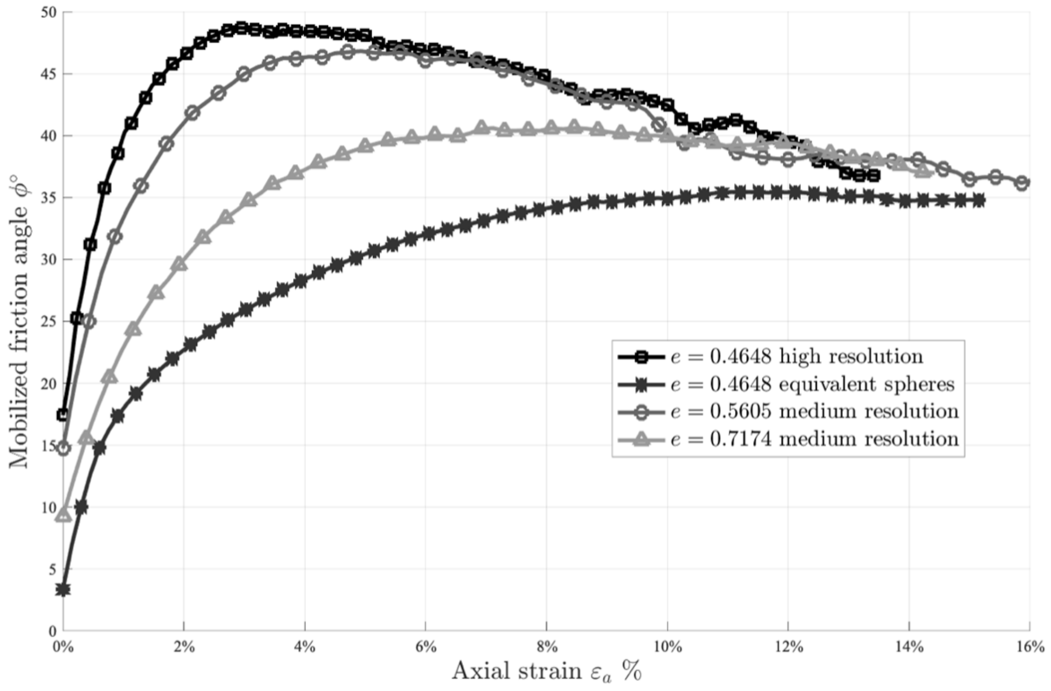

Figure 3.12: The effect of void ratio, considering both the grain shape and reconstruction resolution fidelity.

3.5 VIRTUAL TRIAXIAL COMPRESSION TEST

The model reconstructed from the high resolution images ($4.3\mu m$/voxel) contains $\sim 20,000$ avatars with ~ 750 nodes on average for a fully captured grain geometry. The model reconstructed from slightly lower resolution specimen ($9 \sim 10\mu m$/voxel) scans resulted in $\sim 70,000$ substantially less angular avatars with 250 discretized nodes per grain. We then performed numerical triaxial tests assuming a range of material friction coefficient from $\mu = 0.60$ to $\mu = 0.75$, which spans the range of typical values for quartz rich sand with the upper value corresponding to results obtained from the miniature triaxial tests on the undisturbed samples.

The low-resolution specimen was allowed to settle via a pluviation-like process under artificial gravity to re-establish contact between grains before isotropic consolidation. The void ratio was monitored throughout this process as accompanied by several other indicators such as average coordination number and macroscopic stress. Following settlement, a number of iterations were used to alleviate internal stress between grains and to correct excessive inter-grain overlap due to settlement. This step was continued until the internal stress was sufficiently smaller than the confining pressure, 100kpa considered herein. Not surprisingly, those preparation steps did not preserve the in-situ orientation, contact and structure of grains in the original sample. By contrast, the high-resolution specimen did not require any of those steps. Even though the grains were not yet in contact with neighbors, as indicated by the coordination number at the onset of simulation, due to the fact that avatars were reconstructed one at a time and left only small gaps between them, they were quickly brought together after the application of confining pressure and to a large extent recovered the soil fabric. The high-resolution specimen had initial void ratio $e = 0.60$, which is very close to the experimental data of 0.61. To make the simulations comparable, the low-resolution specimen was also constructed with same initial void ratio $e = 0.59$.

The models were then compressed following a strain-controlled scheme with a sufficiently low loading rate to maintain the quasi-static condition by keeping the inertia number I_{inertia} below 10^{-3} (note that the same loading rate as in the physical experiment is not possible because it would be computational intractable). We monitored several indicators to examine quasi-static condition, making sure that the total kinetic energy of avatars was smaller than 10^{-3} throughout the simulation, and the pressures on both platens was equal to the major principal stress of specimen.

3.5.1 Macroscopic Frictional Behavior

The friction coefficient between grains is a physical characteristic that is difficult to quantify and is strongly dependent on the mineralogy and chemical composition of the grains. Peak stress and mobilized friction angle increase as the inter-grain friction coefficient increases as would be expected and consistent with other studies, (e.g. Cui and O'Sullivan, 2005; Abriak and Caron, 2006; Hazzar et al., 2020). For the high-resolution sample, increasing friction coefficient from $\mu = 0.6$ to $\mu = 0.75$ resulted in a significant increase in peak deviatoric stress (549kPa to 652kPa) but a only a small increase in peak mobilized friction angle ($46.2°$ to $49.0°$). Our experimental results reveal that the critical state angles for naturally deposited sand and reconstituted sand are $38°$ and $32°$, respectively, which corresponds to friction coefficient 0.62 and 0.78. On the micro scale, the mineralogy parameters of quartz, the major constituent of sands, is reported as $35°$ by Mitchell and Soga (2005) and is slightly smaller than the computational value we calibrated. In addition, Mitchell and Soga (2005) collected experimental data suggesting that the macroscopic friction angle is nearly independent of the interparticle friction angle.

The evolution of the mobilized friction angle and volumetric strain is depicted in Figure 3.13 for low- and high-resolution numerical specimens. The models exhibit markedly different macroscopic behavior, despite having the same initial void ratio and being tested under identical conditions. The reconstituted, pluviated model shows strain hardening behavior reaching roughly identical mobilized friction angle between $29°$ and $30°$ regardless of the particle friction angle. In contrast, the model that reproduces the depositional fabric with high fidelity, shows a well-defined peak, with a mobilized friction angle ranging from $46°$ to $49°$ over the range of particle friction angles from $30°$ to $37°$. Most importantly, these results demonstrate the importance of the depositional fabric.

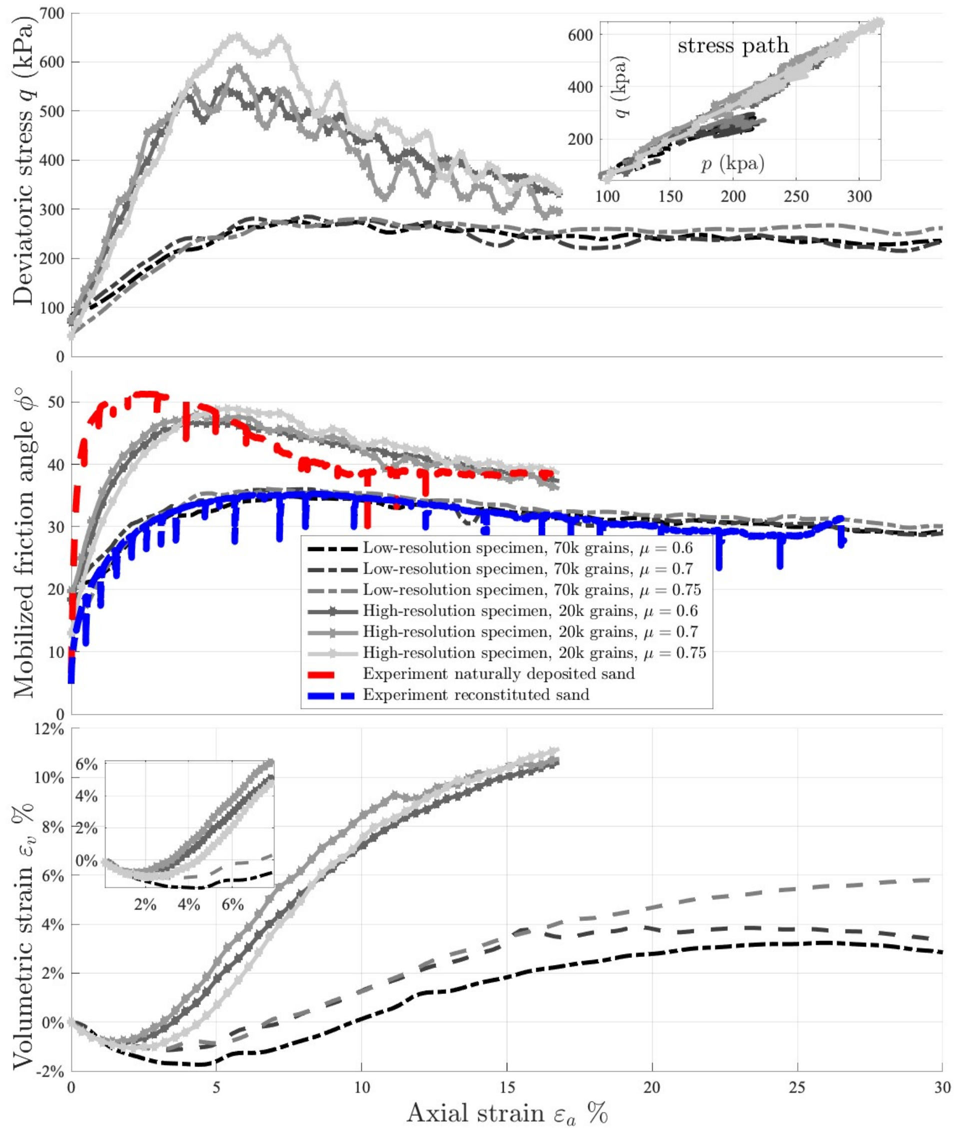

Figure 3.13: Comparison of deviatoric stress, mobilized friction angle and volumetric strain between low- and high-resolution numerical model specimens and experimental data.

3.5.2 Evolution of Mean Coordination Number

We also investigated the mechanical responses of these specimens from a microscopic point of view, with an emphasis on the coordination number which is defined as the number of contacts on a grain and recognized that there could be multiple contacts between each pair of grains. Coordination number is one of the most commonly used important parameters for characterizing granular fabric and is found to be sensitive to particle shape and increases with the increase complexity of grain shape. As displayed in Figure 3.14, the high-resolution specimens have a higher initial average coordination number, which makes rotation more difficult and results in the formation of more contacts. The coordination number of the low-resolution specimen continue to increase and reach a plateau, implying that newly formed and broken contacts balance each other at high strains and reaching a critical state. In contrast, the change in coordination number for high-resolution specimen is greater for grains due to a rougher surface and a more angular shape than. It also significantly affects the deformation in that a drop in mean coordination number corresponds to greater dilatancy. In our application, the evolution of coordination number is strongly correlated with the image resolution, and we found the mean coordination number of a high-resolution assembly to be almost twice as large as that of the low-resolution reconstruction. This effect to some extent reflects the mechanical interlocking enhanced by the greater angularity of the high-resolution grains.

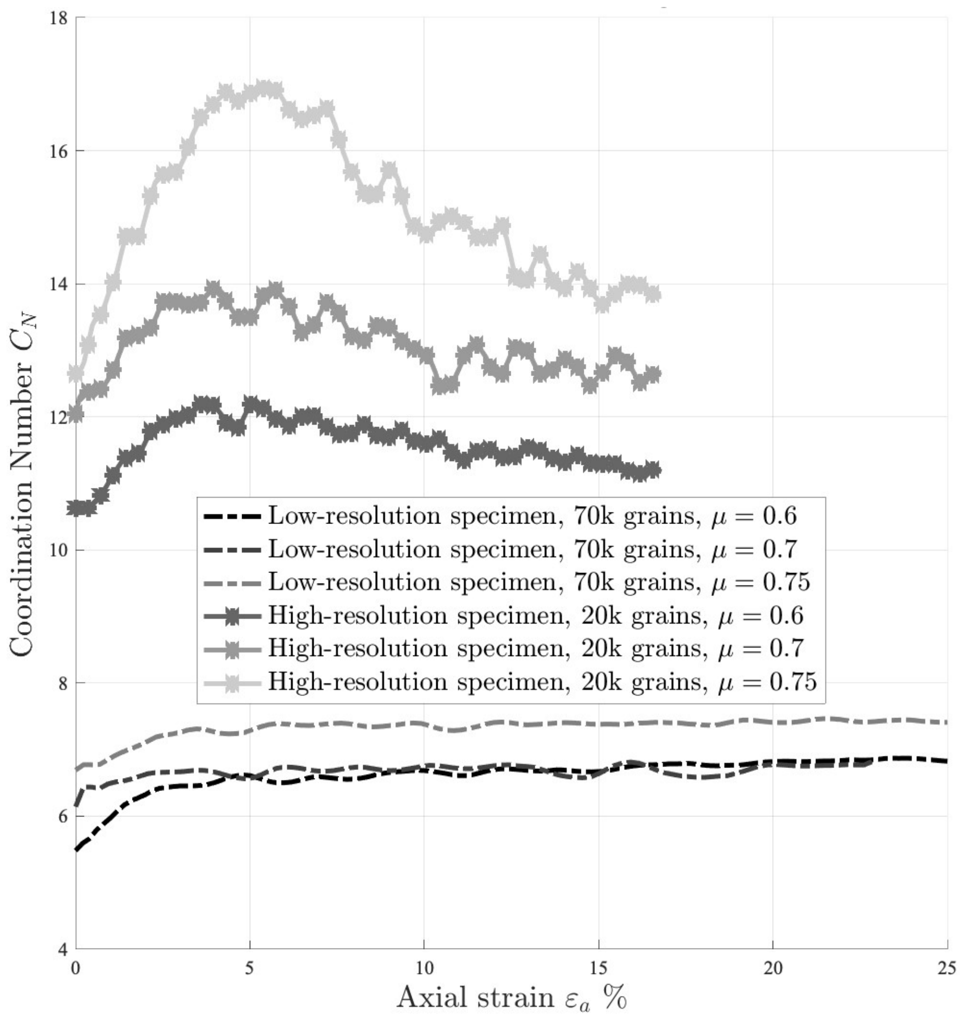

Figure 3.14: Evolution of coordination number during deviatoric loading.

3.5.3 Shear Band Evolution

Compared to conventional servo method using rigid boundary, one advantage of the flexible membrane is the development of a shear band. The shear band is characterized by large amount of localized shear deformation when granular material is deformed beyond its elastic limit, and it is an indicator of the instability of triaxial specimen, as the initial local inhomogeneity in the specimen gives rise to a concentration of deformation in its vicinity (Rudnicki and Rice, 1975; Rice, 1976). One common measurement of the shear band is its thickness in terms of multiples of the mean grain diameter D_{50}. This value varies with grain shape, specimen porosity, and mean stress (Guo, 2012) with a range of t_s values reported by different investigators: $10D_{50}$ (Kawamoto et al., 2018), $30 \sim 60D_{50}$ (Mollon et al., 2020), $9 \sim 14D_{50}$ (Amirrahmat et al., 2019), and $10 \sim 15D_{50}$ in our numerical results (Figure 3.16) and $8 \sim 12D_{50}$ in our experimental data.

The shear band inclination angle is determined by first locating grains with large rotations or local deviatoric strains greater than two standard deviations above the average, and then locating the plane with normal direction and height that minimizes the total sum of squared distances between previously identified grains and the plane. This is a least squares problem with outliers, and an orthogonal matching pursuit (OMP) algorithm is used to eliminate those outliers: grains that are far from the shear band but rotate significantly as a result of conforming to the motion of the loading platen. In terms of the angle of shear band inclination, the Roscoe's theory predicts it (Roscoe, 1970) as:

$$\theta_R = 45^\circ + \frac{\psi_{\max}}{2} \tag{3.16}$$

where $\psi_{\max}$ is the maximum dilatancy angle in the zone of deformation that achieves at peak stress. Our simulated data computes the friction angle between $52 \sim 54^\circ$, which is smaller than the Roscoe's solution (61°). This may be due to the non-associativity between stress and strain in granular materials.

The shear bands formed in the high-resolution specimen with different internal friction coefficient ($\mu = 0.60,\ 0.70,\ 0.75$) are quite similar. The development of the shear band begins as a diffused pattern of rotating grains prior to the formation of one particular shear band as displayed in Figure 3.15. Micro shear bands form during the hardening phase of loading $\varepsilon_a = 5\%$, typically slanting diagonally through the cylindrical specimen. As the simulation progresses into the softening phase at $\varepsilon_a = 10\%$, more micro shear bands are developed parallel or perpendicular to the direction of the final shear band, leading to a wider, zone of shearing. Continuing through the critical state, the final major shear band is fully developed and has a thickness of $t_s = 10 \sim 15D_{50}$, as displayed in Figure 3.16.

For the low-resolution specimen, the evolution of strain localization is displayed in Figure 3.17. A single shear band is still identified but the upper platen was tilted by a large amount to create a more pronounced strain localization. Figure 3.18 depicts the contact network distributions and microscale normal force chains for a low-resolution specimen with an axial shortening of between $\varepsilon_a = 0 \sim 30\%$. At the onset of compression, a fairly randomly oriented distribution of contact force networks is observed following isotropic consolidation. In the subsequent, deviatoric stages, stronger contact forces concentrate near the specimen's central axis and align vertically to resist the applied axial stress. Numerous strong force chains are identified that can withstand increased loading and contribute to the enhanced macro-scale stress-strain behavior. These chains buckle and restructure in subsequent stages as grains displace and rotate. When the specimen reaches the critical state, the forces in the shear band and out of the shear band appear to lose coaxiality, where force chains seem to change direction as they pass through the shear band. Several stronger

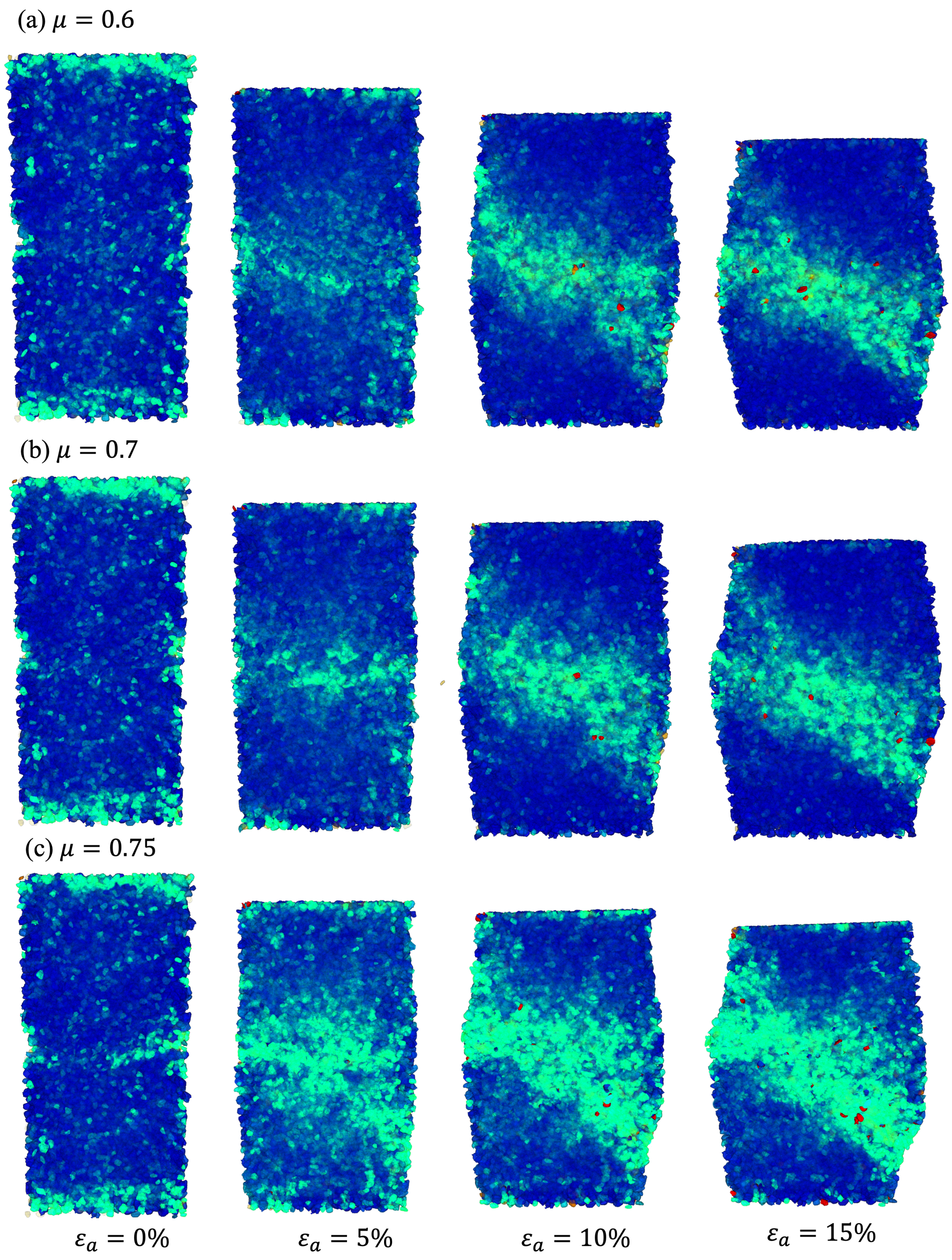

Figure 3.15: Single shear band pattern obtained from high-resolution specimen between $0 \sim 15\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$.

force chains remain but are concentrated near the center of the specimen center, and more lateral contacts are formed and stressed.

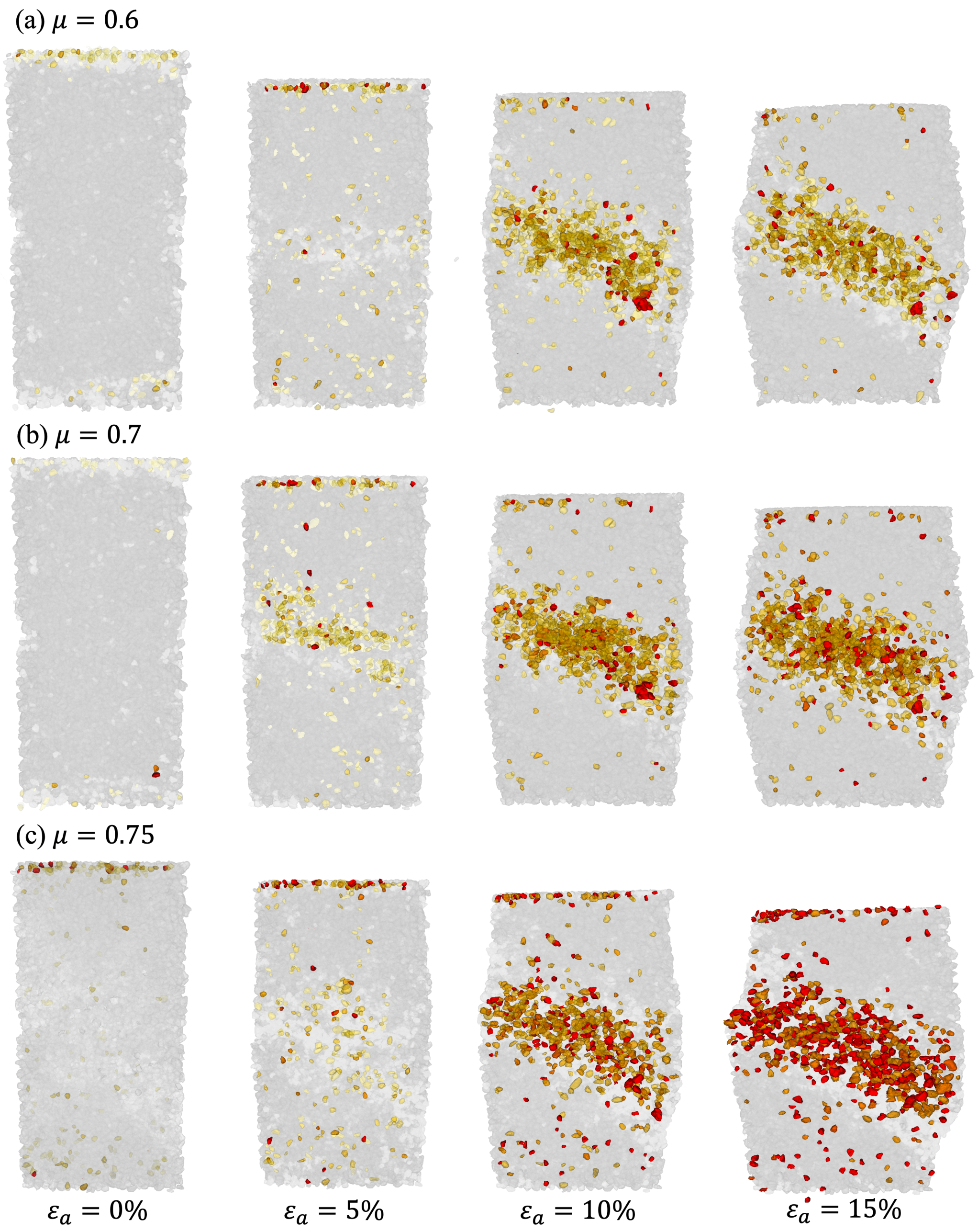

Figure 3.16: Grains with rotation greater than the mean rotation by two standard deviations (2σ) obtained from high-resolution specimen between $0 \sim 15\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$.

Interestingly, the strain localization pattern can change with different parameter settings for the low-resolution specimen and shear band simulations are just as prone to variability as actual experiments (Amirrahmat et al., 2019; Mollon et al., 2020; Rorato et al., 2021). The packing density, boundary conditions, and confining pressure all contribute significantly to mechanical interlocking, relative grain translation, and the tendency to develop a shear band. We further

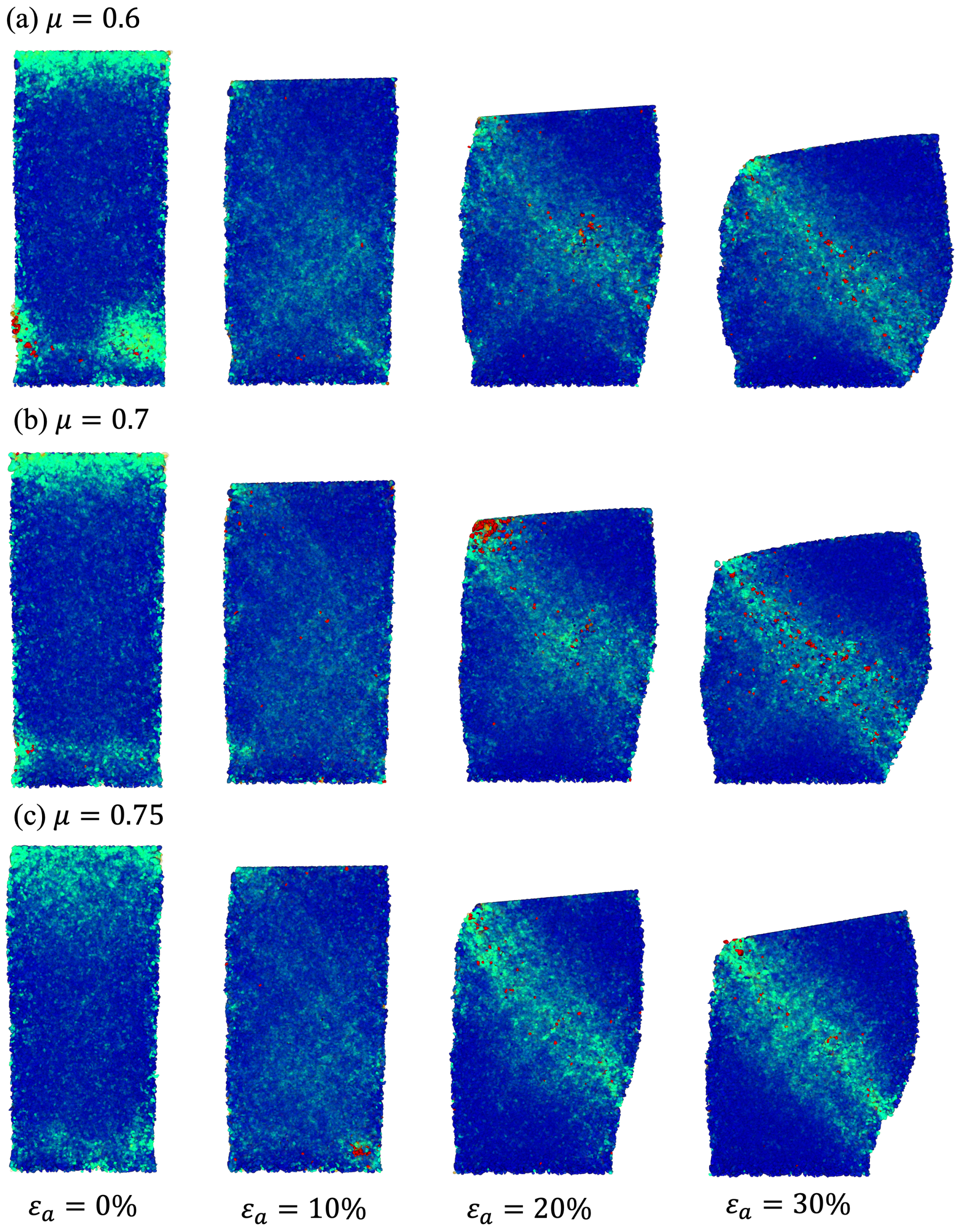

Figure 3.17: Shear band pattern obtained from low-resolution specimen between $0 \sim 30\%$ **axial shortening with internal friction coefficient, (a)** $\mu = 0.6$**; (b)** $\mu = 0.7$**; (c)** $\mu = 0.75$**.**

investigated different strain localization pattern, Figure 3.19 and Figure 3.20 show the shear pattern and the force chain for low-resolution specimen with the same confining pressure, but different membrane sphere-sphere normal stiffness and tangential stiffness to be $K_{nbb} = K_{sbb} = 100N/m$. In this case, randomly oriented micro shear bands coalesced into a centralized zone at central part

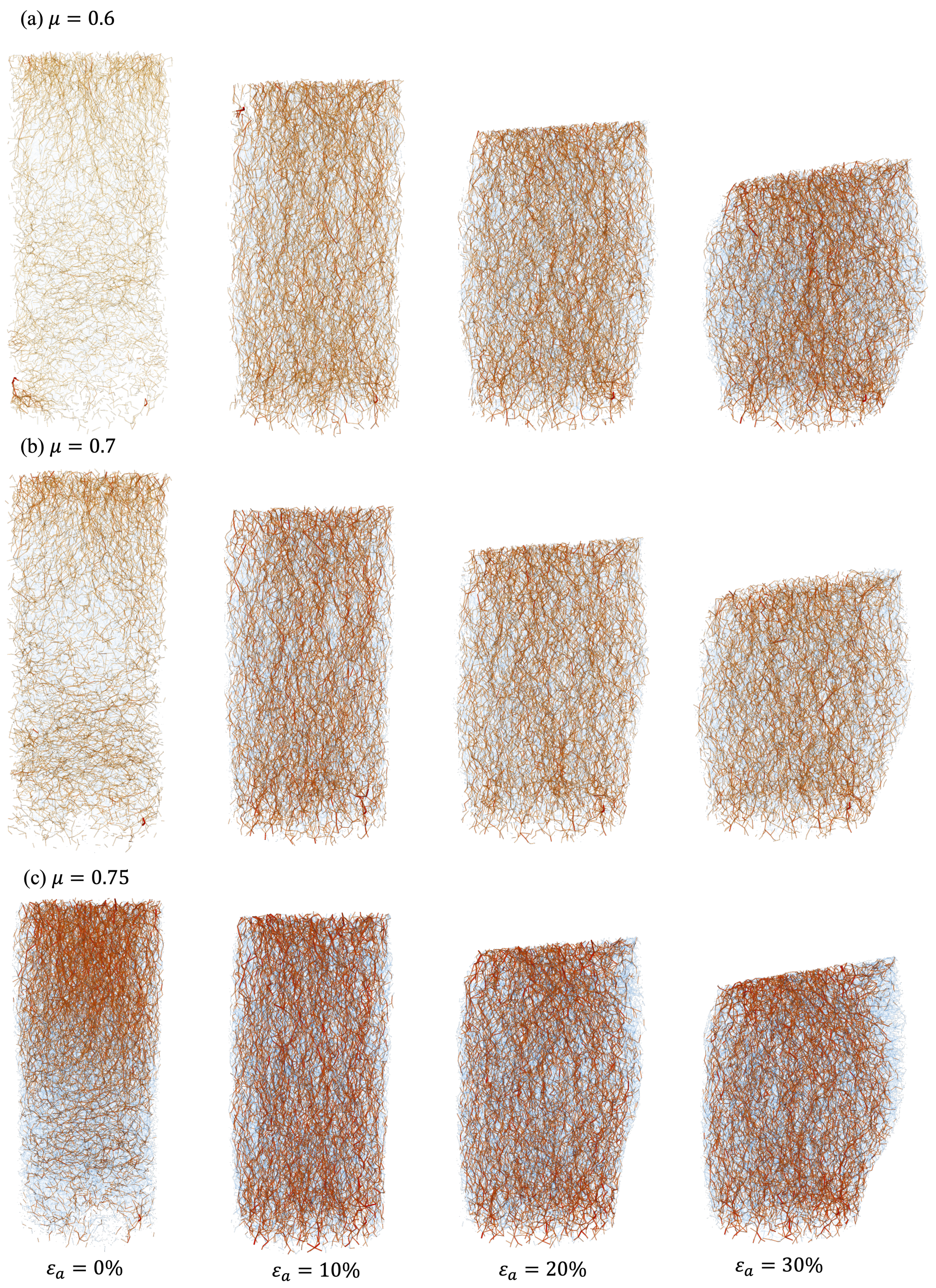

Figure 3.18: Force chain evolution obtained from low-resolution specimen between $0 \sim 30\%$ axial shortening with internal friction coefficient, (a) $\mu = 0.6$; (b) $\mu = 0.7$; (c) $\mu = 0.75$, forces change directions when they pass through the shear band.

of the specimen ($\varepsilon_a = 10\%$). As shearing progresses at $\varepsilon_a = 20\%$, a more complex network of micro shear bands forms in conjugate directions, this leads to a centralized cross-like strain localization pattern that dominates the internal microstructure and forces grains outward in the lateral direction, resulting the surface bulging of the specimen. When the stress plateau is reached, the cross-like pattern of localization persists and becomes more defined and stable; however, the thickness of two shear bands varies significantly, with one being easily identifiable and thinner and the other being more diffusive but thicker.

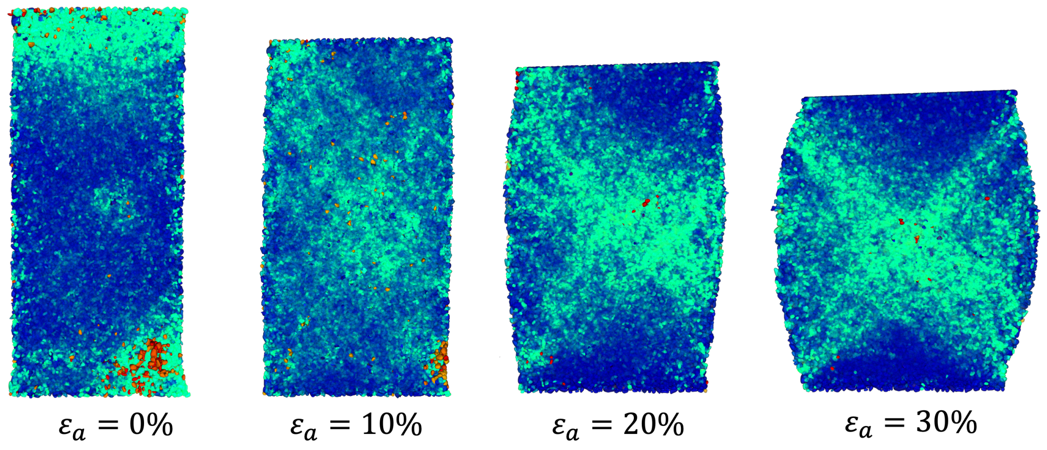

Figure 3.19: Shear band pattern obtained from low-resolution specimen between $0 \sim 30\%$ **axial shortening, with** $K_{nbb} = K_{sbb} = 100N/m$**.**

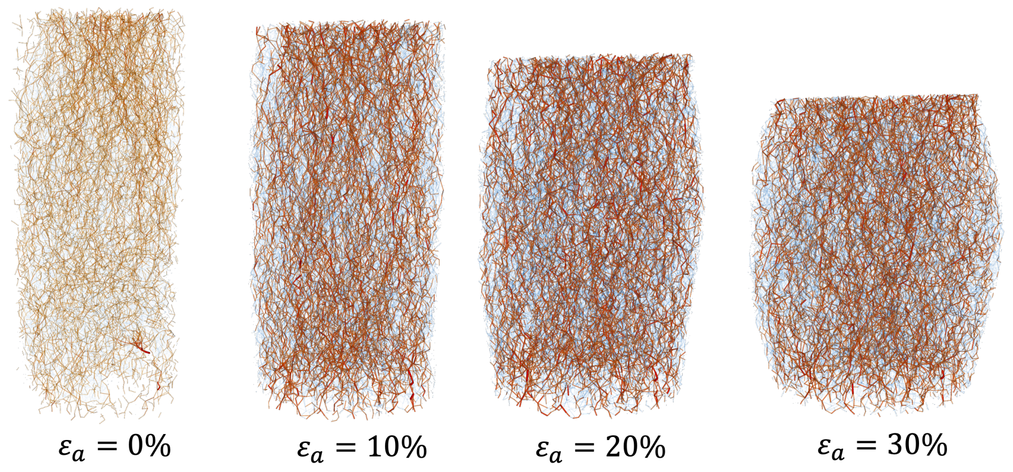

Figure 3.20: Force chain evolution in the low-resolution specimen model between $0 \sim 30\%$ **axial shortening, with** $K_{nbb} = K_{sbb} = 100N/m$**.**

3.5.4 Evolution of the Soil Fabric

We use the anisotropy factor, deviatoric fabric tensor and distribution density of directionality to investigate the evolution of soil fabric as the strain localization pattern develops. In this section, we take high-resolution specimen with $\mu = 0.75$ as an example and we define the representative orientation as the direction of the longest axis of a grain. The grains outside shear region for various shear stage are calculated with the shear band location at 16% axial shortening. The general pattern is that the soil fabric in both windows remains unchanged prior to strain localization and undergoes distinct evolution afterwards. Among numerous descriptors used to study the evolution of fabric, anisotropy is a scalar that describes the spread of the orientation. For an isotropically distributed discrete system, the anisotropy factor approaches zero. In our study, the anisotropy factor is found to be equal and non-zero ($a = 0.41 \sim 0.42$) both inside and outside the shear band at the onset of simulation, implying that the initial fabric is anisotropic due to the fact that the major principal orientations of grains are more likely to align perpendicular to the direction of deviatoric stress for a more stable state. By contrast, Wiebicke et al. (2020) reported that a rounded sample prepared by air pluviation with grains falling vertically exhibits little anisotropy in the initial fabric and a small initial inclination angle.

The anisotropy factor monotonically increases both inside and outside of the shear band, whereas the changes in inclinations are much faster inside the shear band whereby grains rearrange in the direction in which the shear band is forming at large strain. Another descriptor, the deviatoric tensor, which is the deviatoric component of the fabric tensor, is plotted in three dimensions in Figure 3.21 and Figure 3.22. Before the onset of strain localization, the soil fabric tensors are more spherical as also indicated by the anisotropy factor. While the anisotropy factor is nearly identical, the tensor shapes are slightly different. This demonstrates the need to characterize soil fabric with more than just a scalar anisotropy. With ongoing shear within the shear band, the deviatoric tensor evolves into a peanut shape. Outside the shear band, the deviatoric fabric exhibits a similar trend but is much less pronounced. A spherical histogram can be used to quantify and visualize the directionality of grain orientations. Between 0% and 16% axial shortening, Figure 3.23 and Figure 3.24 illustrate the evolution of the distribution of long axes of grains inside and outside the specimen's shear band. Within the shear band, the grains undergo a significantly greater rotations as loading progresses than those outside the shear band. As is the case with the anisotropy scalar and fabric tensor, the directionality does not change significantly prior to the peak ($\varepsilon_a = 5\%$). Following that, grains within the shear band unlock, rotate, and align in a well-defined direction of the shear band. While grains outside the shear band exhibit a similar trend, it is less pronounced and more localized in nature, with only comparably much small and random kinematic outside the shear band. This might be due to the fact the dilated zone is much larger than the area where shear and grain rotation actually occurs, suggesting that grains dilate and rotate relative to one another in a band wider than the shear region (Mollon et al., 2020). Additionally, grains outside the shear band act as individual entities that displace and rotate in response to the shear deformation.

3.5.5 Strain-Controlled Cyclic Loading Tests

We also explored LS-DEM simulations for compression tests on granular assembly subjected to a sequence of loading and unloading cycles, with strain-controlled stress relaxation (Figure 3.25). One objective of this simulation was to replicate the stress state of the specimen in the laboratory, which was stopped and imaged at various stations during loading. According to Kawamoto et al. (2018), no significant granular rearrangement occurs, confirming that interrupting axial loading for imaging has a negligible effect on experimental results. Because the amount of reversible

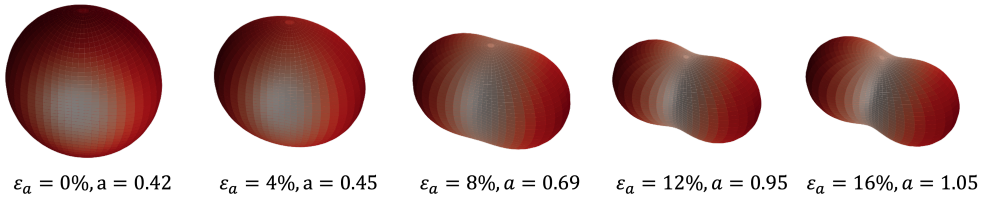
$\varepsilon_a = 0\%, a = 0.42$
$\varepsilon_a = 4\%, a = 0.45$
$\varepsilon_a = 8\%, a = 0.69$
$\varepsilon_a = 12\%, a = 0.95$
$\varepsilon_a = 16\%, a = 1.05$

Figure 3.21: Surface plots of the distribution density of deviatoric fabric tensor and the anisotropy factor of grain inside the shear bands.

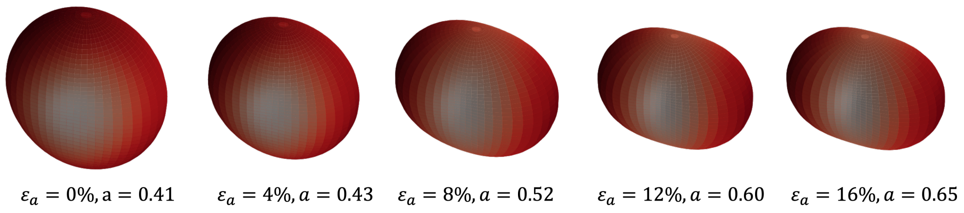
$\varepsilon_a = 0\%, a = 0.41$
$\varepsilon_a = 4\%, a = 0.43$
$\varepsilon_a = 8\%, a = 0.52$
$\varepsilon_a = 12\%, a = 0.60$
$\varepsilon_a = 16\%, a = 0.65$

Figure 3.22: Surface plots of the distribution density of deviatoric fabric tensor and the anisotropy factor of grain outside the shear bands.

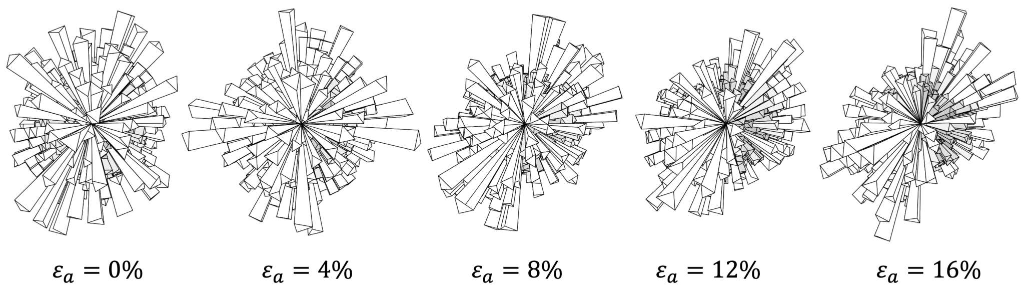

Figure 3.23: Spherical histogram of grain inside the shear bands.

strain is small and the loading stress is sufficiently high (to achieve a mobilized friction angle greater than $45°$, the axial loading should be 6 times greater than the confining pressure), the mechanical behavior of the packing tends to be elastic. When the loading stresses are low, as they are during the initial stage of compression, the assembly's internal structural evolution is likely to be dominated by irreversible porosity reduction caused by pore collapse and grain re-arrangement, and as a result, the assembly is unlikely to fully restore its volume. When loading stresses become sufficiently high but not yet close to the threshold for grain breakage, the room for residual porosity to diminish further is significantly reduced, resulting an elastic response.

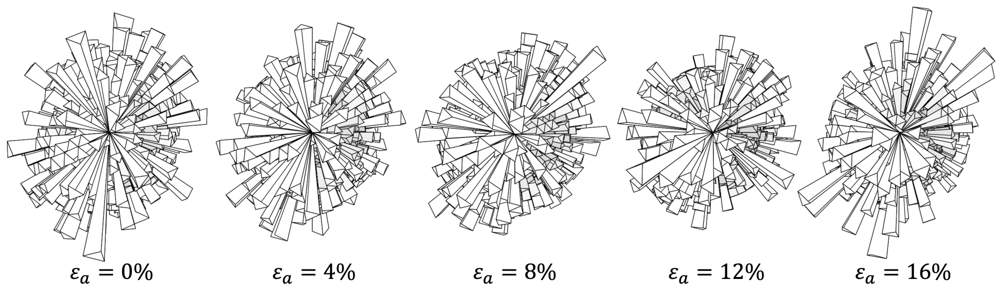

Figure 3.24: Spherical histogram of grain outside the shear bands.

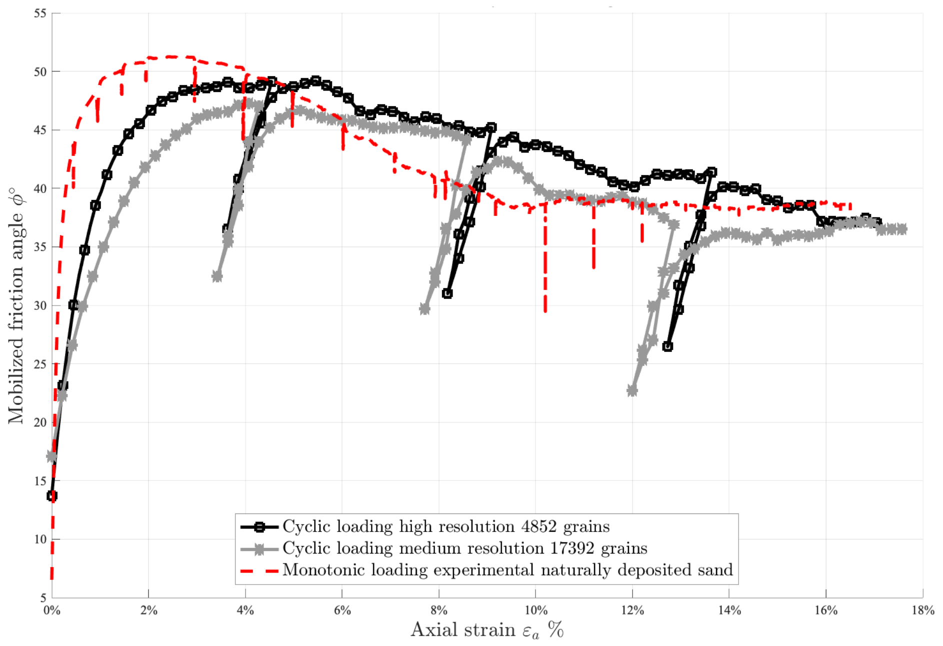

Figure 3.25: Mobilized friction angle in two numerical specimens under strain-controlled loading and unloading cycles.

3.5.6 Computational Effort

As previously discussed, the computational cost is primarily determined by the number of nodes used to represent the grain shape, and the increasingly accurate capture of grain shape comes with substantially fewer grains being reconstructed from a fixed-size window of XRCT scans. The parallel code is sufficiently efficient in that it can solve problems of comparable size in a comparable amount of time as the original code by Kawamoto et al. (2016), while consuming less than 5% of the computing resources. In the current study, we simulated an assembly of $\sim 70,000$ low-resolution grains with 250 nodes per grain in the same amount of time as a high-resolution assembly of $\sim 20,000$ grains with 750 nodes per grain. This demonstrates that the actual factor controlling the execution speed of the code is proportional to the number of nodes on the grain surface. Thus, we found simulating an even higher resolution assembly containing only $1,600$ grains with more than $2,500$ nodes per grain is numerically an almost intractable task.

3.6 CONCLUSIONS

LS-DEM simulations of naturally deposited uncemented sands subjected triaxial compression are presented herein. A linear elastic model with Coulomb friction yield criterion was calibrated, verified, and used to simulate the frictional response of an assembly of LS avatars, which are one-to-one direct mappings from XRCT scans and capable of preserving the grain morphology and surface roughness with high fidelity. A significant conclusion is that the primary source of the apparent frictional resistance is the mechanical interlocking between irregularly shaped grains; thus, even the simplest contact model is capable of reproducing both micro- and macromechanical responses consistent with experimental data, provided that the microstructure of the grain is captured with sufficient fidelity. Flexible membrane simulations with a pivoting loading platen were found to better predict the stress-strain and volumetric response, and the onset and growth of strain localization. Consequently, the results closely match experimentally observed relationships between deviatoric stress and mobilized friction angle with axial shortening for uncemented samples.

LS-DEM also provides an avenue for quantitative study of the kinematic behavior of real-world grain shapes and enables the study of strain localization down to the level of a few grains, which were previously rarely studied in conventional DEM modeling with simplified geometries. Monitoring grain rotations during virtual tests aids in identifying high-shearing zones and provides insight into the failure mode of granular media, e.g., barreling failure or shear banding. Low- and high-resolution specimens with the same initial void ratio and confining stress exhibit distinct strain localization patterns, with complex hourglass-like patterns for the low-resolution reconstruction and a well-defined shear localization for the high-resolution reconstruction. In comparison to macroscopic frictional responses, grain-scale responses in shear band simulations are subject to considerable uncertainty.

The coordination number, anisotropy factor, directionality, and a second-order deviatoric fabric tensor are used to analyze the evolution of the soil fabric. The high-resolution specimen has a higher initial coordination number and the grains are more difficult to rotate, it also results in increased frictional resistance and significant volumetric dilation. Contacts form and break dynamically in low-resolution specimens and balance each other at high shear after reaching the critical state. Following the onset of strain localization, the soil fabric inside and outside the shear band evolves in distinct ways, with anisotropy increasing and orientations aligning with the major shear band direction inside the shear band. Both processes are predicted by micromechanical analysis of grain kinematics. The micro analysis reveal that the major kinematic changes, particularly grain rotations, occur within the shear band, while much smaller and random kinematic changes occur outside the shear band.

While these findings are significant and necessary for a complete micromechanical description, they must be interpreted cautiously. To begin, contact properties are still determined in a heuristic manner, and they vary according to the type of sand. This necessitates the replication and extension of experimental studies under varying initial conditions in order to validate and complement these findings. Second, the predictive power of LS-DEM should be investigated under a variety of loading conditions, as the triaxial compression test has been almost exclusively used to study grain microstructure in the current study. As a trial, our study has been extended to cyclic loading with strain relaxation, with expected results, this piques interest in more sophisticated loading conditions. Finally, and perhaps most importantly, the quality of the images used to generate the LS correspondences is critical to the success of the simulation of the micro- and macro-mechanical response of a granular assembly.

4 Impulse-Based LS-DEM for Dynamic Problems

4.1 INTRODUCTION

Typical DEM codes in general use in research and in the industry use penalty-based approach to resolve contacts between the discrete bodies, which are then converted to forces, accelerations, and finally displacements. This approach has varying amount of complexity in generating a contact interaction force based on separation distance between particles in contact (Zohdi, 2017). However, they are computationally demanding, if not prohibitively expensive, for large-scale DEM simulations with complex-shaped grains or particles, as already discussed. While recent research efforts have been made to accelerate these DEM codes using high-performance computing clusters or graphics processing units (GPUs) (Owen and Feng, 2001; Zheng et al., 2012; Yan and Regueiro, 2018b,a), such resources are not generally or broadly accessible.

Alternatively, an impulse-based technique based on Newton's impact law is significantly more computationally efficient since it directly computes the velocity increment of the modeled objects. It was originally designed for the computer graphics industry (Mirtich and Canny, 1995; Chang and Colgate, 1997; Bender, 2007) and prioritizes code speed, stability, and physical plausibility over simulation fidelity. Recent work (Tang et al., 2014; Lee and Hashash, 2015; Asai et al., 2021; Li et al., 2021) demonstrates that, in addition to much improved code speed and numerical stability, the impulse-based technique also produces results with comparable accuracy to the corresponding penalty-based DEM simulations. This makes the impulse-based approach intriguing, since it is capable of addressing very large dynamic problems.

Herein we summarize the framework of impulse-based rigid body dynamics, which is fundamentally a process of kinetic energy conversion to elastic energy. Rather than providing a broad introduction to the subject, we focus on the numerical aspects of the impulse-based approaches and the possibility of parallelizing impulse-based LS-DEM using the domain decomposition approach. We show that in comparison to parallel implementation for penalty-based LS-DEM, impulse-based DEM saves considerable effort in data communication, but requires additional handling of multi-body assemblies in direct contact. We then introduce a modified Energy Tracking Method (ETM, Tang et al., 2014) in order to resolve collisions within the domain with a wide variety of grain shapes and sizes and to add deformable structures into rigid body dynamics. Finally, we illustrate the performance and applicability of the impulse-based LS-DEM for rock fall and rock avalanche simulations.

4.2 FORMULATION OF IMPULSE-BASED RIGID BODY DYNAMICS

A negative relative velocity between two objects at the time of a collision suggests that they are about to penetrate. To avoid penetration between rigid bodies, the negative relative velocity is increased till it reaches zero by applying a sequence of impulses in the normal direction of contact. The compression phase converts kinetic energy to elastic energy, which is then stored at the contact points. Following that, previously held elastic energy is released to restore relative normal velocity, ejecting two bodies from one another and concluding the collision process. As it is almost impossible to predict the microstructure of materials during collisions in practice, we adopt Stronge's hypothesis (Stronge, 2018): the amount of energy absorbed and released may vary according to the restitution coefficient. For example, for a perfectly elastic collision with no energy loss. Stronge's hypothesis states that energy is dissipated as follows:

$$W_{release} = \epsilon^2 W_{absorbed} \tag{4.1}$$

Where $W_{release}$ and $W_{absorbed}$ denote the energy released and absorbed during release and compression phases, respectively. $\epsilon (0 \leq \epsilon \leq 1)$ denotes the coefficient of restitution that determines the elasticity of the contact. Without loss of generality, we formalize a single collision process by considering a body b_A collides with another b_B due to the i-th impulse $\mathbf{P}^i$, which effectively changes their linear velocities ($\mathbf{V}_A$ and $\mathbf{V}_B$) and angular velocities ($\boldsymbol{\omega}_A$ and $\boldsymbol{\omega}_B$), as shown in Figure 4.1 via:

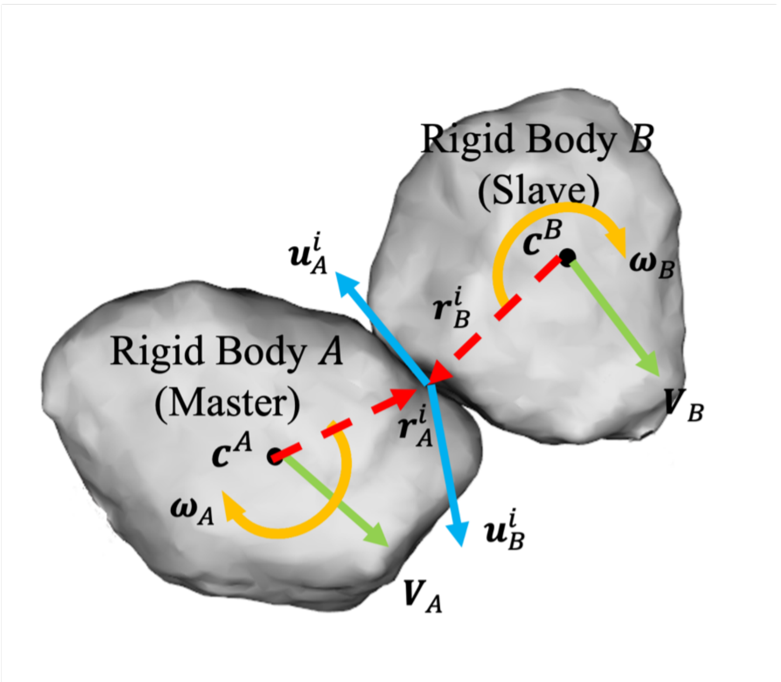

Figure 4.1: Collision between bodies b_A and b_B: V_A and V_B are linear velocities of the center of mass, ω_A and ω_B are angular velocities, r_A^i and r_B^i are the vectors from the center of mass of bodies to the contact point where the i-th impulse P^i applied, u_A^i and u_B^i are the velocities at the contact points, and n^i is the contact normal.

$$\mathbf{P}_A^i = M_A \Delta \mathbf{V}_A \tag{4.2}$$

$$\mathbf{P}_B^i = M_B \Delta \mathbf{V}_B \tag{4.3}$$

$$\mathbf{r}_A^i \times \mathbf{P}_A^i = \mathbf{I}_A \Delta \boldsymbol{\omega}_A \tag{4.4}$$

$$\mathbf{r}_B^i \times \mathbf{P}_B^i = \mathbf{I}_B \Delta \boldsymbol{\omega}_B \tag{4.5}$$

$$\mathbf{P}_A^i = -\mathbf{P}_B^i \tag{4.6}$$

Where M_A and M_B are the masses, I_A and I_B are the inertial tensors, $\mathbf{r}_A^i$ and $\mathbf{r}_B^i$ are the relative positions from the center of gravity of b_A or b_B to the contact point at which the i-th impulse was applied. The velocities $\mathbf{u}_A^i$ and $\mathbf{u}_B^i$ of contact points on b_A and b_B in global frame are:

$$\mathbf{u}_A^i = \mathbf{V}_A + \boldsymbol{\omega}_A \times \mathbf{r}_A^i \tag{4.7}$$

Therefore, the velocity changes at a point of contact between body b_A and body b_B due to the i-th impulse respectively is given below. Here, we suppose that the collision occurs and resolves quickly enough that the contact point i remains stationary.

$$\Delta \mathbf{u}_A^i = \Delta \mathbf{V}_A + \Delta \boldsymbol{\omega}_A \times \mathbf{r}_A^i = \frac{1}{M_A}\mathbf{P}_A^i + \mathbf{I}_A^{-1}\mathbf{r}_A^i \times \mathbf{P}_A^i \times \mathbf{r}_A^i = \left(\frac{1}{M_A} - \tilde{\mathbf{r}}_A^i \mathbf{I}_A^{-1} \tilde{\mathbf{r}}_A^i\right)\mathbf{P}_A^i \tag{4.8}$$

$$\Delta \mathbf{u}_B^i = \Delta \mathbf{V}_B + \Delta \boldsymbol{\omega}_B \times \mathbf{r}_B^i = \frac{1}{M_B}\mathbf{P}_B^i + \mathbf{I}_B^{-1}\mathbf{r}_B^i \times \mathbf{P}_B^i \times \mathbf{r}_B^i = \left(\frac{1}{M_B} - \tilde{\mathbf{r}}_B^i \mathbf{I}_B^{-1} \tilde{\mathbf{r}}_B^i\right)\mathbf{P}_B^i \tag{4.9}$$

The second equality converts cross product between vectors into equivalent matrix-vector multiplication. In the vector space $\mathcal{R}^3$, the cross produce $(\times)$ is an operator taking two vectors to a third vector:

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix} = \tilde{\mathbf{a}}\mathbf{b} \tag{4.10}$$

Where $\tilde{\mathbf{a}}$ is a 3×3 skew-symmetric matrix. In the collision formula above:

$$\tilde{\mathbf{r}} = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \tag{4.11}$$

Above formulas imply that if we know the change of relative velocity at the contact points, the resulted impulse can be computed via:

$$\Delta\left(\mathbf{u}_A^i - \mathbf{u}_B^i\right) = \Delta\mathbf{u}_A^i - \Delta\mathbf{u}_B^i = \left[\left(\frac{1}{M_A} + \frac{1}{M_B}\right)\mathbf{I} - \left(\tilde{\mathbf{r}}_A^i \mathbf{I}_A^{-1} \tilde{\mathbf{r}}_A^i + \tilde{\mathbf{r}}_B^i \mathbf{I}_B^{-1} \tilde{\mathbf{r}}_B^i\right)\right]\mathbf{P}_A^i \tag{4.12}$$

$$\mathbf{K} = \left(\frac{1}{M_A} + \frac{1}{M_B}\right)\mathbf{I} - \left(\tilde{\mathbf{r}}_A^i \mathbf{I}_A^{-1} \tilde{\mathbf{r}}_A^i + \tilde{\mathbf{r}}_B^i \mathbf{I}_B^{-1} \tilde{\mathbf{r}}_B^i\right) \tag{4.13}$$

Where $\mathbf{P}_A^i$ is the i-th impulse exerted on the body b_A, and $\mathbf{K}$ is termed as the collision matrix and it is non-singular, symmetric, positive definite and defined in global frame for a given collision (Mirtich, 1996). Collision contact occurs within a very short time interval for a rigid body, implying that displacement and change in the contact area are insignificant. This indicates that the collision matrix $\mathbf{K}$ remains constant throughout the collision, and that once the change in relative velocity is known, the collision matrix is used to compute the impulses:

$$\mathbf{P}^i = \mathbf{K}^{-1}\Delta\mathbf{u}^i \tag{4.14}$$

Where $\mathbf{P}^i = \mathbf{P}_A^i = -\mathbf{P}_B^i$ is the i-th impulse, $\mathbf{u}^i = \mathbf{u}_A^i - \mathbf{u}_B^i$ is the relative velocities at the contact points where the i-th impulse is applied, and $\Delta\mathbf{u}^i$ is the change of relative velocities. This formula remains true if the friction forces are ignored or if the friction forces are constant. As a result, if there is no slip between the bodies, only the normal component of the change in relative velocity $\Delta\mathbf{u}_n^i$ is non-zero, because it is the relative normal velocity at the contact locations that drives the bodies to collide. Thus, the preceding formula is reduced to:

$$\mathbf{n}^i \cdot \mathbf{K}\mathbf{P}^i = \mathbf{n}^i \cdot \Delta\mathbf{u}^i = \Delta\mathbf{u}_n^i \tag{4.15}$$

$$\left|\mathbf{P}_n^i\right| = \mathbf{n}^i \cdot \mathbf{P}^i \mathbf{n}^i \tag{4.16}$$

$$\mathbf{P}_t^i = \mathbf{P}^i - \mathbf{P}_n^i \tag{4.17}$$

If sliding occurs, apart from ensuring that relative normal velocity changes are consistent with the prescribed value, the impulse must be recalculated to meet Coulomb's friction requirement, which limits the magnitude of tangential impulse:

$$\left|\mathbf{P}_t^i\right| = \mu\left|\mathbf{P}_n^i\right| \tag{4.18}$$

$$\mathbf{P}^i = \left|\mathbf{P}_n^i\right| \mathbf{n}^i + \mu \left|\mathbf{P}_n^i\right| \mathbf{t}^i \tag{4.19}$$

$$\left|\mathbf{P}_n^i\right| = \frac{\Delta \mathbf{u}_n^i}{\mathbf{n}^i \cdot \left(\mathbf{K}\mathbf{n}^i + \mu \mathbf{t}^i\right)} \tag{4.20}$$

Where $\mathbf{n}^i$, $\mathbf{t}^i$ are the contact normal and tangential direction at the i-th collision, and $\mathbf{P}_n^i$ and $\mathbf{P}_t^i$ are the normal and tangential components of applied impulse. This concludes the compression phase where the corresponding impulse is generated based on the change of relative velocity and the kinetic energy is stored in the form of elastic energy at the contact.

The next step is to dissipate the elastic energy and invert the sign of the relative normal velocity to separate the bodies. This is the inverse of the compression phase: a predefined amount of elastic energy is dissipated to increase the relative normal velocity via a series of impulses. Mirtich (1996) proved that if the relative contact velocity $\mathbf{u}^i$ proceeds from $\mathbf{u}_0^i$ to $\mathbf{u}_f^i$ during a collision and over some arbitrary path, the total work done by the collision impulse $\mathbf{P}^i$ is independent of the path taken and is given below, which enables the change in relative velocities to be calculated from the released elastic energy.

$$\Delta W^i = \frac{1}{2}\left(\mathbf{u}_f^i + \mathbf{u}_0^i\right)^T \mathbf{K}^{-1}\left(\mathbf{u}_f^i - \mathbf{u}_0^i\right) = \frac{1}{2}\left(\mathbf{u}_f^i + \mathbf{u}_0^i\right)^T \mathbf{P}^i \tag{4.21}$$

Given that rigid body motion is better defined in terms of its principal or local frame, it is more convenient to address collisions in the local coordinate system via the rotation matrix $\mathbf{R}$, which defines the transformation from body to global coordinates. In our implementation, rotations are handled numerically via a singularity-free quaternion technique.

$$\Delta W^i = \frac{1}{2}\left(\mathbf{u}_f^i + \mathbf{u}_0^i\right)^T \mathbf{P}^i = \frac{1}{2}\left(\mathbf{R}\tilde{\mathbf{u}}_f^i + \mathbf{R}\tilde{\mathbf{u}}_0^i\right)^T \mathbf{R}\tilde{\mathbf{P}}^i = \frac{1}{2}\left(\tilde{\mathbf{u}}_f^i + \tilde{\mathbf{u}}_0^i\right)^T \tilde{\mathbf{P}}^i \tag{4.22}$$

Where $\tilde{\mathbf{u}}_0$, $\tilde{\mathbf{u}}_f$ and $\tilde{\mathbf{P}}^i$ denote initial relative velocity, final relative velocity, and applied impulse, all with respect to the body frame. During a collision, the work done by the impulse can be decomposed into the work done by the impulse in the normal direction ΔW_n^i and the work done by the impulse in the tangential direction ΔW_t^i.

$$\Delta W^i = \Delta W_n^i + \Delta W_t^i \tag{4.23}$$

$$\Delta W_n^i = \frac{1}{2}\left(2\tilde{u}_n^i + \Delta\tilde{u}_n^i\right)\tilde{P}_n^i \tag{4.24}$$

$$\Delta W_t^i = \frac{1}{2}\left(2\tilde{u}_t^i + \Delta\tilde{u}_t^i\right)\tilde{P}_t^i \tag{4.25}$$

Where $\tilde{u}_n^i$, $\tilde{u}_t^i$ and $\tilde{P}_n^i$, $\tilde{P}_t^i$ are the normal and tangential component of relative velocities and impulses at contact points defined in the body frame. It is more usual to depict a plane perpendicular to the contact normal using two orthogonal vectors, which is also compatible with the rotation operation using three orthogonal vectors. We decided not to differentiate the two tangential directions here since we were only concerned with the normal components: tangential velocities would remain unchanged because they did not break the impenetrability requirements between rigid entities. Due to the fact that work performed in the different directions is independent, the change in relative normal velocity has no effect on the amount of work performed in the normal direction by the impulse, which later inverts the relative normal velocity by releasing the energy absorbed during the compression phase.

The change of relative velocity $\Delta\mathbf{u}^i$ and the corresponding work done ΔW^i by the impulse implies that there should be a way to compute $\Delta\mathbf{u}^i$ (ΔW^i) if the collision work ΔW^i ($\Delta\mathbf{u}^i$) is known. Indeed, changing simply the relative normal velocity results in the generation of tangential impulses and thus the associated work. However, because the work performed in normal directions is decoupled, the change in relative normal velocity $\Delta\mathbf{u}_n^i$ can be retrieved provided that we get access the normal impulse $\tilde{P}_n^i$ and the absorbed/released work caused by $\tilde{P}_n^i$. Additionally, the tangential restitution coefficient is set to zero, which requires calibration but is plausible in many impulse-based simulators. As a result, the collision work ΔW^i and ΔW_n^i are used interchangeably, and we focus on the conversion of kinetic to elastic energy in the contact normal direction. For instance, during the compression phase, the relative normal velocity decreases and eventually converts to the elastic energy:

$$\Delta W_n^i = \frac{1}{2}\left(2\tilde{u}_n^i + \Delta\tilde{u}_n^i\right)\tilde{P}_n^i \tag{4.26}$$

Where $\tilde{P}_n^i$ is the normal component of applied impulse. In the separation phase, the absorbed elastic energy reduces and changes the sign of the relative normal velocity via an impulse along the contact normal:

$$\mathbf{K}\mathbf{P}^i = \Delta\mathbf{u}^i \tag{4.27}$$

$$\mathbf{n}^i \cdot \mathbf{K}\left(\tilde{P}_n^i\mathbf{n}^i\right) = \mathbf{n}^i\Delta\mathbf{u}^i = \Delta\tilde{u}_n^i \tag{4.28}$$

$$\tilde{P}_n^i = \frac{\Delta\tilde{u}_n^i}{\mathbf{n}^i \cdot (\mathbf{K}\mathbf{n}^i)} \tag{4.29}$$

$$\Delta\tilde{u}_n^i = -\tilde{u}_n^i + \sqrt{(\tilde{u}_n^i)^2 + 2\Delta W_n^i\mathbf{n}^i \cdot (\mathbf{K}\mathbf{n}^i)} \tag{4.30}$$

4.3 IMPULSE-BASED LS-DEM IMPLEMENTATION

4.3.1 Contact Model

Just as in a penalty-based method, the contact in impulse-based LS-DEM is handled through node-to-surface contact algorithm and the contact normal direction is also interpolated from the underlying signed distance function grid of a grain. The difference is that contact force calculation is skipped in the impulse-based formulation; however, it could be retrieved from the applied impulse with high-precision. The original LS-DEM code (Kawamoto et al., 2016) considers a history-dependent Coulomb friction model that requires grains to be associated with their contact histories. The inter-grain tangential contact is significantly easier to manage in an impulse-based DEM because it does not track friction evolution and updates velocities directly. During the force resolution phase, the impulse-based DEM iterates through the surface nodes between colliding bodies, identifying all contact points with negative relative velocities, and calculating repulsive impulses to satisfy impenetrability constraints.

The friction type contact is examined to ensure that the magnitude of the tangential component of the collision impulse follows Coulomb's friction law. Due to the fact that LS-DEM aims to simulate arbitrarily complex grain shapes, it is unavoidable that multiple contact points collide when two reconstructed avatars are on a collision course. To handle many collisions and various contacts, the simultaneous impulse methods (SMM, Barzel and Barr, 1988; Baraff, 1989) assemble all collisions into a system of linear equations, enforce non-penetration restrictions as a linear complementary problem (LCP) and evaluate outcomes in a single step. However, the SMM fails to capture the propagation of contact forces during a collision. The Sequential Impulse Method (SQM) introduced by Guendelman et al. (2003) resolves each collision within a single time interval and treats the contacts as if they occurred sequentially. This method inverts the relative velocity directly using Newton's law and prioritizes contact points based on the depth of inter-penetration; it is iterative and continues until all contact points have positive relative velocities. The primary disadvantage of SQM is that edge-surface or surface-surface contacts do not end up with similar repulsive velocities and the obtained results being dependent on the sequence in which collisions are addressed. Tang et al. (2014) and Li et al. (2021) address this issue by gradually changing the relative normal velocity and elastic energy in such a way that all contact points with similar normal directions can adjust velocities to similar magnitudes. The compression phase involves incrementally increasing the relative normal velocities at contact points until all relative normal velocities are non-negative. During the separation phase, the absorbed energy is gradually released until no energy remains at the contact points, again in an iterative fashion.

4.3.2 Discrete Equations of Rigid Body Motion in Impulse-Based LS-DEM

Original implementation of LS-DEM (Kawamoto et al., 2016) solves Newton's and Euler's governing equations of motion for translational and rotational components of motion, respectively. For the translational component of motion, the positions, forces, and velocities are known at the end of each time step so that grain motion can be explicitly updated via the governing translational equation given by Newton's law using a centered finite-difference integration scheme. For the rotational components of motion, the time derivatives of the angular accelerations in the principal frame are given by Euler's equations of motion, which is nonlinear and implicit, hence prone to numerical instability.

Again, it is much simpler to change the velocities of two rigid bodies with impulse-based method. For example, if the contact master body b_A experiences a collision contact with b_B under the corresponding impulse $\mathbf{P}^i$, in accordance with Newton's law, the changes in velocity for each rigid body are given by:

$$\Delta\dot{\mathbf{x}}_{A,B} = M^{-1}_{A,B}\mathbf{P}^i_{A,B} \tag{4.31}$$

$$\Delta\dot{\boldsymbol{\theta}}_{A,B} = \mathbf{I}^{-1}_{A,B}\left(\mathbf{r}^i_{A,B} \times \mathbf{P}^i_{A,B}\right) \tag{4.32}$$

The velocities and positions for the rigid bodies are subsequently updated via symplectic Euler scheme:

$$\dot{\mathbf{x}}_{A,B} = \dot{\mathbf{x}}_{A,B} + \Delta\dot{\mathbf{x}}_{A,B} \tag{4.33}$$

$$\dot{\boldsymbol{\theta}}_{A,B} = \dot{\boldsymbol{\theta}}_{A,B} + \Delta\dot{\boldsymbol{\theta}}_{A,B} \tag{4.34}$$

$$\mathbf{x}_{A,B} = \mathbf{x}_{A,B} + \dot{\mathbf{x}}_{A,B}\Delta t \tag{4.35}$$

$$\boldsymbol{\theta}_{A,B} = \boldsymbol{\theta}_{A,B} + \dot{\boldsymbol{\theta}}_{A,B}\Delta t \tag{4.36}$$

4.3.3 Collision Resolution Algorithm

The ETM algorithm is demonstrated with three vertically stacked spheres as shown in Figure 4.2. The concurrent collisions are treated as a series of single point collisions and included an additional iteration within a compression and separation phase to make the results insensitive to the order of the impulses. When dealing with numerous contact collisions, ETM gradually delivers impulses to adjust the relative normal velocities of several collisions, yielding low angular velocity errors. Later, Li et al. (2021) adjusted the time step within the sub-cycle loop adaptively to obtain more stable and energy-conservative rigid body dynamic simulations. In impulse-based DEM, if any pair of rigid bodies is connected to a group of LS avatars via a chain of contacting avatars, collisions at all contact points are resolved simultaneously, as the impulse can propagate within the group. The algorithm begins by iterating through contact points created by collisions and prioritizing those with a negative relative normal velocity. During the compression phase, the priority queue is sorted by minimum relative normal velocity; after the algorithm enters the separation phase, it is sorted by elastic energy. To expedite the process, a group of adjacent contact points is glued

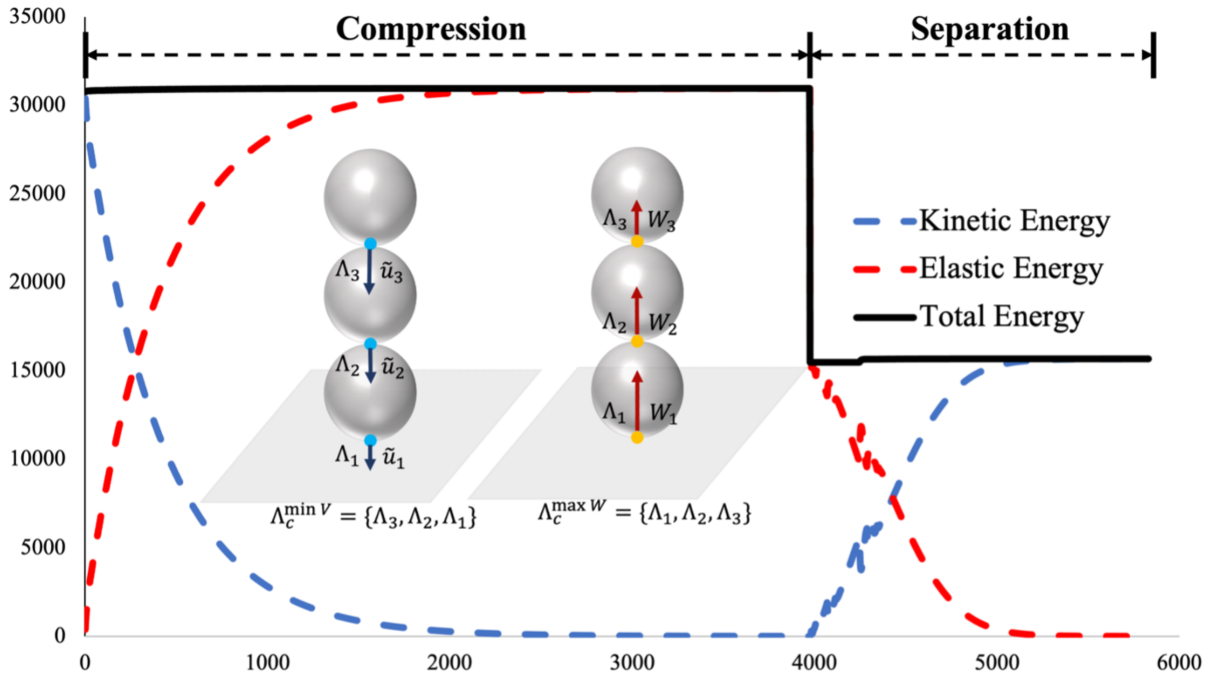

Figure 4.2: Energy conservative property of impulse-based collusion resolution, kinetic energy (blue) and elastic energy (red) during compression and separation phases. The coefficient of restitution is 0.5.

together and treated as a single contact. Kinetic and elastic energy are converted to one another during the compression and separation phases.

The kinetic energy gradually decreases to zero as a result of application of a series of impulses in the compression phase, and the relative velocity reverts to its initial sign during the energy release phase. Following Stronge's hypothesis (setting the coefficient of restitution to 0.5), the restored elastic energy abruptly decreases at the end of the compression phase and gradually decreases to zero during the energy release phase. Although the relative velocity or elastic energy converge arbitrarily close to zero, they do not achieve it numerically. Rather, a threshold value is chosen to truncate small values and to make the algorithm more efficient. The threshold value should be less than the time step, as otherwise velocity changes may not be reflected, thus accumulating errors. The resulting algorithm is presented in Algorithm 3.

4.3.4 Time Stepping Algorithm

Explicit time integration is ideally suited to dynamic contact/impact problems, as a small time step allows for the handling of contact/impact discontinuities. They are frequently used to simulate discrete systems because the explicit methods are robust and simple to implement; they allow element-by-element evaluation and they do not require a global stiffness matrix for a discrete system, making them appropriate for initial value problem like DEM. In a traditional penalty-based DEM, the governing equations determine the resultant force acting on objects and then update accelerations and velocities via a numerical integrator. In comparison, the impulse-based

Algorithm 3 Collision Resolution Algorithm

Input: A list of contacts within a contact island including contact IDs of colliding bodies A and B; branch vectors from centers to contact point $\boldsymbol{r}$; contact normal direction $\tilde{\boldsymbol{n}}$; relative velocity $\tilde{\boldsymbol{u}}$; elastic energy W and body mass M.

```
1: Define parameter $\alpha$ to control velocity increment in sub-iteration, Coulomb's friction coefficient $\mu$, similarity $\beta$ and Stronge's coefficient $\epsilon$.
2: **for** all contact points **do**
3:     Compute inertial matrix in global frame $\boldsymbol{I} = R\boldsymbol{I_0}R^T$, cross-product matrix of colliding bodies $\tilde{\boldsymbol{r}}$, relative normal velocity $u_n$, collision matrix $\boldsymbol{K} = (\frac{1}{M_A} + \frac{1}{M_B})\boldsymbol{I} - (\tilde{\boldsymbol{r}}_A\boldsymbol{I_A}^{-1}\tilde{\boldsymbol{r}}_A + \tilde{\boldsymbol{r}}_B\boldsymbol{I_B}^{-1}\tilde{\boldsymbol{r}}_B)$, wrap computed quantities as $\Lambda_c$.
4: Prepare a priority queue for all contact points with negative relative normal velocity, sorted by the minimum value. $\Lambda^{\min V} := \{\Lambda_c,\ \text{key} = u_n\}$.
5: **while** $\Lambda^{\min V}$ is not empty **do**
6:     /* Compression Phase */
7:     **while** $\Lambda^{\min V}$ is not empty **do**
8:         Generate a list $\Gamma^V$ containing $\Lambda_c$ with minimum relative normal velocity $u_n^{min}$ and those with similar magnitude, $\frac{|u_n' - u_n^{min}|}{|u_n^{min}|} \leq \beta$.
9:         Obtain increment of relative normal velocity: $\Delta u_n^{\min V} = \frac{u_n^{min}}{\alpha \cdot |\Gamma^V|}$.
10:        **for** all contact points in $\Gamma^V$ **do**
11:            $\tilde{\boldsymbol{P}} = \Delta u_n^{\min V}\boldsymbol{K}^{-1}\tilde{\boldsymbol{n}}$, $\tilde{\boldsymbol{P}}_n = \tilde{\boldsymbol{n}}(\tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{n}})$ and $\tilde{\boldsymbol{P}}_t = \tilde{\boldsymbol{P}} - \tilde{\boldsymbol{P}}_n$.
12:            **if** $|\tilde{\boldsymbol{P}}_t| \geq \mu|\tilde{\boldsymbol{P}}_n|$ **then**
13:                $|\tilde{\boldsymbol{P}}_n| = \frac{\Delta u_n^{\min V}}{\tilde{\boldsymbol{n}} \cdot [\boldsymbol{K}(\tilde{\boldsymbol{n}} + \mu\tilde{\boldsymbol{t}})]}$
14:                $\tilde{\boldsymbol{P}} = |\tilde{\boldsymbol{P}}_n|\tilde{\boldsymbol{n}} + \mu|\tilde{\boldsymbol{P}}_n|\tilde{\boldsymbol{t}}$
15:            Calculate change in velocity, $\Delta\tilde{\boldsymbol{u}} = M^{-1}\tilde{\boldsymbol{P}}$, $\Delta\tilde{\boldsymbol{\omega}} = \boldsymbol{I}^{-1}(\boldsymbol{r} \times \tilde{\boldsymbol{P}})$.
16:            Restore elastic energy, $\Delta W = \frac{1}{2}(2u_n + \Delta u_n^{\min V})|\tilde{\boldsymbol{P}}_n|$.
17:        Compute updated relative normal velocities and reset $\Lambda^{\min V}$.
18:    /* Restitution Phase */
19:    **for** all contact points in the contact island **do**
20:        Multiply restored elastic energy by Stronge's coefficient to imitate energy dissipation. $W \leftarrow \epsilon W$.
21:    /* Separation Phase */
22:    Build a priority queue for all contact points with non-negative elastic energy, sorted by the maximum value. $\Lambda^{\max W} := \{\Lambda_c,\ \text{key} = W\}$.
23:    **while** $\Lambda^{\max W}$ is not empty **do**
24:        Generate a list $\Gamma^W$ containing $\Lambda_c$ with maximum restored elastic energy $W^{max}$ and those with similar magnitude, $\frac{|W' - W^{max}|}{|W^{max}|} \leq \beta$.
25:        Obtain change of relative normal velocity at point with largest elastic energy. $\Delta u_n^{\max W} = -u_n^{\max W} + \sqrt{(u_n^{\max W})^2 + \tilde{\boldsymbol{n}} \cdot (\boldsymbol{K}\tilde{\boldsymbol{n}})}$
26:        **for** all contact points in $\Gamma^W$ **do**
27:            $\tilde{\boldsymbol{P}} = \Delta u_n^{\min V}\boldsymbol{K}^{-1}\tilde{\boldsymbol{n}}$, $\tilde{\boldsymbol{P}}_n = \tilde{\boldsymbol{n}}(\tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{n}})$ and $\tilde{\boldsymbol{P}}_t = \tilde{\boldsymbol{P}} - \tilde{\boldsymbol{P}}_n$.
```

```
28:         **if** $|\tilde{\boldsymbol{P}}_t| \geq \mu|\tilde{\boldsymbol{P}}_n|$ **then**
29:             $|\tilde{\boldsymbol{P}}_n| = \frac{\Delta u_n^{\max W}}{\tilde{\boldsymbol{n}}\cdot[\boldsymbol{K}(\tilde{\boldsymbol{n}}+\mu\tilde{\boldsymbol{t}})]}$
30:             $\tilde{\boldsymbol{P}} = |\tilde{\boldsymbol{P}}_n|\tilde{\boldsymbol{n}} + \mu|\tilde{\boldsymbol{P}}_n|\tilde{\boldsymbol{t}}$
31:         Calculate change in velocity, $\Delta\tilde{\boldsymbol{u}} = M^{-1}\tilde{\boldsymbol{P}}$, $\Delta\tilde{\boldsymbol{\omega}} = \boldsymbol{I}^{-1}(\boldsymbol{r}\times\tilde{\boldsymbol{P}})$.
32:         Release elastic energy, $\Delta W = -\frac{1}{2}(2u_n + \Delta u_n^{\max W})|\tilde{\boldsymbol{P}}_n|$.
33:     Compute updated restored elastic energy and reset $\Lambda^{\max W}$.
34: /* some velocities can be negative after above two iterations */
35: Identify all contact points with negative relative normal velocity and update priority queue
    $\Lambda^{\min V}$, $\Lambda^{\min V} := \{\Lambda_c,\ \text{key} = u_n\}$.
```

dynamic simulation modifies velocity directly using a different numerical scheme, obviating the requirement for force computations. This results in the symplectic Euler scheme of first order.

$$\mathbf{v}^{t+1} = \dot{\mathbf{x}}^t + \Delta\dot{\mathbf{x}}^t = \mathbf{v}^t + \Delta\mathbf{v}^t \tag{4.37}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^{t+1}\Delta t \tag{4.38}$$

$$\boldsymbol{\omega}^{t+1} = \dot{\boldsymbol{\theta}}^t + \Delta\dot{\boldsymbol{\theta}}^t = \boldsymbol{\omega}^t + \Delta\boldsymbol{\omega}^t \tag{4.39}$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \boldsymbol{\omega}^{t+1}\Delta t \tag{4.40}$$

Above formulations are different from the the second order time integration methods used in penalty-based DEM, such as the time-centered form below (Walton and Braun, 1993).

$$\mathbf{v}^{t+\frac{1}{2}} = \mathbf{v}^{t-\frac{1}{2}} + \dot{\mathbf{v}}^t\Delta t \tag{4.41}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^{t+\frac{1}{2}}\Delta t \tag{4.42}$$

Which is equivalent to the formulation of Lee and Hashash (2015):

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \dot{\mathbf{x}}^t\Delta t + \frac{1}{2}\ddot{\mathbf{x}}^t\Delta t^2 \tag{4.43}$$

For motion update, the penalty-based DEM is second order accurate, but the impulse-based dynamic simulation updates the motion via a linear change in velocities. In other words, the symplectic Euler scheme analyzes velocity changes in the 'secant' direction, whereas the time-centered Euler scheme considers velocity changes in the 'tangent' direction. Regarding the numerical stability, both schemes are conditionally stable and require a critical time step such that there is limited oscillation in the solution and any numerical error does not build up whereby the computed solution stays close to the truth (Lee and Hashash, 2015). Apart from that, the rigid body dynamic simulation also requires the numerical method to not produce spurious energy gains in the modeled system for a sufficiently long simulation time.

The symplectic Euler integration is also conditionally stable in a sense that the numerical results are bounded only if a small Δt is used that is less than Δt_{cr}. However, because the contact stiffness is omitted from the impulse-based dynamic formulation, the critical time step is determined by the Courant-Friedrichs-Lewy (CFL) condition, which is substantially larger than the criterion for the penalty-based DEM formulation. The CFL condition prevents obvious penetration errors between objects, which implies that the time step is limited by the physics of the problem, where an excessively large time step size results in objects passing through one another, but not by numerical stability issues. Additionally, it possesses the virtue of numerical stability over a long simulation period due to the fact that the total energy of a modeled system is nearly conserved even at large Δt. Thus, the impulse-based simulation may use a time step several orders greater than penalty-based DEM simulations, maintaining comparable computational fidelity. Additionally, the symplectic integrator is superior to other numerical integrators in that it is designed to preserve phase space regions even for very large time steps (Haier et al., 2006). This feature is especially advantageous for modeling naturally deposited granular material with a large grain size distribution, for which standard DEM is prohibitively expensive. Also, while mass scaling is widely employed in DEM (Thornton, 2000), it is not advisable if a dynamic analysis requires high frequency response.

Energy conservation is ensured by the combined usage of the symplectic scheme and iterative collision resolution algorithm. This trait, however, is less desirable when modeling a granular system with a broad variety of shapes and sizes. Consider a small grain sandwiched between two considerably larger grains, both of which are attempting to crush the small grain at opposing contact locations (Figure 4.3 left). The small grain may elastically bounce back and forth in a protracted succession of near-simultaneous encounters. This procedure is unstable in the long run and brings the simulation to a halt. Indeed, even a very small numerical inaccuracy is likely to accumulate and result in the simulation becoming unstable. To address this issue, global damping is introduced, and the velocity change caused by global damping is updated in the following manner:

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}(1 - \xi \Delta t) \tag{4.44}$$

$$\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}(1 - \xi \Delta t) \tag{4.45}$$

$$0 \leq \xi \Delta t \leq 1 \tag{4.46}$$

Where ξ is the global damping ratio. While this approach helps accelerate convergence in the case of multiple collisions in a complicated system, it may still fail when a stiff boundary is involved. Rather being squashed by larger grains, for example, a small grain may rest on a rigid plane in one direction while colliding with a larger grain in the opposite direction (Figure 4.3

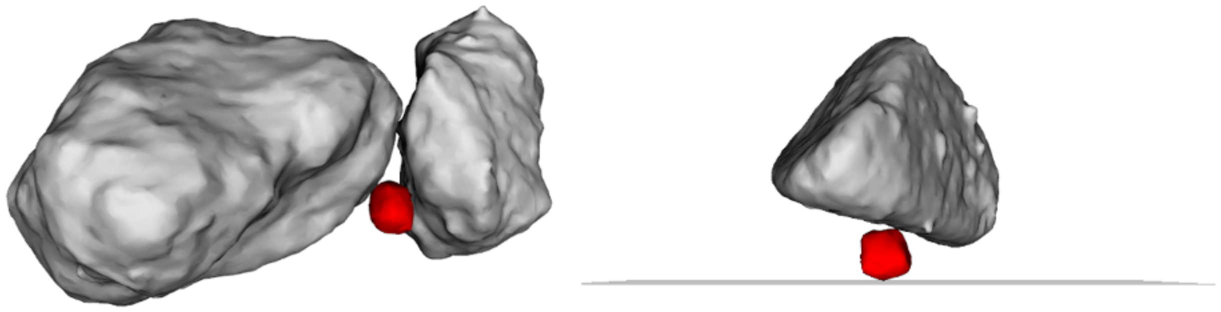

Figure 4.3: Left: A small grain is sandwiched by two larger grains. Right: A small grain is resting on rigid plane while colliding against a large grain. The collision resolution algorithm for these cases might take extremely large number of iterations.

right). The rationale for applying damping to dynamic collisions between grains is that collisions will eventually resolve after a sufficient number of repetitions, and damping is a last choice for expediting this process. Interacting with a rigid boundary, on the other hand, is fundamentally different, as the rigid boundary's velocity cannot be dampened, resulting in a numerically stiff problem as discussed next.

4.4 NUMERICAL CHALLENGES OF THE IMPULSE-BASED LS-DEM

4.4.1 Contact with Rigid Boundaries

The conventional approach for impulse-based DEM is to integrate the equations for rigid body evolution forward in time and then resolve collisions to update velocities. This requires the ETM algorithm to modify the velocity discontinuously and to treat all types of interactions in the same manner to reap the algorithmic benefits. That is, regardless of the magnitude of relative velocity and friction status, both inter-grain and wall-grain interactions are considered equivalent. We will demonstrate how resting contact on rigid wall causes the ETM algorithm to enter an infinite loop when dealing with a system of complex shaped objects. We define the following terms: resting contact refers to the state of objects being stationary, whereas dynamic collision refers to the state of grains experiencing a change in force due to interference. A simplified explanation is that because the velocity of a rigid wall remains unchanged, it violates the premise of the ETM algorithm, which separates two objects by sequentially applying repulsive impulses to both. Collisions require impulses to modify the velocity, whereas contacts are more closely associated with forces and accelerations. As a result, a special treatment is required to distinguish impulse-based collision treatment and needs a penalty-springs approach for at rest contact.

We used a time sequence similar to Guendelman et al. (2003) to distinguish the two types of contacts with the magnitude of the relative velocity suggested by Moore and Wilhelms (1988) and Mirtich and Canny (1995). The critical concept behind this procedure is to detect static contacts that violate impenetrability constraints and correct velocity immediately after the ETM algorithm is used to update the velocity. To avoid an infinite loop during the collision resolution, only dynamic collisions are considered. The general application of velocity correction is dependent on two critical factors: a low relative velocity and rigid boundary interaction. If one of these two criteria is not met, a wall-grain contact is still classified as a normal collision and treated using a collision resolution algorithm. After the ETM algorithm is completed, the object is advanced in the next time step using the newly solved velocities. The interference between rigid walls and grain is then checked, and the grain velocity is corrected, as we want the objects to move to positions where there are no rigid wall interactions. To ensure this, we use the old velocities to predict the positions of the rigid bodies and use corrected velocities to progress through time steps. For instance, in the collision phase, if an object's current position and velocity are $\mathbf{x}$ and $\mathbf{v}$, we test for interference with rigid walls using the predicted position $\mathbf{x}^{'} = \mathbf{x} + \mathbf{v} \cdot \Delta t$, and then apply correction to the current velocity such that the resulting velocity $\mathbf{v}^{*}$ does not interfere with rigid walls. Finally, we advance the object's position $\mathbf{x}^{t+1} = \mathbf{x}^{t} + \mathbf{v}^{*} \cdot \Delta t$. The algorithm's overall structure is as shown in Figure 4.4: it moves all rigid bodies to their predicted locations first, and then it identifies and resolves all grains that penetrate rigid boundaries. This idea is visualized in Figure 4.5, i.e. we consider all nodes within the interpenetrating edges of a grain and move the one with deepest penetration till it is no longer in contact with the rigid wall, angular velocity is also corrected by integrating the repulsive forces due to temporary grain-wall overlaps. After velocity correction, new contacts with negative relative velocities are identified and included in contact islands; we then re-evolve the position using the new post-collision velocity and locate grains penetrating with rigid walls once again. We repeat the process until all contacts are either non-interpenetrating or separating. By design, this time sequence also ensures impenetrability between a grain and the rigid wall; if the velocity correction step occurs before collision resolution, the objects would otherwise pass through the rigid wall. Wriggers and Laursen (2006) and Zohdi (2014) implemented a similar adaptive time-stepping algorithm with convergence criterion to resolve the interaction between the network fabric and the rigid body.

4.4.2 Modeling Deformable Structures

In comparison to rigid bodies, dynamic simulation excludes a wide variety of methods for modeling deformable structures, as the motion of a deformable structure is frequently unpredictable using a mathematical formula, for this reason, it is not considered in a dynamic simulation. As a result, impulse-based methods are frequently ignored or avoided in dynamic simulations of deformable structures, and are instead used exclusively for rigid bodies. The assertion that impulse-based methods can be applied only to rigid body models (Mirtich, 1996) is intuitive: impulses are instantaneous changes in momentum, whereas deformation is a gradual process that occurs over time. However, many applications of DEM, such as numerical modeling of experimental triaxial tests or studies of the interaction between retention barriers and boulders in debris flow simulation, require the modeling of a flexible membrane. The central idea is to represent structures using a group of rigid balls: while each ball changes velocity in a sequence of instantaneous impulses, the configuration of the balls in the group changes in a seemingly gradual manner over time. This is identical to the way we modeled flexible membrane in numerical triaxial compression and we claim that this design does not contradict to the rigid body assumption of impulse-based formulation because individual balls are still rigid. This method is different from the soft-contact model

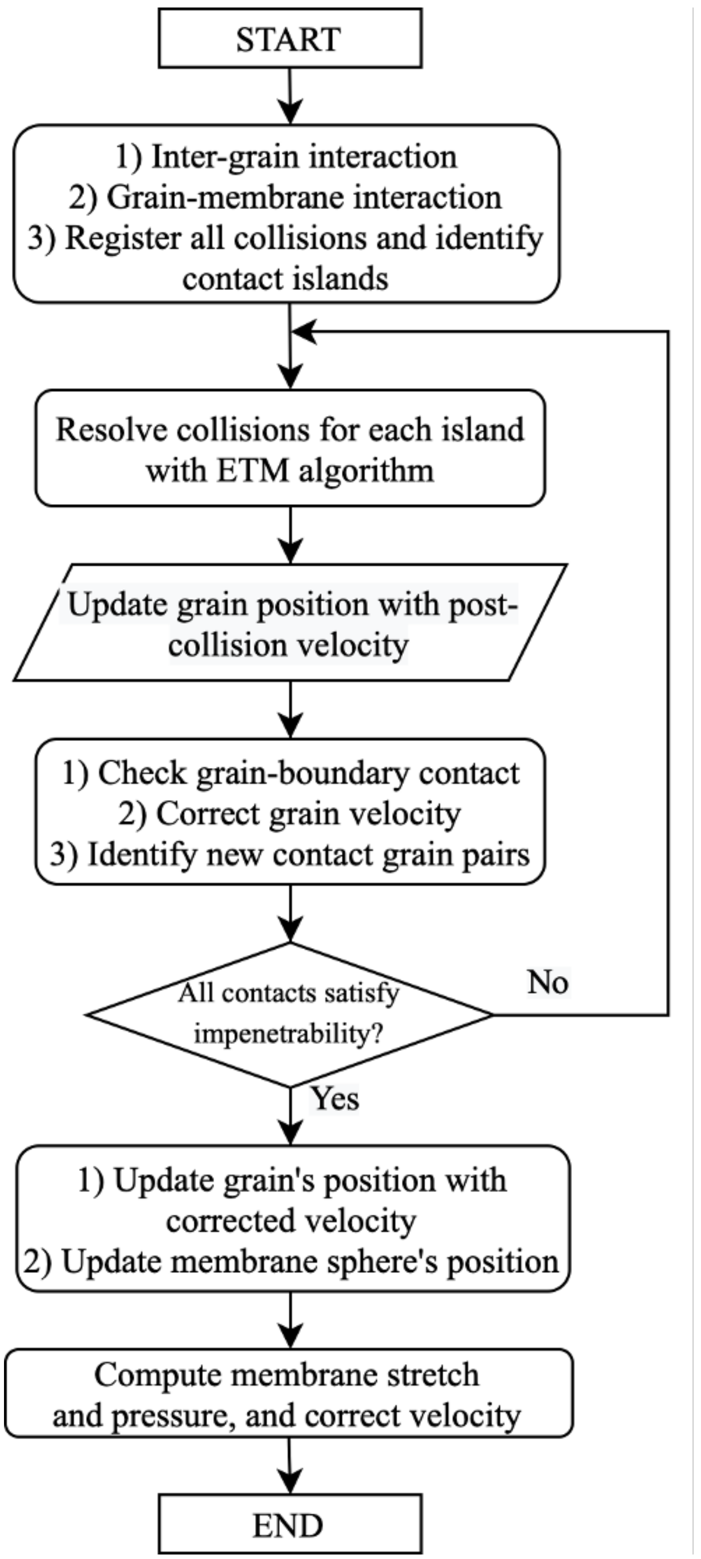
START
1) Inter-grain interaction
2) Grain-membrane interaction
3) Register all collisions and identify contact islands
Resolve collisions for each island with ETM algorithm
Update grain position with post-collision velocity
1) Check grain-boundary contact
2) Correct grain velocity
3) Identify new contact grain pairs
All contacts satisfy impenetrability?
No
Yes
1) Update grain's position with corrected velocity
2) Update membrane sphere's position
Compute membrane stretch and pressure, and correct velocity
END

Figure 4.4: Flowchart of the time stepping integration, post-collision velocities were corrected by checking grain-boundary contact, membrane sphere velocity was corrected after it advanced to the next time step to ensure impenetrability.

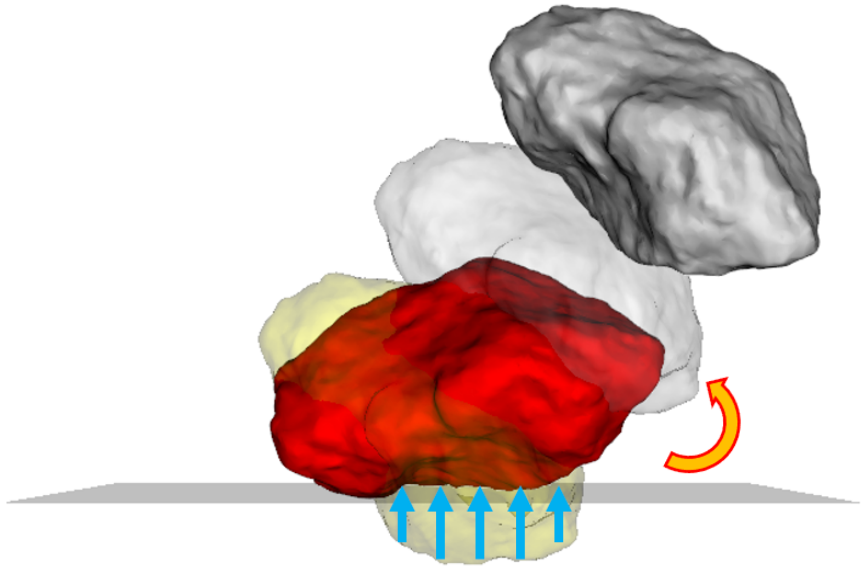

Figure 4.5: Collision time sequence: gray grain obtains post-collision velocity, temporarily translates to the position marked in yellow, subject to repulsive forces and correct velocities to satisfy impenetrability constraints.

used for penalty-based DEM in that the springs used in contact detection and force resolution in penalty-based DEMs are inserted temporarily between detached colliding objects, whereas the springs used in the flexible membrane are permanent, and the distinct balls represent an entire entity. Using a similar approach, Zohdi (2014) simulated a network of coated yarn using coupled fiber-segments, which include damage and plastification of the yarn.

To demonstrate the flexible membrane model, a representative example of several complex shaped grains falling into a flexible net is considered. The flexible nature of the net is modeled as a blanket of linked balls organized in hexagonal patterns with a stiffness of $K = 0.1$ for the internal springs between balls. The grains are subjected to vertical body force and the time step is $t = 2 \times 10^{-4}$; the simulation takes approximately $16,000$ iterations to reach an equilibrium state equivalent to 3.2 physical seconds. Figure 4.6 depicts the simulation results after 0 iteration, $4,000$ iterations, $8,000$ iterations, and $12,000$ iterations. The flexible membrane or net is square and has four clamped edges. All balls along the clamped edges are given an initial velocity of zero, and forces are zeroed out, effectively immobilizing them. External downward forces are applied to grains at regular time intervals via impulses and the grains begin to fall and contact the initially flat membrane. When the membrane comes into contact, it begins to sag and then exhibits a wave-like pattern as a result of internal velocity propagation triggered by the initial fall via a chain of springs. After $12,000$ iterations, the induced waves have mostly subsided, and the grains have settled to form a depression in the membrane. This simulation took only two minutes to complete, which is compelling because it eliminated the need to select regular time intervals. The selection of an appropriate time interval in the case of granular systems is complicated by the fact that interacting entities can differ in size and momentum by many orders of magnitude. Because the size and mass of the membrane ball are much smaller than grains in this example, they would have governed the time step in the conventional penalty-based DEM for the sake of numerical stability, necessitating significantly more iterations to complete the simulation.

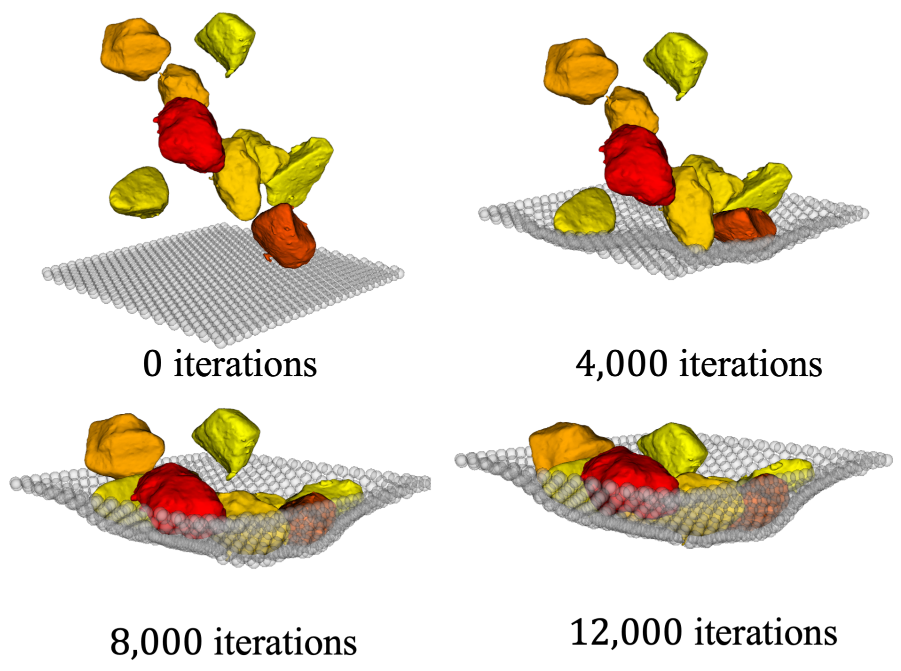

Figure 4.6: Flexible membrane model: grains falling into a flexible net.

Another numerical example is illustrated in Figure 4.7, where an assembly of grains is initially seated on an inclined plane and then subjected to gravity force. Grain collisions are first resolved using the ETM algorithm and then the grain positions are temporarily updated using our new time stepping scheme with the post-collision velocities. When grains collide with a rigid plane as the result of gravity and interference, post-collision velocities are corrected to enforce impenetrability constraints. Then, additional pairs of grains that may collide at new velocities are identified and resolved with the ETM algorithm. When the assembly descends and rolls into the protection net, the membrane spheres interact with the grains, registering and resolving their collisions using the ETM algorithm. The membrane spheres update their positions after evolving velocities to avoid colliding with other spheres and grains. Following that, the velocity of the sphere is altered to account for spring and pressure forces acting on the membrane's surface. The sphere velocity must be corrected immediately following position advancement via post-collision velocity, and the order is critical, as the grain may otherwise pass through the membrane. A network of membrane spheres propagates a velocity wave and causes the entire entity to appear deformable in the presence of sequential collisions. Interestingly, some small grains pass through the protection net due to the stretched spheres creating gaps among them. It is worth noting that the domain re-decomposition strategy improves the efficiency of the parallel code by ensuring that the entire computational domain only encompasses the assembly's geometric configuration and distributes workload evenly across all processors. The mass difference is greater than 400 in this simulation and it does not cause the algorithm to stall, due to the fact that this is a dynamic problem which results in fewer resting contacts.

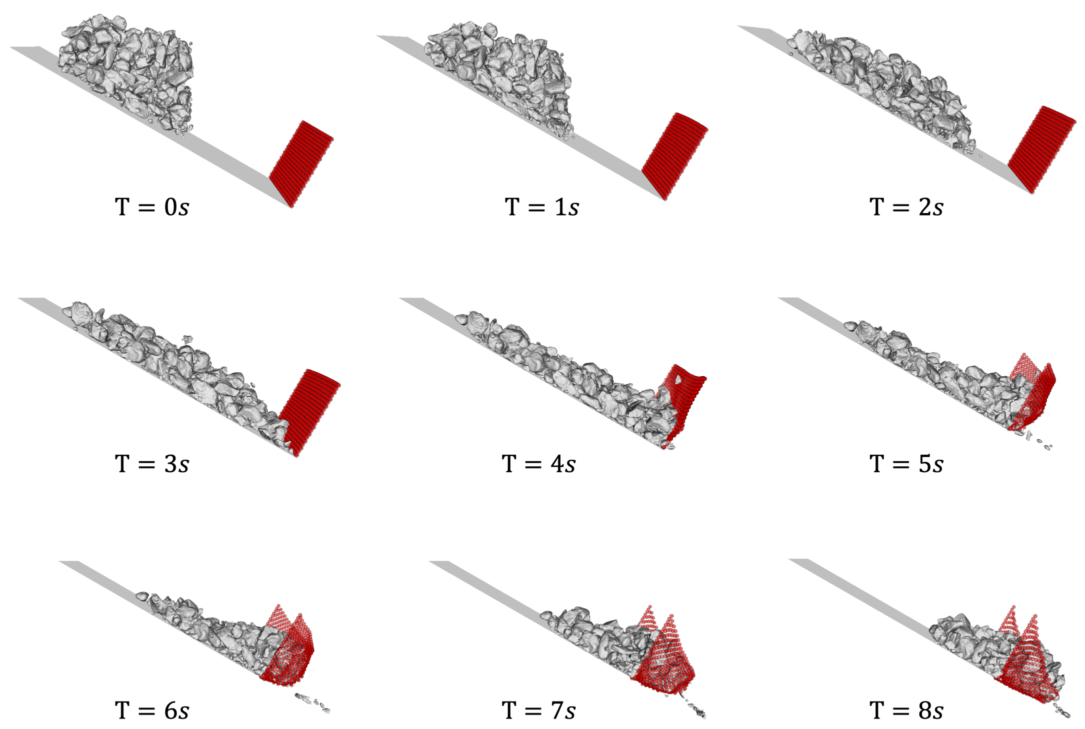

Figure 4.7: Demonstration of the rigid boundary-grain interaction and flexible membrane interaction with the impulse-based LS-DEM code.

4.4.3 Weighted Relative Velocity Implementation

The impulse-based approach is generally used to model similar-sized objects or decompose large objects into uniform elements (Asai et al., 2021), implying that the collision matrix between attached objects is of the same order of magnitude. This does not hold for our application, as we attempted to simulate a granular system of naturally deposited sand with a wide range of grain shape and size, resulting in a colliding pair having significantly different masses. Indeed, this may introduce numerical oscillation, causing small grains to be bounced back and forth unnecessarily due to their greater sensitivity to applied impulses. To address this issue, we modified the original ETM algorithm by considering weighted relative normal velocity, defined as: $\tilde{\mathbf{u}} = \frac{m_1 m_2}{m_1+m_2}\mathbf{u}$, where m_1 and m_2 are the masses of the colliding pairs. In comparison to smaller grains. In this way, the contact points between larger grains are resolved earlier. This approach is superior to simply relying on relative velocity to determine the sequence of contacts, because larger grains respond more gently to impulses, whereas tiny grains tend to oscillate violently even with insignificant amounts of impulses. In addition, this change enables larger grains to achieve desired post-collision velocities more quickly, and when smaller grains take control of the algorithm the resulting update has a negligible effect on larger grains.

4.4.4 Poorly Reconstructed Avatars

There are several issues with low-resolution grain reconstruction that may contribute to the ETM algorithm's inability to converge demonstrated in Figure 4.8. Although we do not require grain avatars to be convex, avatars constructed from low-resolution images with excessive impurities and blurred inter-grain boundaries do not accurately capture grain morphology, i.e., two avatars are merged and form fictitious geometry like the one shown below. When another grain is trapped by this strange-shaped avatar and is subjected to opposing impulses, it results in a numerical instability. Another instance occurs when a small avatar is moved or generated within a larger one, thereby subjecting all nodes to impulses. Both scenarios introduce convergence issues for the ETM algorithm because one object is subjected to impulses from opposite directions exerted by a single object; this is analogous to one object being squeezed by two rigid boundaries concurrently. As previously stated, this makes it difficult for a collision resolution algorithm to converge. Numerous numerical issues also arise during high-resolution reconstruction of particles from XRCT, primarily as a result of the preservation and capture of even the tiniest grains in high-resolution images. As a result, the mass difference between the largest and smallest grains can be as large as $1,000$, resulting in a numerically intractable system. Of course, this issue can be avoided by artificially creating avatars that do not suffer from the above mentioned artifacts.

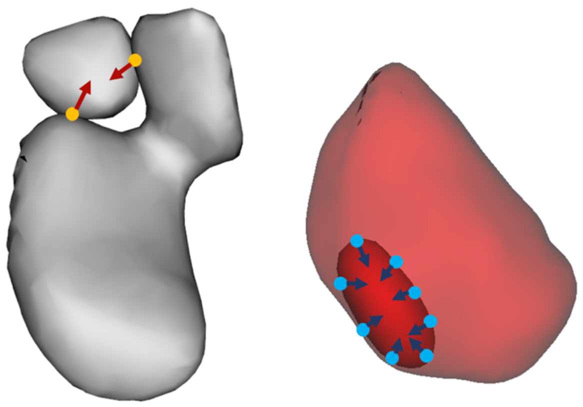

Figure 4.8: Two scenarios resulting in numerical instability. Left : One avatar is trapped by another; Right : Small avatar is completely enclosed in another.

4.5 PARALLEL IMPLEMENTATION OF IMPULSE-BASED LS-DEM

The impulse-based LS-DEM continues to employ a domain decomposition strategy to improve performance. As with the penalty-based LS-DEM, the binning algorithm is implemented through a series of linked lists that map relationships between grains and bins with a search complexity of $O(1)$. There are two types of data communication to consider: border communication and grain migration. The first type is treated identically to the penalty-based LS-DEM, and an efficient algorithm consisting of three sequential calls to the optimally tuned MPI routine MPI_Sendrecv is sufficient to update exchange halo layers and automatically handle edge cases. Grain migration across sub-domains is a complicated step in the penalty-based LS-DEM because the contact history of a migrating grain is also brought along if the history-dependent tangential contact model is used. However, it is much easier to handle in an impulse-based method because it does not track the evolution of friction forces, allowing all quantities to be packed and communicated collectively, thereby lowering communication overheads.

The domain decomposition strategy divides a large computation task into smaller tasks and assumes that each sub-domain has access to all necessary resources to run independently. This is consistent with the penalty-based method, in which all grain motions are a result of contact forces and have no relationship to distant grains. Because the collision reaction is associated with a change in the microstructure at the contact point, the change in acceleration must occur over a positive time interval. Indeed, the forces take some time to propagate throughout the body due to the body's elastic nature. Collision contact, however, occurs within a very short time interval for a rigid body, which means that displacement and change in the contact area are negligible or remain unchanged. Thus, in a penalty-based DEM simulation, a body is influenced solely by its immediate contact neighbors, and it is sufficient to consider a layer of halo region for data communication across computational sub-domains. While the grain interactions can be reconciled element by element using a penalty-based approach, a contact island, as illustrated in Figure 4.9, in which any pair of grains is connected via a chain of immediately contacting grains, must be solved concurrently to satisfy the impenetrability constraints in the impulse-based formulation. Thus, one of the difficulties in parallelizing impulse-based LS-DEM is that a contact island can span an arbitrary number of sub-domains, resulting in grains within a sub-domain becoming indistinguishable from those in other sub-domains. Additionally, the penalty-based method obtains boundary interactions from the halo layer and updates the grain's motion independently of other processors, whereas the impulse-based method requires one processor to gather and resolve all collisions in a contact island.

The force resolution step of impulse-based method continues to use an $O(n)$ binning algorithm. This step identifies and registers all contact points that violate impenetrability constraints, including contact IDs, branch vectors, and the normal direction of a collision between two avatars. Following that, the master MPI processor (rank 0) collects these contacts and summarizes a list of contact IDs, determining the number of contact islands and grains contained within. The concept of obtaining contact islands was inspired by the fact that colliding bodies make contact with one another, and that the system of all colliding pairs forms an undirected graph. As a result, the problem of determining contact islands becomes a graph partition problem of determining all connected components. Additionally, we can consider the graph as a sparse, symmetric square matrix whose dimension equals the number of grains and whose entries are non-zero only when the corresponding pair of grains collides. When the problem is rephrased as permuting such a matrix into a band matrix with a small bandwidth, the Cuthill-Mckee algorithm (Cuthill and McKee, 1969) is used, which is based on the graph's Breath First Search (BFS) algorithm. Algorithm 4 contains the Cuthill-Mckee algorithm optimized for our application. Following the identification

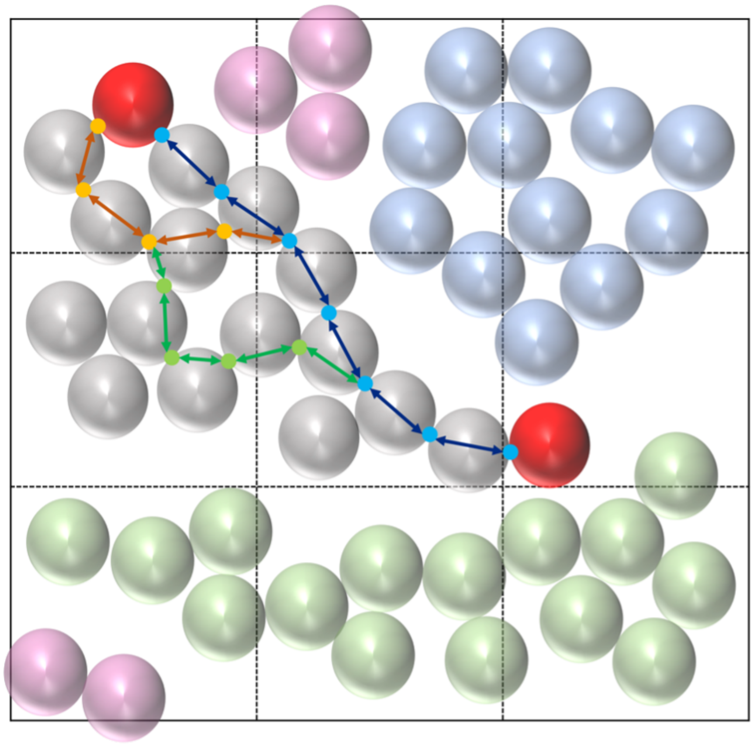

Figure 4.9: Illustration of contact islands (indicated in different colors) in which a group of bodies is associated via a contact chain which can span over several sub-domains. For example, two red bodies in a same group could influence each other via alternate paths.

of contact islands, collisions between rigid bodies are resolved within each contact island, which may span multiple sub-domains. To minimize data communication, a processor gathers a contact island that already contains the majority of collisions. Due to the small and specific size of the data being communicated, it can be packed and transferred efficiently and collectively. After collision resolution is complete, the change in velocity of each grain is computed and distributed back to the grain's original process, which updates the velocity of the grain and advances the time step. The preceding procedure formalizes the parallel strategy used in the Algorithm 5 for impulse-based LS-DEM.

The flowchart of the parallel impulse-based LS-DEM code is shown in Figure 4.10. There are several implementation details worth noting. As the basic element of impulse-based method is an island of collisions which could spread over multiple processors, each processor still needs to generate a simplified avatar containing mass, moment of inertia for those that do not belong to the sub-domain, and it has to keep track of the position, rotation, velocity, and angular velocity of all grains. In addition, both post-collision velocity that is attained from solving a contact island and the corrected velocity after interacting with rigid boundaries must be broadcast to all processors before updating grain's motion. Moreover, since each processor interacts with its neighbors via halo layers, the grain-halo contacts are duplicated when contacts in an island that spreads multiple processors are gathered to one processor, hence half of interactions with halo layers needs to be removed.

Algorithm 4 Optimized CutHill-Mckee Algorithm

```
Input: A list $L^{\text{ID}}$ with length of $N_{\text{grains}}$ including IDs of contact neighbors for all grains in assembly.
Output: the number of contact islands in an assembly $N_{\text{islands}}$, a list $\Lambda^{\text{islands}}$ of size $N_{\text{islands}}$ such that $\Lambda^{\text{islands}}[i]$ denotes IDs of elements in the $i$-th contact island.
1: From $L^{\text{ID}}$ generate a list $L^{\text{degree}}$ denoting the degree of an element, and a list $L^{\text{checked}}$ denoting if an element is looked up already.
2: Set the number of contact islands $N_{\text{islands}} = 0$, the total number of checked elements $N_{\text{checked}} = 0$; and prepare a priority queue $\Lambda^d$ for elements in the assembly, sorted by the minimum degree.
3: while $N_{\text{checked}} < N_{\text{grains}}$ do
4:     Iterate $L^{\text{degree}}$ to find an element $i$ with minimum degree among all unchecked elements.
5:     Sort all elements in $L^{\text{ID}}[i]$ into $\Lambda^d$ if they are not already, and set $L^{\text{checked}}[i] = \textit{true}$.
6:     while $\Lambda^d$ is not empty do
7:         Pop out the first element $j$ from $\Lambda^d$.
8:         if $j$ is not already in $\Lambda^{\text{islands}}[N_{\text{islands}}]$ then
9:             Insert $j$ into $\Lambda^{\text{islands}}[N_{\text{islands}}]$, and set $L^{\text{checked}}[j] = \textit{true}$.
10:            Sort all elements in $L^{\text{ID}}[j]$ into $\Lambda^d$ if not already.
11:    $N_{\text{checked}} \leftarrow N_{\text{checked}} + |\Lambda^{\text{islands}}[N_{\text{islands}}]|$
12:    $N_{\text{islands}} \leftarrow N_{\text{islands}} + 1$
```

4.6 NUMERICAL TESTS

4.6.1 Performance Speedups with Impulse-Based LS-DEM

The speed-up of impulse-based LS-DEM was investigated by modeling the flow of 1600 arbitrarily complex shaped grains under artificial gravity from a smooth container, as shown in Figure 4.11. The grains employed in the assembly have wide range of grain shapes and sizes, making the time-step of conventional penalty-based approach very small. Table 4.1 shows model parameters for the penalty-based and the impulse-based LS-DEM. The values for inter-grains friction coefficient μ, normal contact stiffness K_n, and shear contact stiffness K_s were adopted from Kawamoto et al. (2016). The coefficient of normal restitution according to Stronge's hypothesis is taken $R_n = 0.65$, which was obtained by Li et al. (2021) by conducting experiments. The coefficient of tangential restitution R_t needs also to be calibrated, but many impulse-based simulators show good physical plausibility with a default value of zero (Lee and Hashash, 2015). Global damping is $\xi_g = 0.5$ for both approaches. Different time steps for impulse-based LS-DEM (Δt_i) were chosen to study the speed up as well as the simulation fidelity over the reference penalty-based simulation. The time step of penalty-based LS-DEM was computed by assuming a factor of safety 0.1 and the value 2.5×10^{-4} is selected for the ease of comparison.

$$\Delta t = 0.1 \times \sqrt{\frac{K_n}{M_{\min}}} \approx 2.98 \times 10^{-4} \tag{4.47}$$

In the simulations, the outer walls of the container were removed and the sand was allowed to flow out. The geometries of the resulting mounds, approaching the angle of repose, were com-

Algorithm 5 Algorithm to Parallel Contact Islands

Input: MPI rank i maintains a list Λ_C^i of contact details Φ_C including contact IDs of colliding bodies A and B; branch vectors from centers to contact point $\boldsymbol{r}$; contact normal direction $\tilde{\boldsymbol{n}}$; relative velocity $\tilde{\boldsymbol{u}}$; elastic energy W and body mass M, such that $\Lambda_C^i := \{\Phi_C : \Phi_C \in \text{rank i}\}$.

```
1: /* Start Graph Partition to Detect Contact Islands */
2: if $rank\_id \neq 0$ then
3:     Generate a list $\Lambda^i_{\text{ID}}$ contains IDs of contacting neighbors for each grain belongs to the rank.
4:     MPI_Send $\Lambda^i_{\text{ID}}$ to rank 0.
5: else
6:     for $rank\_id = 1 : N_{\text{ranks}}$ do
7:         MPI_Recv $\Lambda^i_{\text{ID}}$ from rank $rank\_id$.
8:     /* Prepare For the CutHill-McKee Algorithm */
9:     Build a list $L^{\text{ID}}$ with length of $N_{\text{grains}}$ to register contact IDs for all grains in assembly.
10:    Prepare a list $\Lambda^{\text{islands}}$ of size $N_{\text{islands}}$ such that $\Lambda^{\text{islands}}[i]$ denotes IDs of elements in the $i$-th contact island.
11:    $[N_{\text{islands}}, \quad \Lambda^{\text{islands}}] =$ CutHill-McKee$(L^{\text{ID}})$.
12:    for $island\_id = 1 : N_{\text{islands}}$ do
13:        determine the host rank that contains most collisions of the $i$-th contact island, and register the host ranks in list $L^{\text{host}}$ with length of $N_{\text{islands}}$.
14: /* Broadcast Contact Islands Information */
15: MPI_Bcast $\Lambda^{\text{island}}$ and $L^{\text{host}}$.
16: /* All-To-All Communicate Contact Detail $\Phi_C$ */
17: MPI_Alltoall number of contact details that each rank should receive from others.
18: MPI_Alltoallv specific contact details that each rank should receive from others.
19: /* Collision-Resolution Phase */
20: for $island\_id = 1 : N_{\text{islands}}$ do
21:     if $L^{\text{host}}[island\_id] == rank\_id$ then
22:         Resolve collisions via Collision-Resolution Algorithm.
23: /* Broadcast and Update Velocities */
24: MPI_Allgatherv linear and angular velocities from other ranks.
```

pared at the end of 8s time interval. In each time step, the post-collision velocities were corrected for rigid boundary contact until the computed velocities did not violate either inter-grain or grain-wall impenetrability. In this example, we repeated this procedure several times within a single time step, and we found 10 iterations were enough to stabilize the velocity of grains. The code speed-up was measured in terms of central processing unit (CPU) time. Five simulations using impulse-based LS-DEM were conducted, with the time step used varying from $\Delta t_i = 1\Delta t = 2.5 \times 10^{-4}$ to $\Delta t_i = 16\Delta t = 4.0 \times 10^{-3}$. Figure 4.12 shows the mound geometry for each simulation and Table 4.2 tabulates the speed up times for the different time steps in the impulse-based DEM simulations. The simulation fidelity can be checked in terms of final height and spread of mound. Although all simulations produced similar shaped mounds, discrepancies in positions and rotations of individual grain can be large. Calibrating impulse-based method to achieve reasonable agreement to reference penalty-based simulation is not easy, as it requires the identification of actual or ad hoc internal

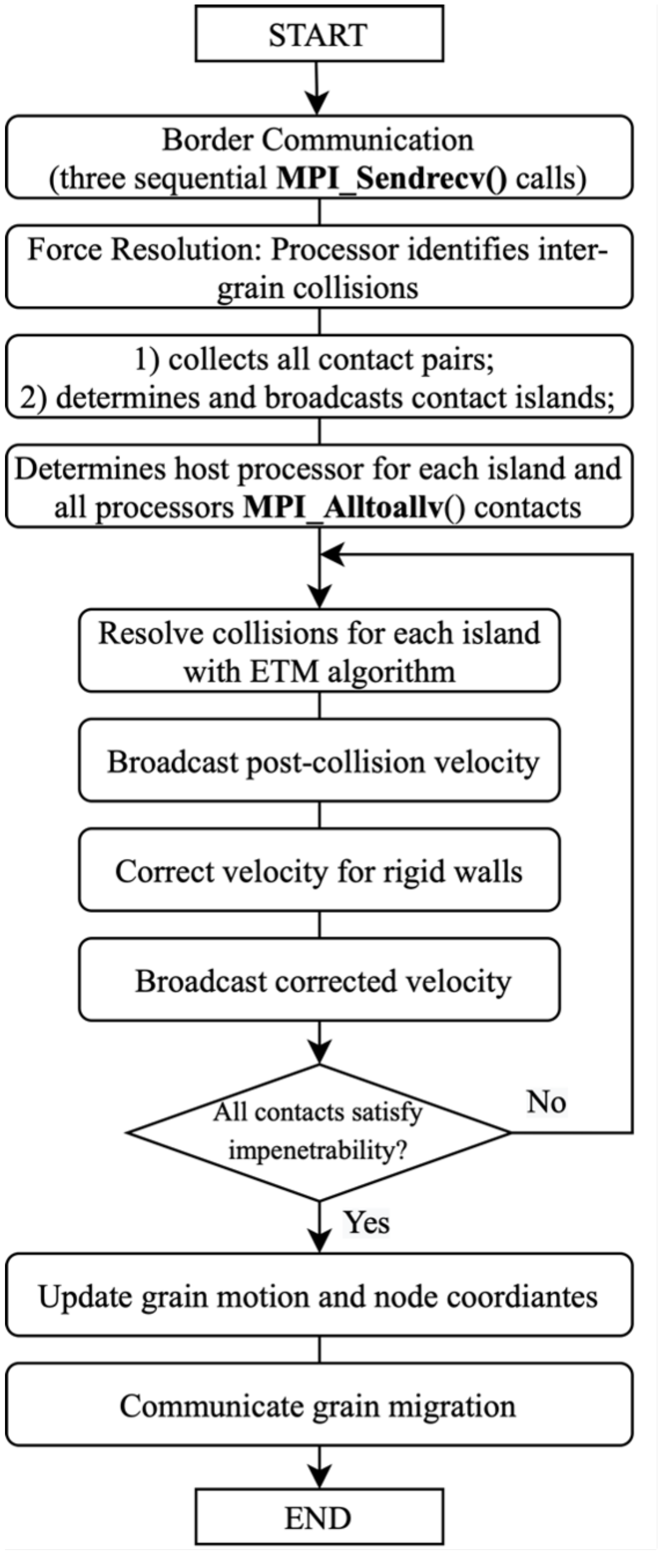
START
Border Communication (three sequential **MPI_Sendrecv()** calls)
Force Resolution: Processor identifies inter-grain collisions
1) collects all contact pairs; 2) determines and broadcasts contact islands;
Determines host processor for each island and all processors **MPI_Alltoallv**() contacts
Resolve collisions for each island with ETM algorithm
Broadcast post-collision velocity
Correct velocity for rigid walls
Broadcast corrected velocity
All contacts satisfy impenetrability?
No
Yes
Update grain motion and node coordiantes
Communicate grain migration
END

Figure 4.10: Flowchart of parallelized impulse-based method.

variables and constitutive relations. In addition, there are too many degrees of freedom even for a single grain and to make a one-to-one comparison is not feasible. Instead, we aimed to explore to

Table 4.1: List of model parameters and values used for the simulations.

Common Parameters		
Specific gravity of solids	G_s	2.65
Inter-grain friction coefficient	μ	0.55
Global damping coefficient	ξ_g	0.5

Penalty-based LS-DEM			Impulse-based LS-DEM		
Normal contact stiffness	K_n	3×10^4	Coefficient of normal restitution	R_n	0.65
Tangential contact stiffness	K_s	2.7×10^4			

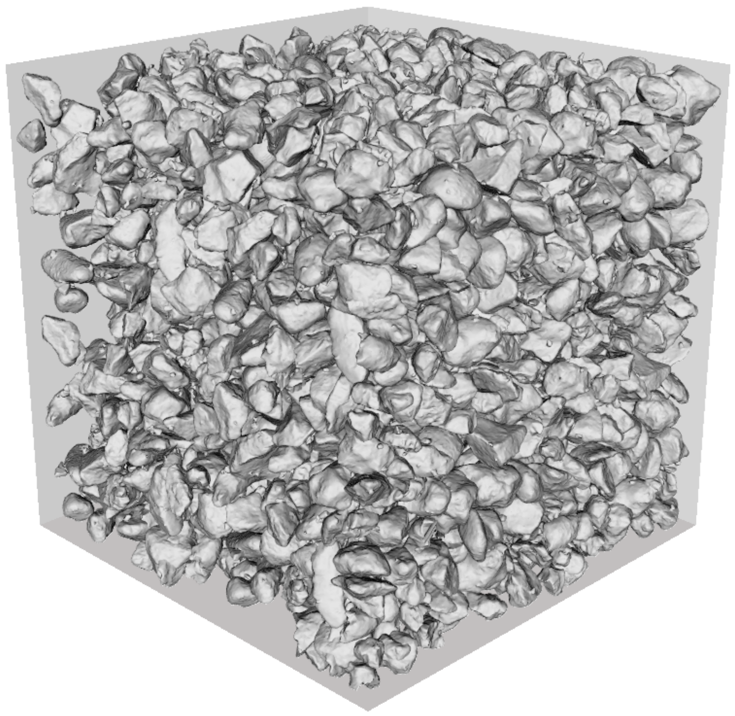

Figure 4.11: Initial configuration of 1600 very high-resolution grains in $800 \times 800 \times 800$ domain for impulse-based method with large time steps.

what extent both methods agree with each other in a qualitative sense. At the same time we found that the simulation results, even using a very large time steps, showed no obvious penetration errors between particles and the geometric fidelity was maintained across all simulations.

In this simulation, the particles were reconstructed from a very high-resolution data set ($3.4\mu m/$pixel) and each grain was represented by $2,561$ nodes on the average and storing $1,600$ grains required 3.36 GB of memory. We rarely used very high-resolution avatars for large scale simulation with penalty-based method although we have an efficient parallel code, because actual workload corresponds to the total number of nodes rather than the number of grains. In contrast,

Table 4.2: Comparison of CPU time and speed up between different methods.

	Δt (sec)	CPU Time (mins)	Time Step Diff	Speed-up
Penalty-based	2.5×10^{-4}	$\sim 4,000$		
Impulse-based	2.5×10^{-4}	265	$1\Delta t$	15.1
	5.0×10^{-4}	132	$2\Delta t$	29.4
	1.0×10^{-3}	72	$4\Delta t$	55.6
	2.0×10^{-3}	42	$8\Delta t$	95.2
	4.0×10^{-3}	42	$16\Delta t$	95.2

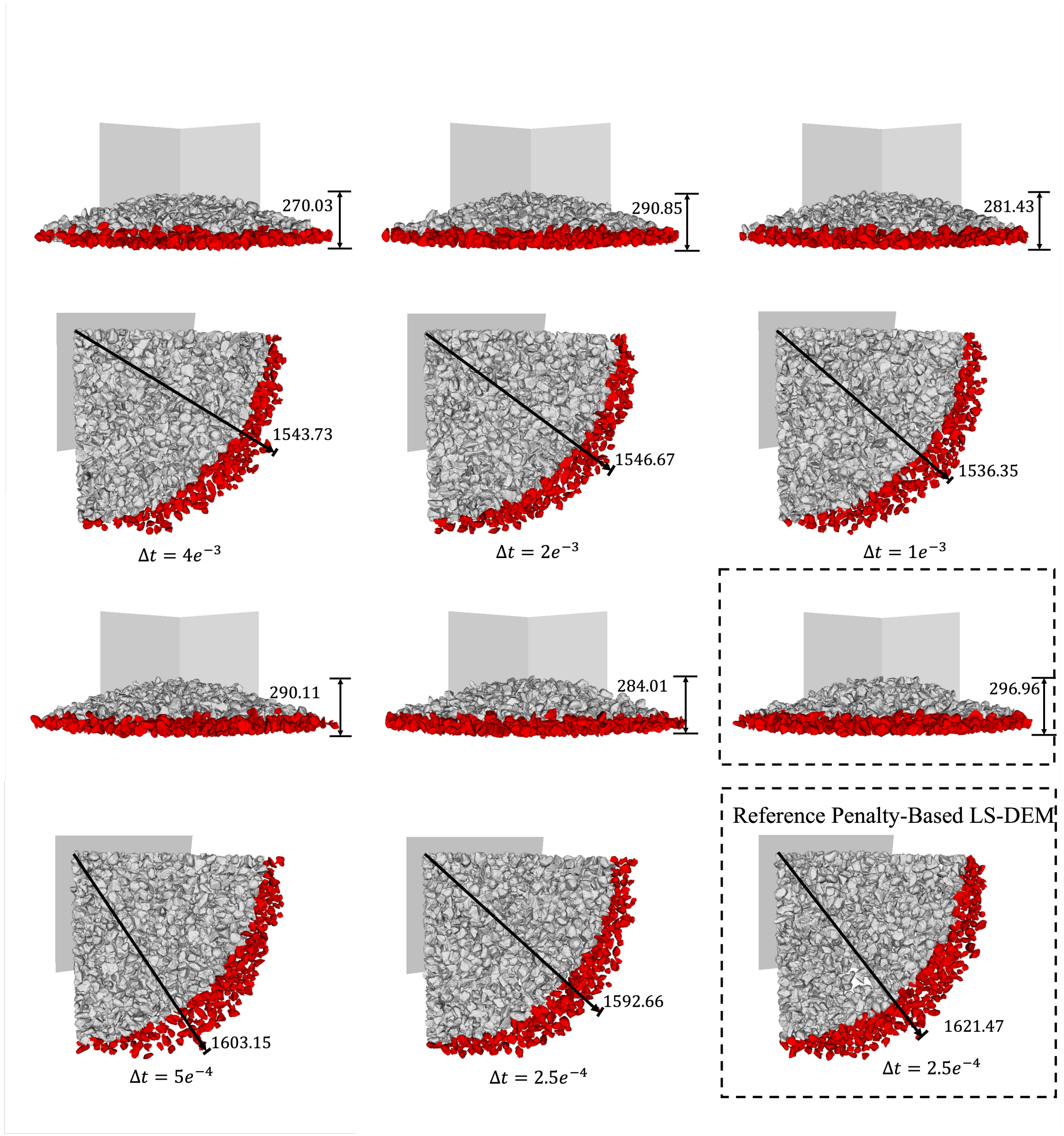

Figure 4.12: Specimen height and spread after 8s settlement under gravity with varying time steps.

there are fewer restrictions to model high-resolution avatars with impulse-based method. Even though impulse-based method still interpolates normal direction from the LS grids, the impulse-based LS-DEM does not compute, update, and track complicated history-dependent friction. This simplified force resolution and helped to keep LS grids in cache and enabled continuous, fast access. Besides, the amount of time required for sub-iterations to resolve multiple collisions is less demanding because doing arithmetic operations is hundred times faster than moving data around in modern machines.

Significant speed up shown in Figure 4.13 is achieved in impulse-based method with larger time steps, which is not likely for conventional penalty-based DEM. As grains settled down, the number of contact islands decreased and the number of contacts in an island increased. We observed that as many as 300 contact islands still remained at the end of the simulation, the largest number of contacts in an island was more than $2,000$, which implies that most of the grains were grouped into several clusters and the time used to resolve collisions increased. We also found that both the number of contact islands and contacts increase with the time step as a larger time step changes configuration more rapidly and adds more collisions. The ratio of mass differences between the largest and the smallest grains in this example was around 30, and we found this is a ratio that our algorithm remained stable and converged at different time steps, accelerations, and boundary conditions.

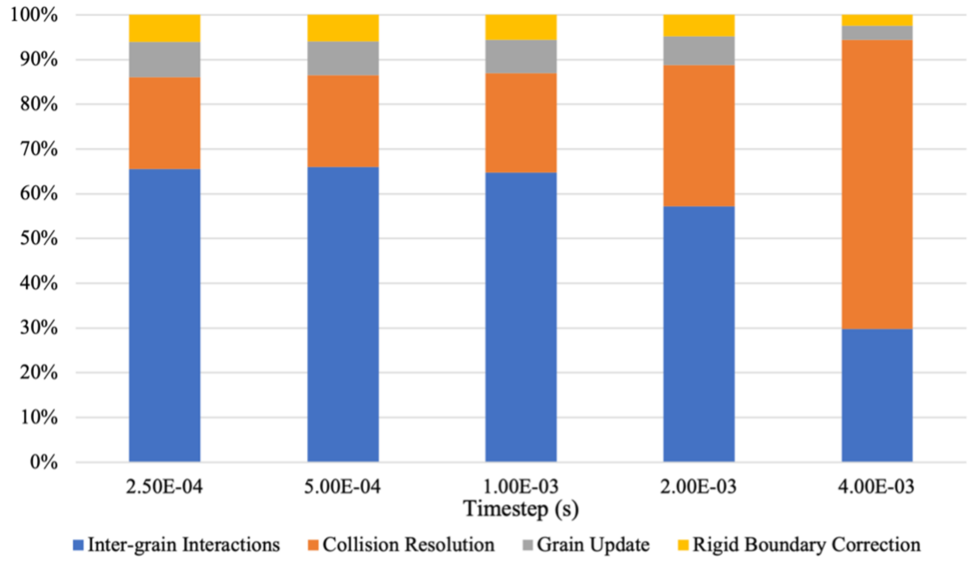

Figure 4.13: Simulation time breakdown as a function of time step.

Figure 4.14 shows why computing time for simulations using time step 2.0×10^{-3} and 4.0×10^{-3} does not change substantially. Although none of simulation results violated CFL condition and incurred obvious penetration errors, larger time step identified more contacts and required substantially more time to solve ETM algorithm. As shown in Figure 4.14, the resolution time for contact islands with various number of collisions is nonlinear. This implies that, although not violating the physics of the problem whereby large time step results in grains passing through each other, the time step should also be chosen to avoid spending too much time in collision resolution.

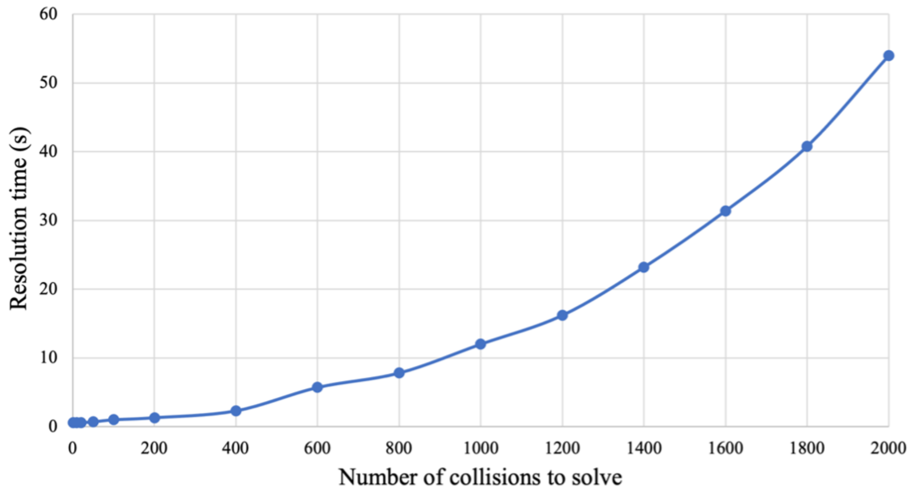

Figure 4.14: Computing time for islands with different numbers of collisions, all measured for 5 time steps and 10 sub-iterations for rigid boundary correction.

4.6.2 Efficiency Analysis of Parallel Impulse-Based LS-DEM

To benchmark the performance of the parallel impulse-based LS-DEM, a series of numerical experiments were performed on the Savio system of UC Berkeley. The objective was to measure the performance improvement by integrating an $O(n)$ algorithm and using parallel techniques. We used XRCT data to reconstruct the observed sand fabric with $155,016$ avatars as shown in Figure 4.15. The reconstructed avatars were tightly packed which is not the favored type of problem for an paralleled impulse-based code for two reasons. First, the system of granular particles has wide range of masses, shapes, and sizes; second, almost every grain is connected in a single cluster and all inter-grain interactions are static contacts. These issues make the problem numerically too stiff to be modeled as an impulse dynamic problem. To tackle such limitations, we removed avatars with mass two standard deviations away from the mean to make the mass ratio between the grains at most 30 and with 299 nodes per grain on average, the model required 12.6GB memory for morphology files. The example problem is simple: domain boundaries are modeled as undeformed planes; grains are restricted from leaving the domain and would be bounced back if it intends to do so. For simplicity, grains are subjected to random accelerations in each time step to secure load balance. The bookkeeping and adaptive binning subroutines are switched off when the performance is benchmarked. With given computing resources, the code could adaptively determine the number of processors that are actually utilized such that the total border/ghost area or the amount of communication across processors is minimized.

Figure 4.16 lists the simulation time breakdown for the parallel impulse-based LS-DEM. While the serial portion of code consumed less time with more processors involved, communication overheads nevertheless began to govern and computing time for collision resolution phase soon ceased to decrease. All speed-ups were measured relative to a serial run, which is equivalent to an MPI simulation with a $1 \times 1 \times 1$ decomposition. The performance gains by using 8, 27, 64 and 216 processors are 5.5, 7.0, 7.3 and 5.0, respectively. As is evident, a parallel code cannot achieve high efficiency unless communication overheads are kept to a level, beyond which more

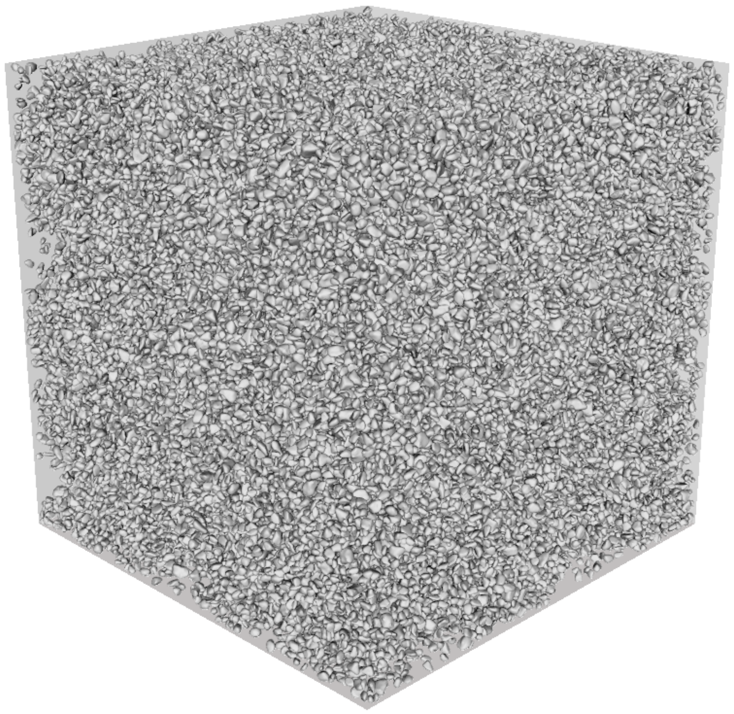

Figure 4.15: $155,016$ **low-resolution grains in** $1,200 \times 1,200 \times 1,200$ **domain to benchmark parallel impulse-based LS-DEM.**

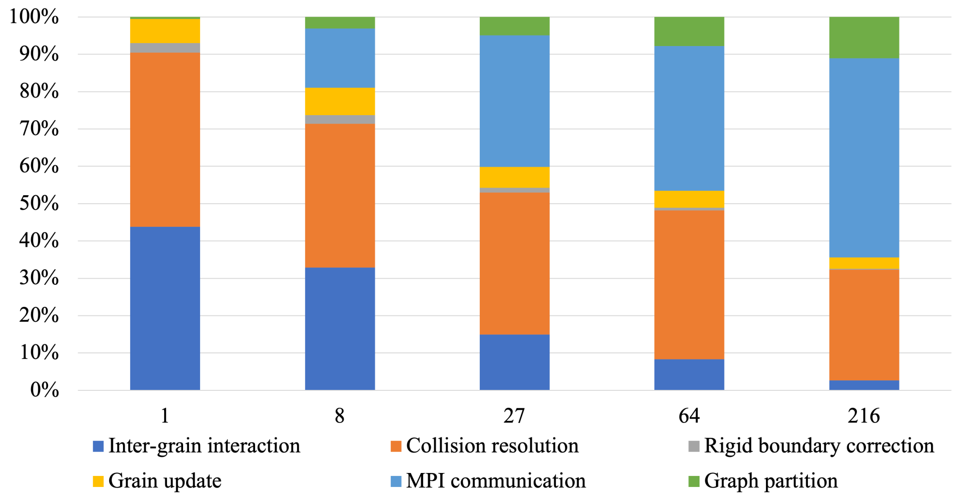

Figure 4.16: Simulation time breakdown for parallel, impulse-based LS-DEM, modeled with $155,016$ **grains, with** 1**,** 8**,** 27**,** 64**,** 216 **processors, respectively.**

MPI processors lead to an increased amount of MPI traffic and stall or even deteriorate scalability. In this case, the impulse-based method could not be as perfectly parallelized as we achieved with the penalty-based method, but there is still a sweet spot, eight processors in this specific case, where the communication is still modest, while increasing the number of processors to beyond 64 no longer brings additional benefit as the MPI communication starts overwhelming the other operations.

4.6.3 Numerical Modeling of Rock Avalanches

Even though Impulse-based DEM is capable of taking much larger time steps and thus eliminate a substantial amount of the computational effort, the fundamental issue of a discontinuum-based method to model an assembly with wide range of object sizes still remains. This is the issue we already addressed when discussing the problem of modeling fixed, or rigid boundaries. In that context, modeling topography presents an analogous problem.

Within the LS framework, any arbitrarily shaped surface, such as a real topography with ridges and valleys, can be captured. In this example, the LS algorithm was first used to locate the zero-valued contour, from which the fast-marching method (Sethian, 1999) was used to construct the underlying signed distance grid by solving a boundary value problem of Eikonal equation:

$$|\nabla\phi\,(x)| = \frac{1}{f\,(x)} \quad \text{for} \quad x \in \Omega \tag{4.48}$$

$$\phi\,(x) = 0 \quad \text{for} \quad x \in \partial\Omega \tag{4.49}$$

Such a problem describes the evolution of a surface as a function of LS ϕ with speed $f(x)$ in the normal direction at a point x on the propagating surface. Alternatively, $\phi(x)$ can be thought of as the shortest time required to reach x starting from the zero contour, of course, $\phi\,(x) = 0$ for x on the zero LS contour $\partial\Omega$. The algorithm is similar to Dijkstra's algorithm and uses the fact that information only flows outward from the area that is already labelled with a value. In this example, a digitized topography (see, Figure 4.17) and its LS representation are constructed as a large LS-DEM object to exploit algorithmic advantages and we treat grain-topography interaction identically to grain-grain interaction. The digitized topography occupies the entire computational domain and consumes as much as 10GB computer memory. To avoid large memory demand, we only save a narrow band of the grid adjacent to the zero LS contours, which is sufficient to interpolate the amount of penetration between objects and keep the memory requirement low.

We show three simulated landslides in which a granular deposit assembly was perturbed by gravity and allowed to slide as gravity overcame the internal friction of the deposit. The digitized topography is inclined at $25°$, and the coefficient of internal friction between the particle and the topography is 0.2. Larger particles were visualized in red, while smaller particles were displayed in orange. In all cases, the particle mass, rock avalanche, moves fluid-like down the stream channel. In the first example (Figure 4.18), the avalanche ran out on a flat plane and formed fan-shaped deposit, as is typical of rock avalanche deposits. In the second example, shown in Figure 4.19, the deposit ran out onto a surface sloping at only $3°$, again similar to many natural settings, and was deflected downslope. In the third example (Figure 4.20 and Figure 4.21), we simulated different geometries of flexible barriers, such as are often used for rock fall protection. In both cases, the

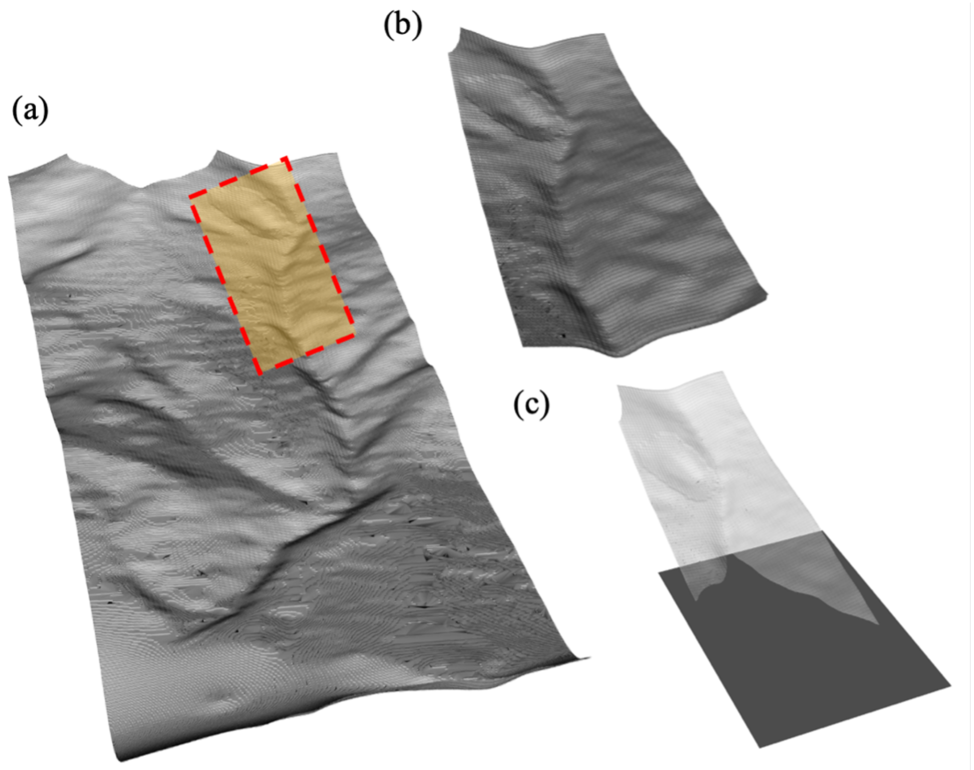

Figure 4.17: (a) Entire digitized topography. (b) A zoomed in region marked in yellow. (c) A zoomed in region concatenated with a horizontal plane.

avalanche was initially contained until it overflowed the barriers. Our interest in these simulations was to explore the potential of the impulse-based DEM to realistically handle field-size problems rather than solve a particular problem. The first two simulations ran on a single CPU and completed $20,000$ steps in 2 hours, while the third simulation completed the same number of steps in 5 hours due to the more complicated membrane-grain interaction. Overall, the results are very promising, as rock avalanches have been typically modeled as equivalent viscous fluid (see e.g. Setiasabda, 2020; Ho et al., 2021), rather than as assemblages of particles, in order to make the simulation tractable.

4.7 CONCLUSION

We demonstrated that the impulse-based DEM is attractive in large part that it alleviates the need to choose regular time intervals and saves substantial amount of force computation for complex shaped objects. The combined effects speed-up the simulation by at least factor of 10 with reasonable levels of fidelity. We modified ETM algorithm making it more suitable for dealing with a system with wide range of particle shapes and sizes. Instead of removing the smallest relative velocity from priority queue each time, we prioritize the weighted relative velocity which considers the masses of colliding bodies for better convergence. This measure is effective because it first considers larger objects, as later updates of small objects trivially influence the larger objects. Compared to SMM and SQM, ETM algorithm is advantageous that it is less sensitive to the order of

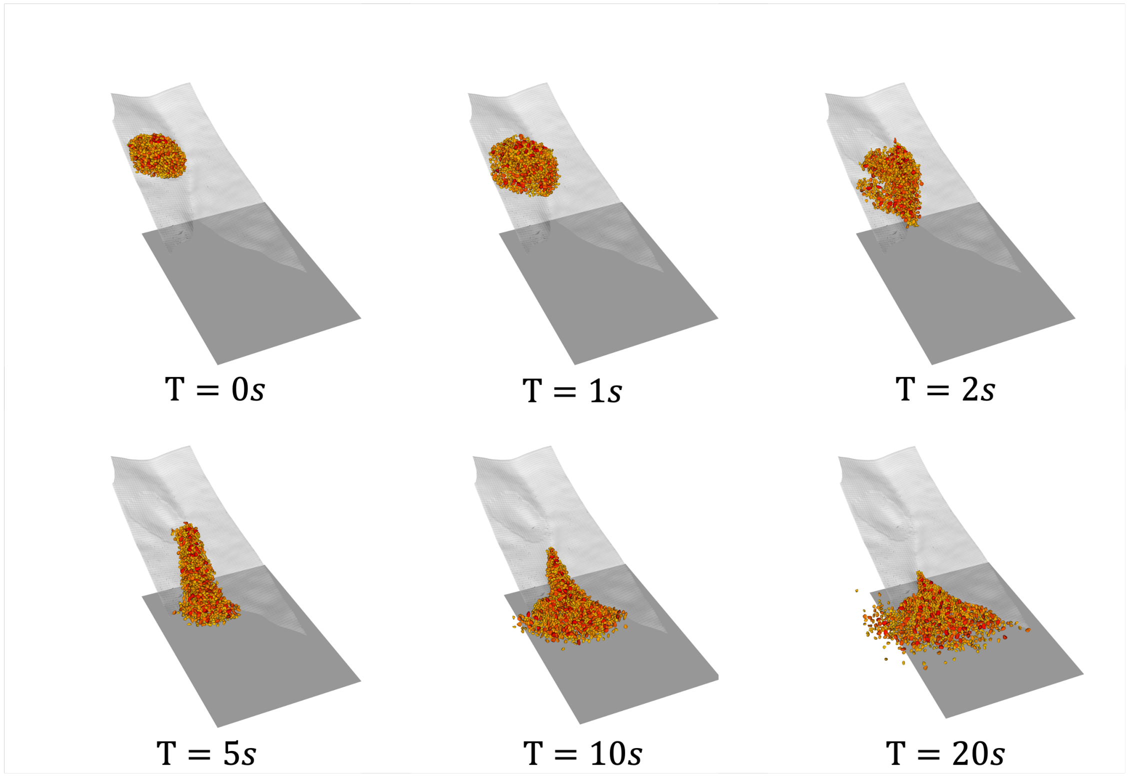

Figure 4.18: Rock avalanche runout onto a horizontal surface.

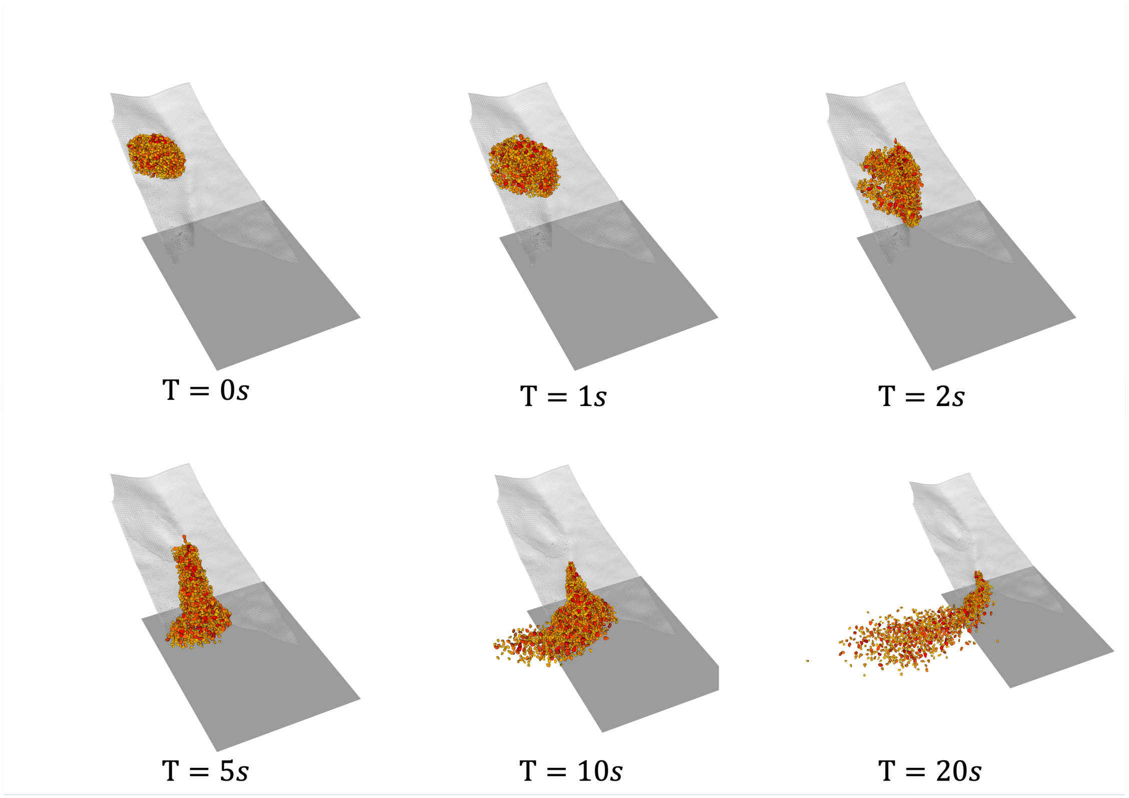

Figure 4.19: Rock avalanche runout onto a sloping surface.

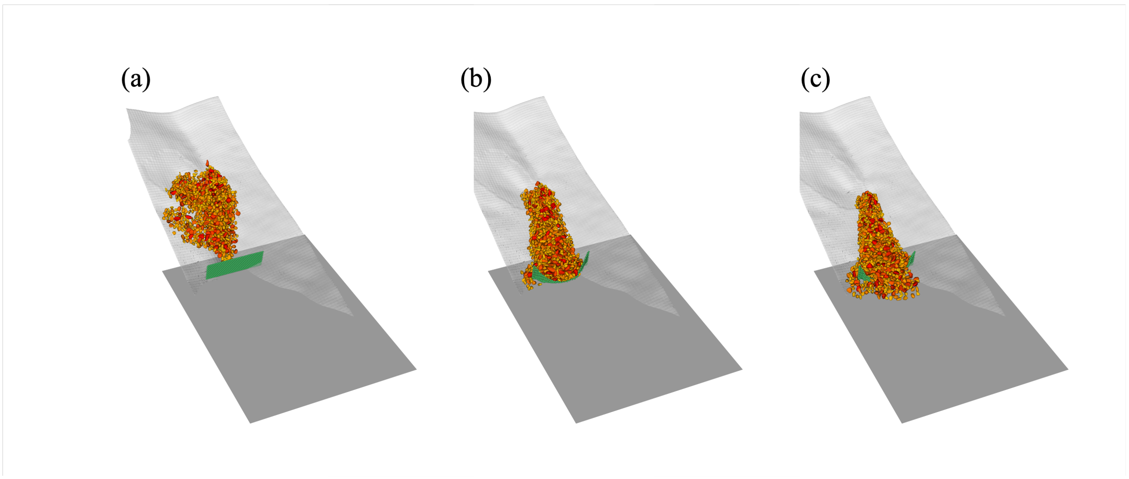

Figure 4.20: Rock avalanche impact on a single protection net.

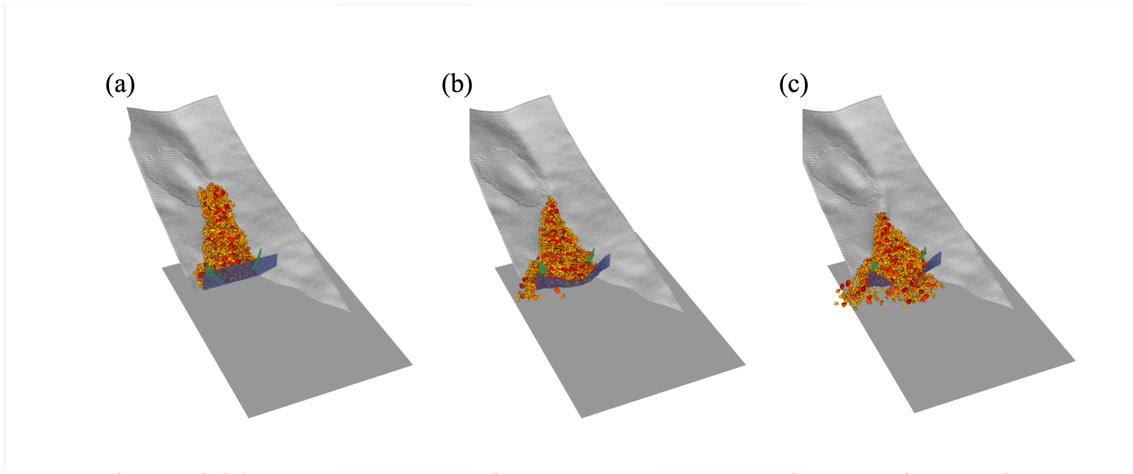

Figure 4.21: Rock avalanche impact on two protection nets in a series.

impulses and tends to be more energy conservative. By applying impulses gradually to the contact points, the impulses can influence multiple contact points at the same time and enable propagation of impulse effects and improve the overall ability of the method to capture propagation of forces during dynamics simulations. We further incorporated impulse-based method into LS framework. We also used a novel time integration scheme that separates the treatment of inter-grain collision and wall-grain resting contact. This improves the robustness of collision resolution algorithm, thereby avoiding numerical oscillation and unrealistic energy dissipation. Also, the demonstration of protection nets modeling provides convincing evidence that impulse-based methods can be used to simulate the dynamics of deformable structures.

The major limitation of the impulse-based method still lies in modeling a system of irregular, non-convex, non-uniform objects especially in a highly confined quasi-static setting, whereby objects with very different shapes and sizes interact with each other at many contact points and have small velocities. However, our modified ETM algorithm and time integration scheme is capable of simulating a range of very different objects and a range of dynamic phenomena. The current work provides a starting point for further enhancement and development of a more robust analysis tool to probe contact forces and impulses for wider range of problems. Finally, we examined the parallel performance of the impulse-based method using domain decomposition strategy, which showed that the current code was sped-up with few processors, but using more processors stalled and even deteriorated scalability due to severe workload imbalance and unaffordable communication overhead. This aspect of problem deserves further evaluation of different strategies and data abstraction in future work.

5 Conclusions and Future Work

The primary objective of the research presented herein was to use the results of detailed X-Ray Computed Tomography (XRCT) characterization of the fabric of a naturally deposited sand in order to build a high-fidelity micromechanical DEM model and then explore the effect of soil fabric on the macroscopic, mechanical behavior of the sand. Samples of fine sand obtained from a shoal in the San Francisco Bay scanned and tested in miniature triaxial tests by Garcia et al. (2022) provided the data used for the model development.

The principal challenge in being able to perform numerical experiments using micromechanical model of the sand was having an efficient DEM code. To this end a new parallel LS-DEM code was developed based on an existing framework. The new code uses the binning algorithm to reduce the computational complexity from $O(n^2)$ to $O(n)$. The code maps relationship between bins and grains with linked-list like data structure and considers MPI communication in two major parts: border/halo exchange and across-block migration. Numerical experiments show that the code has an excellent weak scalability and has the potential for simulating large scale DEM problems with complex-shaped grains. An important advantage of the MPI implementation is that the code is able to run on wide variety of parallel systems, including shared-memory computers and clustered systems, which makes the code highly portable. Additionally, a quick, efficient code enables systematic parameter exploration and full-scale simulation in other applications.

The new LS-DEM code was then used to replicate the triaxial compression tests performed on undisturbed sand samples and to demonstrate the feasibility of modeling complex natural fabric of sands. Both micro- and macromechanical behavior of natural materials were well-captured and validated with experimental data, which includes the initial stiffness, peak mobilized friction angle, force chain formation, shear band pattern and fabric evolution. The results of the numerical modeling show that the primary source of peak strength of sand is the mechanical interlocking between irregularly shaped grains; thus, even the simplest contact model is capable of reproducing both micro- and macro-mechanical responses consistent with experimental data, provided that the microstructure of the grain is captured with sufficient fidelity using high quality scans. Flexible membrane simulations with a pivoting loading platen were found to better predict the stress-strain and volumetric response, the onset and growth of strain localization, and accurately matched experimentally observed relationships between deviatoric stress and mobilized friction angle in naturally deposited sand.

Finally, we investigated the viability of modeling dynamic problems with newly formulated impulse-based LS-DEM. The new formulation has both numerical attractions and challenges.

While it is stable, fast, and energy conservative, it may be numerically stiff when the assembly has a substantial mass difference between particles or poorly reconstructed particles. We demonstrated the feasibility of modeling deformable structures in the rigid body framework and proposed several enhancements to improve the convergence of collision resolution, including a hybrid time integration scheme to separately handle at rest contacts and dynamic collisions. We then demonstrated that the LS-DEM is a generalizable framework which can accommodate any arbitrarily shaped topography and take algorithmic advantages for particle-topography interaction resolution. Specifically, the new formulation allows efficient modeling of granular flows such as rock avalanches with realistic geometry representation without having to assume equivalent fluid behavior as has been done in the past.

The new, parallel implementation of LS-DEM has potential for further enhancements for other applications. The most immediate may be adding capability for modeling particle fracture and disintegration in shear which frequently occurs in granular materials. Modeling heat generation in high velocity simulation of frictional materials is another potential enhancement for exploring phenomena such as rock avalanches. The parallel implementation of the impulse-based LS-DEM deserves additional exploration because the presence of contact islands fundamentally undermines the concept of domain decomposition and hence necessitates a new approach to domain partitioning and communication between processors. Probably the most significant future enhancement would be to integrate liquid phase into the LS-DEM framework and to examine the phase front using LS curve evolution techniques. Thus, the effect of capillary pressure drawing irregularly shaped particles together could be studied at the micro level. Modeling the third phase may be difficult since it requires the underlying mesh to keep track of the deformation of liquid and air phase. Both LS based front propagation and multi-phase flow are often evolved on Cartesian grids, which leads to the continual update of mesh surrounding the solid phase. The problem might be resolved with the immersed boundary method, which incorporates the solid-fluid interaction as an additional force component in the Navier-Stokes equation and imposes slip and velocity compatibility as boundary conditions.

The methodologies presented herein are not tied to granular materials or any one particular stress path, and they open a door to simulation of other families of materials, which can fracture, deform, or flow under more complex loading conditions. The parallel LS-DEM is promising in that it can capture object morphology at highest level and is backward compatible, allowing researchers to choose the appropriate geometry precision, model scale, and dynamic type for the problem at hand.

REFERENCES

Abriak, N. and Caron, J.-F. (2006). “Experimental study of shear in granular media.” *Advanced Powder Technology*, 17(3), 297–318. https://doi.org/10.1163/156855206777213348.

Amirrahmat, S., Druckrey, A. M., Alshibli, K. A., and Al-Raoush, R. I. (2019). “Micro shear bands: Precursor for strain localization in sheared granular materials.” *Journal of Geotechnical and Geoenvironmental Engineering*, 145(2). https://doi.org/10.1061/(ASCE)GT.1943-5606.0001989.

Amritkar, A., Deb, S., and Tafti, D. (2014). “Efficient parallel cfd-dem simulations using openmp.” *Journal of Computational Physics*, 256, 501–519. https://doi.org/10.1016/j.jcp.2013.09.007.

Asai, M., Li, Y., Chandra, B., and Takase, S. (2021). “Fluid–rigid-body interaction simulations and validations using a coupled stabilized isph–dem incorporated with the energy-tracking impulse method for multiple-body contacts.” *Computer Methods in Applied Mechanics and Engineering*, 377, 113681. https://doi.org/10.1016/j.cma.2021.113681.

Baraff, D. (1989). “Analytical methods for dynamic simulation of non-penetrating rigid bodies.” *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, 223–232. https://doi.org/10.1145/74334.74356.

Barzel, R. and Barr, A. H. (1988). “A modeling system based on dynamic constraints.” *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 179–188. https://doi.org/10.1145/54852.378509.

Baugh Jr, J. W. and Konduri, R. (2001). “Discrete element modelling on a cluster of workstations.” *Engineering with Computers*, 17(1), 1–15. https://doi.org/10.1007/PL00007192.

Bender, J. (2007). “Impulse-based dynamic simulation in linear time.” *Computer Animation and Virtual Worlds*, 18(4-5), 225–233. https://doi.org/10.1002/cav.179.

Chang, B. and Colgate, J. E. (1997). “Real-time impulse-based simulation of rigid body systems for haptic display.” *Proceedings of the 1997 ASME international mechanical engineering congress and exhibition*, ASME, 145–152. https://doi.org/10.1115/IMECE1997-0389.

Cho, N. a., Martin, C., and Sego, D. (2007). “A clumped particle model for rock.” *International journal of rock mechanics and mining sciences*, 44(7), 997–1010. https://doi.org/10.1016/j.ijrmms.2007.02.002.

Chorley, M. J. and Walker, D. W. (2010). “Performance analysis of a hybrid mpi/openmp application on multi-core clusters.” *Journal of Computational Science*, 1(3), 168–174. https://doi.org/10.1016/j.jocs.2010.05.001.

Christoffersen, J., Mehrabadi, M. M., and Nemat-Nasser, S. (1981). “A micromechanical description of granular material behavior.” *J. Appl. Mech.*, 48(2), 339–344. https://doi.org/10.1115/1.3157619.

Cui, L. and O’Sullivan, C. (2005). “Development of a mixed boundary environment for axi-symmetric dem analyses.” *Proc. 5th Int. Conf. on Micromechanics of Granular Media, Stuttgart*, Vol. 1, 301–305. https://doi.org/10.1201/NOE0415383486.

Cundall, P. A. and Strack, O. D. (1979). “A discrete numerical model for granular assemblies.” *Geotechnique*, 29(1), 47–65. https://doi.org/10.1680/geot.1979.29.1.47.

Cuthill, E. and McKee, J. (1969). “Reducing the bandwidth of sparse symmetric matrices.” *Proceedings of the 1969 24th national conference*, 157–172. https://doi.org/10.1145/800195.805928.

Da Cruz, F., Emam, S., Prochnow, M., Roux, J.-N., and Chevoir, F. (2005). “Rheophysics of dense granular materials: Discrete simulation of plane shear flows.” *Physical Review E*, 72(2), 021309. https://doi.org/10.1103/PhysRevE.72.021309.

De Bono, J., McDowell, G., and Wanatowski, D. (2012). “Discrete element modelling of a flexible membrane for triaxial testing of granular material at high pressures.” *Géotechnique Letters*, 2(4), 199–203. https://doi.org/10.1680/geolett.12.00040.

Evans, D. J. and Murad, S. (1977). “Singularity free algorithm for molecular dynamics simulation of rigid polyatomics.” *Molecular physics*, 34(2), 327–331. https://doi.org/10.1080/00268977700101761.

Fakhimi, A. and Villegas, T. (2007). “Application of dimensional analysis in calibration of a discrete element model for rock deformation and fracture.” *Rock Mechanics and Rock Engineering*, 40(2), 193–211. https://doi.org/10.1680/geolett.12.00040.

Garboczi, E. J. (2002). “Three-dimensional mathematical analysis of particle shape using x-ray tomography and spherical harmonics: Application to aggregates used in concrete.” *Cement and concrete research*, 32(10), 1621–1638. https://doi.org/10.1016/S0008-8846(02)00836-0.

Garcia, E., Ando, E., Viggiani, G., and Sitar, N. (2022). “Influence of depositional fabric on mechanical properties of naturally deposited sands.” *Geotechnique*, 1–15. https://doi.org/10.1680/jgeot.21.00230.

Garcia, X., Latham, J.-P., XIANG, J.-s., and Harrison, J. (2009). “A clustered overlapping sphere algorithm to represent real particles in discrete element modelling.” *Geotechnique*, 59(9), 779–784. https://doi.org/10.1680/geot.8.T.037.

Gopalakrishnan, P. and Tafti, D. (2013). “Development of parallel dem for the open source code mfix.” *Powder technology*, 235, 33–41. https://doi.org/10.1016/j.powtec.2012.09.006.

Gray, A. and Moore, A. (2000). “`n-body'problems in statistical learning.” *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp, eds., Vol. 13, MIT Press, 1–7, https://proceedings.neurips.cc/paper/2000/file/7385db9a3f11415bc0e9e2625fae3734-Paper.pdf.

Grest, G. S., Dünweg, B., and Kremer, K. (1989). "Vectorized link cell fortran code for molecular dynamics simulations for a large number of particles." *Computer Physics Communications*, 55(3), 269–285. https://doi.org/10.1016/0010-4655(89)90125-2.

Guendelman, E., Bridson, R., and Fedkiw, R. (2003). "Nonconvex rigid bodies with stacking." *ACM transactions on graphics (TOG)*, 22(3), 871–878. https://doi.org/10.1145/882262.882358.

Guo, P. (2012). "Critical length of force chains and shear band thickness in dense granular materials." *Acta Geotechnica*, 7(1), 41–55. https://doi.org/10.1007/s11440-011-0154-3.

Haier, E., Lubich, C., and Wanner, G. (2006). *Geometric Numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer.

Hazzar, L., Nuth, M., and Chekired, M. (2020). "Dem simulation of drained triaxial tests for glass-beads." *Powder Technology*, 364, 123–134. https://doi.org/10.1016/j.powtec.2019.09.095.

Hazzard, J. F., Collins, D. S., Pettitt, W. S., and Young, R. P. (2002). "Simulation of unstable fault slip in granite using a bonded-particle model." *The mechanism of induced seismicity*, Springer, 221–245. https://doi.org/10.1007/978-3-0348-8179-1_11.

Henty, D. S. (2000). "Performance of hybrid message-passing and shared-memory parallelism for discrete element modeling." *SC'00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, IEEE, 10–10. https://doi.org/10.1109/SC.2000.10005.

Ho, K. K., Koo, R. C., and Kwan, J. S. (2021). "Mitigation of debris flows—research and practice in hong kong." *Environmental & Engineering Geoscience*, 27(2), 231–243. https://doi.org/10.2113/EEG-D-20-00009.

Ishihara, K. (1993). "Liquefaction and flow failure during earthquakes." *Geotechnique*, 43(3), 351–451. https://doi.org/10.1680/geot.1993.43.3.351.

Itasca (1998). *2.00 Particle Flow Code in Two Dimensions, PFC2D*. Itasca, PFC2D, Minneapolis Minnesota.

Itasca (2004). *PFC2D (Particle Flow Code in Two Dimensions)*. Itasca, CGI, Minneapolis, Minnesota, USA.

Jagadish, H. V., Ooi, B. C., Tan, K.-L., Yu, C., and Zhang, R. (2005). "idistance: An adaptive b+-tree based indexing method for nearest neighbor search." *ACM Transactions on Database Systems (TODS)*, 30(2), 364–397. https://doi.org/10.1145/1071610.1071612.

Kačianauskas, R., Maknickas, A., Kačeniauskas, A., Markauskas, D., and Balevičius, R. (2010). "Parallel discrete element simulation of poly-dispersed granular material." *Advances in Engineering Software*, 41(1), 52–63. https://doi.org/10.1016/j.advengsoft.2008.12.004.

Karp, A. H. and Flatt, H. P. (1990). "Measuring parallel processor performance." *Communications of the ACM*, 33(5), 539–543. https://doi.org/10.1145/78607.78614.

Kawamoto, R., Andò, E., Viggiani, G., and Andrade, J. E. (2016). "Level set discrete element method for three-dimensional computations with triaxial case study." *Journal of the Mechanics and Physics of Solids*, 91, 1–13. https://doi.org/10.1016/j.jmps.2016.02.021.

Kawamoto, R., Andò, E., Viggiani, G., and Andrade, J. E. (2018). "All you need is shape: predicting shear banding in sand with ls-dem." *Journal of the Mechanics and Physics of Solids*, 111, 375–392. https://doi.org/10.1016/j.jmps.2017.10.003.

Kawamoto, R. Y. (2018). "The avatar paradigm in granular materials." Ph.D. thesis, California Institute of Technology, Pasadena, CA.

Kim, H., Han, J., and Han, T. Y.-J. (2020). "Machine vision-driven automatic recognition of particle size and morphology in sem images." *Nanoscale*, 12(37), 19461–19469. https://doi.org/10.1039/D0NR04140H.

Lam, W.-K. and Tatsuoka, F. (1988). "Triaxial compressive and extension strength of sand affected by strength anisotropy and sample slenderness." *Advanced triaxial testing of soil and rock*, ASTM International, 655–666. https://doi.org/10.1520/STP29105S.

Lee, S. J. and Hashash, Y. M. (2015). "idem: An impulse-based discrete element method for fast granular dynamics." *International Journal for Numerical Methods in Engineering*, 104(2), 79–103. https://doi.org/10.1002/nme.4923.

Lee, S. J., Hashash, Y. M., and Nezami, E. G. (2012). "Simulation of triaxial compression tests with polyhedral discrete elements." *Computers and Geotechnics*, 43, 92–100. https://doi.org/10.1016/j.compgeo.2012.02.011.

Li, C., Xu, C., Gui, C., and Fox, M. D. (2005). "Level set evolution without re-initialization: a new variational formulation." *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1, IEEE, 430–436. https://doi.org/10.1109/CVPR.2005.213.

Li, Y., Asai, M., Chandra, B., and Isshiki, M. (2021). "Energy-tracking impulse method for particle-discretized rigid-body simulations with frictional contact." *Computational Particle Mechanics*, 8(2), 237–258. https://doi.org/10.1007/s40571-020-00326-5.

Liang, W. and Zhao, J. (2019). "Multiscale modeling of large deformation in geomechanics." *International Journal for Numerical and Analytical Methods in Geomechanics*, 43(5), 1080–1114. https://doi.org/10.1007/s00603-012-0281-7.

Lim, K.-W. and Andrade, J. E. (2014). "Granular element method for three-dimensional discrete element calculations." *International Journal for Numerical and Analytical Methods in Geomechanics*, 38(2), 167–188. https://doi.org/10.1002/nag.2203.

Lim, K.-W., Krabbenhoft, K., and Andrade, J. E. (2014). "A contact dynamics approach to the granular element method." *Computer Methods in Applied Mechanics and Engineering*, 268, 557–573. https://doi.org/10.1016/j.cma.2013.10.004.

Maknickas, A., Kačeniauskas, A., Kačianauskas, R., Balevičius, R., and Džiugys, A. (2006). "Parallel dem software for simulation of granular media." *Informatica*, 17(2), 207–224. https://doi.org/10.15388/INFORMATICA.2006.134.

McCallen, D., Petersson, A., Rodgers, A., Pitarka, A., Miah, M., Petrone, F., Sjogreen, B., Abrahamson, N., and Tang, H. (2021). "Eqsim—a multidisciplinary framework for fault-to-structure earthquake simulations on exascale computers part i: Computational models and workflow." *Earthquake Spectra*, 37(2), 707–735. 10.1177/8755293020970982.

MiDi, G. (2004). "On dense granular flows." *The European Physical Journal E*, 14, 341–365. https://doi.org/10.1140/epje/i2003-10153-0.

Mirtich, B. (1996). "Impulse-based dynamic simulation of rigid body systems." Ph.D. thesis, University of California, Berkeley, CA.

Mirtich, B. and Canny, J. (1995). "Impulse-based simulation of rigid bodies." *Proceedings of the 1995 symposium on Interactive 3D graphics*, 181–188. https://doi.org/10.1145/199404.199436.

Mitchell, J. K. and Soga, K. (2005). *Fundamentals of soil behavior*, Vol. 3. John Wiley & Sons New York.

Mollon, G., Quacquarelli, A., Andò, E., and Viggiani, G. (2020). "Can friction replace roughness in the numerical simulation of granular materials?." *Granular Matter*, 22(2), 1–16. https://doi.org/10.1007/s10035-020-1004-5.

Mollon, G. and Zhao, J. (2012). "Fourier–voronoi-based generation of realistic samples for discrete modelling of granular materials." *Granular matter*, 14(5), 621–638. https://doi.org/10.1007/s10035-012-0356-x.

Moore, M. and Wilhelms, J. (1988). "Collision detection and response for computer animation." *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 289–298. https://doi.org/10.1145/378456.378528.

Muja, M. and Lowe, D. G. (2009). "Fast approximate nearest neighbors with automatic algorithm configuration.." *VISAPP (1)*, 2(331-340), 2. https://doi.org/10.5220/0001787803310340.

Mulilis, J. P., Seed, H. B., Chan, C. K., Mitchell, J. K., and Arulanandan, K. (1977). "Effects of sample preparation on sand liquefaction." *Journal of the Geotechnical Engineering Division*, 103(2), 91–108. https://doi.org/10.1061/AJGEB6.0000387.

Munjiza, A. and Andrews, K. (1998). "Nbs contact detection algorithm for bodies of similar size." *International Journal for Numerical Methods in Engineering*, 43(1), 131–149. https://doi.org/10.1002/(SICI)1097-0207(19980915)43:1¡131::AID-NME447¿3.0.CO;2-S.

Ng, T.-T. (2006). "Input parameters of discrete element methods." *Journal of Engineering Mechanics*, 132(7), 723–729. https://doi.org/10.1061/(ASCE)0733-9399(2006)132:7(723).

Nouguier-Lehon, C., Cambou, B., and Vincens, E. (2003). “Influence of particle shape and angularity on the behaviour of granular materials: a numerical analysis.” *International journal for numerical and analytical methods in geomechanics*, 27(14), 1207–1226. https://doi.org/10.1002/nag.314.

Ochiai, H. and Lade, P. V. (1983). “Three-dimensional behavior of sand with anisotropic fabric.” *Journal of Geotechnical Engineering*, 109(10), 1313–1328. http://dx.doi.org/10.1061/(ASCE)0733-9410(1983)109:10(1313).

Oda, M. (1972). “The mechanism of fabric changes during compressional deformation of sand.” *Soils and foundations*, 12(2), 1–18. https://doi.org/10.3208/sandf1972.12.1.

Osher, S. and Sethian, J. A. (1988). “Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations.” *Journal of computational physics*, 79(1), 12–49. https://doi.org/10.1016/0021-9991(88)90002-2.

O’Sullivan, C. and Bray, J. D. (2004). “Selecting a suitable time step for discrete element simulations that use the central difference time integration scheme.” *Engineering Computations*, 21(2/3/4), 278–303. https://doi.org/10.1108/02644400410519794.

Owen, D. and Feng, Y. (2001). “Parallelised finite/discrete element simulation of multi-fracturing solids and discrete systems.” *Engineering computations*, 18(3/4), 557–576. https://doi.org/10.1108/02644400110387154.

Park, J.-W. and Song, J.-J. (2009). “Numerical simulation of a direct shear test on a rock joint using a bonded-particle model.” *International Journal of Rock Mechanics and Mining Sciences*, 46(8), 1315–1328. https://doi.org/10.1016/j.ijrmms.2009.03.007.

Peters, J. F., Hopkins, M. A., Kala, R., and Wahl, R. E. (2009). “A poly-ellipsoid particle for non-spherical discrete element method.” *Engineering Computations*, 26(6), 645–657. https://doi.org/10.1108/02644400910975441.

Potyondy, D. O. and Cundall, P. (2004). “A bonded-particle model for rock.” *International journal of rock mechanics and mining sciences*, 41(8), 1329–1364. https://doi.org/10.1016/j.ijrmms.2004.09.011.

Rice, J. (1976). “The localization of plastic deformation.” *Proc. of the 14th IUTAM Congress, Delft*, W. Koiter, ed., North-Holland Publishing Co., 207–220.

Rodgers, D. P. (1985). “Improvements in multiprocessor system design.” *ACM SIGARCH Computer Architecture News*, 13(3), 225–231. https://doi.org/10.1145/327070.327215.

Rorato, R., Arroyo, M., Gens, A., Andò, E., and Viggiani, G. (2021). “Image-based calibration of rolling resistance in discrete element models of sand.” *Computers and Geotechnics*, 131, 103929. http://dx.doi.org/10.1016/j.compgeo.2020.103929.

Roscoe, K. H. (1970). “The influence of strains in soil mechanics.” *Geotechnique*, 20(2), 129–170. https://doi.org/10.1680/geot.1970.20.2.129.

Rudnicki, J. W. and Rice, J. (1975). "Conditions for the localization of deformation in pressure-sensitive dilatant materials." *Journal of the Mechanics and Physics of Solids*, 23(6), 371–394. https://doi.org/10.1016/0022-5096(75)90001-0.

Salgado, R., Bandini, P., and Karim, A. (2000). "Shear strength and stiffness of silty sand." *Journal of geotechnical and geoenvironmental engineering*, 126(5), 451–462. https://doi.org/10.1061/(ASCE)1090-0241(2000)126:5(451).

Schanz, T. and Vermeer, P. (1996). "Angles of friction and dilatancy of sand." *Geotechnique*, 46(1), 145–151. https://doi.org/10.1680/geot.1996.46.1.145.

Sethian, J. A. (1996). "A fast marching level set method for monotonically advancing fronts." *Proceedings of the National Academy of Sciences*, 93(4), 1591–1595. https://doi.org/10.1073/pnas.93.4.1591.

Sethian, J. A. (1999). "Fast marching methods." *SIAM review*, 41(2), 199–235. https://doi.org/10.1137/S0036144598347059.

Setiasabda, E. Y. (2020). "Material point method for large deformation modeling in geomechanics using isoparametric elements." Ph.D. thesis, University of California, Berkeley, CA.

Stronge, W. J. (2018). *Impact mechanics*. Cambridge university press.

Tamadondar, M. R., de Martín, L., and Rasmuson, A. (2019). "Agglomerate breakage and adhesion upon impact with complex-shaped particles." *AIChE Journal*, 65(6), e16581. https://doi.org/10.1002/aic.16581.

Tan, P. (2022). "Numerical modeling of soil fabric of naturally deposited sand." Ph.D. thesis, University of California, Berkeley, CA.

Tang, X., Paluszny, A., and Zimmerman, R. W. (2014). "An impulse-based energy tracking method for collision resolution." *Computer Methods in Applied Mechanics and Engineering*, 278, 160–185. https://doi.org/10.1016/j.cma.2014.05.004.

Taylor, M. A., Garboczi, E., Erdogan, S., and Fowler, D. (2006). "Some properties of irregular 3-d particles." *Powder Technology*, 162(1), 1–15. https://doi.org/10.1016/j.powtec.2005.10.013.

Terzaghi, K. (1955). "Influence of geological factors on the engineering properties of sediments." *50th Anniversary Volume: 1905-1955*, Society of Economic Geologists, 557–618. https://doi.org/10.5382/AV50.15.

Thornton, C. (2000). "Numerical simulations of deviatoric shear deformation of granular media." *Géotechnique*, 50(1), 43–53. https://doi.org/10.1680/geot.2000.50.1.43.

Vaid, Y. P., Sivathayalan, S., and Stedman, D. (1999). "Influence of specimen-reconstituting method on the undrained response of sand." *Geotechnical Testing Journal*, 22(3), 187–195. https://doi.org/10.1520/GTJ11110J.

Vlahinić, I., Andrade, J., Andö, E., and Viggiani, G. (2013). “From 3d tomography to physics-based mechanics of geomaterials.” *Computing in Civil Engineering (2013)*, ASCE, 339–345. https://doi.org/10.1061/9780784413029.043.

Walther, J. H. and Sbalzarini, I. F. (2009). “Large-scale parallel discrete element simulations of granular flow.” *Engineering Computations*, 26(6), 688–697. https://doi.org/10.1108/02644400910975478.

Walton, O. and Braun, R. (1993). “Simulation of rotary-drum and repose tests for frictional spheres and rigid sphere clusters.” *Report No. UCRL-JC-115749*, Lawrence Livermore National Lab., CA, https://www.osti.gov/biblio/10132451.

Washington, D. W. and Meegoda, J. N. (2003). “Micro-mechanical simulation of geotechnical problems using massively parallel computers.” *International journal for numerical and analytical methods in geomechanics*, 27(14), 1227–1234. https://doi.org/10.1002/nag.317.

Wiebicke, M., Andò, E., Viggiani, G., and Herle, I. (2020). “Measuring the evolution of contact fabric in shear bands with x-ray tomography.” *Acta Geotechnica*, 15(1), 79–93. https://doi.org/10.1007/s11440-019-00869-9.

Williams, J. R., Perkins, E., and Cook, B. (2004). “A contact algorithm for partitioning n arbitrary sized objects.” *Engineering Computations*, 21(2/3/4), 235–248. https://doi.org/10.1108/02644400410519767.

Wood, D. and Maeda, K. (2008). “Changing grading of soil: effect on critical states.” *Acta Geotechnica*, 3(1), 3–14. https://doi.org/10.1007/s11440-007-0041-0.

Wriggers, P. and Laursen, T. A. (2006). *Computational contact mechanics*, Vol. 2. Springer.

Wu, M., Wang, J., Russell, A., and Cheng, Z. (2021). “Dem modelling of mini-triaxial test based on one-to-one mapping of sand particles.” *Géotechnique*, 71(8), 714–727. https://doi.org/10.1680/jgeot.19.P.212.

Yan, B. and Regueiro, R. (2018a). “Comparison between o (n2) and o (n) neighbor search algorithm and its influence on superlinear speedup in parallel discrete element method (dem) for complex-shaped particles.” *Engineering Computations*, 35(3). https://doi.org/10.1108/EC-01-2018-0023.

Yan, B. and Regueiro, R. A. (2018b). “A comprehensive study of mpi parallelism in three-dimensional discrete element method (dem) simulation of complex-shaped granular particles.” *Computational Particle Mechanics*, 5(4), 553–577. https://doi.org/10.1007/s40571-018-0190-y.

Yan, B. and Regueiro, R. A. (2019). “Comparison between pure mpi and hybrid mpi-openmp parallelism for discrete element method (dem) of ellipsoidal and poly-ellipsoidal particles.” *Computational Particle Mechanics*, 6(2), 271–295. https://doi.org/10.1007/s40571-018-0213-8.

Yoshimine, M., Ishihara, K., and Vargas, W. (1998). “Effects of principal stress direction and intermediate principal stress on undrained shear behavior of sand.” *Soils and Foundations*, 38(3), 179–188. https://doi.org/10.3208/sandf.38.3_179.

Zhao, D., Nezami, E. G., Hashash, Y. M., and Ghaboussi, J. (2006). “Three-dimensional discrete element simulation for granular materials.” *Engineering Computations*, 23(7), 749–770. https://doi.org/10.1108/02644400610689884.

Zhao, S. and Zhao, J. (2019). “A poly-superellipsoid-based approach on particle morphology for dem modeling of granular media.” *International Journal for Numerical and Analytical Methods in Geomechanics*, 43(13), 2147–2169. https://doi.org/10.1002/nag.2951.

Zhao, S. and Zhao, J. (2021). “Sudodem: Unleashing the predictive power of the discrete element method on simulation for non-spherical granular particles.” *Computer Physics Communications*, 259. https://doi.org/10.1016/j.cpc.2020.107670.

Zheng, J., An, X., and Huang, M. (2012). “Gpu-based parallel algorithm for particle contact detection and its application in self-compacting concrete flow simulations.” *Computers & Structures*, 112, 193–204. https://doi.org/10.1016/j.compstruc.2012.08.003.

Zhou, B., Wang, J., and Zhao, B. (2015). “Micromorphology characterization and reconstruction of sand particles using micro x-ray tomography and spherical harmonics.” *Engineering geology*, 184, 126–137. https://doi.org/10.1016/j.enggeo.2014.11.009.

Zlatovic, S. and Ishihara, K. (1997). “Normalized behavior of very loose non-plastic soils: effects of fabric.” *Soils and foundations*, 37(4), 47–56. https://doi.org/10.3208/sandf.37.4_47.

Zohdi, T. I. (2003). “Genetic design of solids possessing a random–particulate microstructure.” *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1806), 1021–1043. https://doi.org/10.1098/rsta.2003.1179.

Zohdi, T. I. (2004a). “A computational framework for agglomeration in thermochemically reacting granular flows.” *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 460(2052), 3421–3445. https://doi.org/10.1098/rspa.2004.1277.

Zohdi, T. I. (2004b). “Staggering error control of a class of inelastic processes in random microheterogeneous solids.” *International journal of non-linear mechanics*, 39(2), 281–297. https://doi.org/10.1016/S0020-7462(02)00188-9.

Zohdi, T. I. (2007). “Particle collision and adhesion under the influence of near-fields.” *Journal of Mechanics of Materials and Structures*, 2(6), 1011–1018. http://dx.doi.org/10.2140/jomms.2007.2.1011.

Zohdi, T. I. (2010). “Simulation of coupled microscale multiphysical-fields in particulate-doped dielectrics with staggered adaptive fdtd.” *Computer Methods in Applied Mechanics and Engineering*, 199(49-52), 3250–3269. https://doi.org/10.1016/j.cma.2010.06.032.

Zohdi, T. I. (2012). “Estimation of electrical heating load-shares for sintering of powder mixtures.” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 468(2144), 2174–2190. https://doi.org/10.1098/rspa.2011.0755.

Zohdi, T. I. (2013). “Numerical simulation of charged particulate cluster-droplet impact on electrified surfaces.” *J Comput Phys*, 233, 509–526. http://dx.doi.org/10.1016/j.jcp.2012.09.012.

Zohdi, T. I. (2014). "Impact and penetration resistance of network models of coated lightweight fabric shielding." *GAMM-Mitteilungen*, 37(1), 124–150. https://doi.org/10.1002/gamm.201410006.

Zohdi, T. I. (2017). *Modeling and simulation of functionalized materials for additive manufacturing and 3d printing: continuous and discrete media: continuum and discrete element methods*, Vol. 60. Springer.

Zohdi, T. I. and Wriggers, P. (1999). "A domain decomposition method for bodies with heterogeneous microstructure basedon material regularization." *International Journal of Solids and Structures*, 36(17), 2507–2525. https://doi.org/10.1016/S0020-7683(98)00124-3.

Zohdi, T. I. and Wriggers, P. (2004). *An introduction to computational micromechanics*, Vol. 20. Springer Science & Business Media.

ISSN 2770-8314
https://doi.org/10.55461/KMIZ5819