Contact Investigations of Granular Mechanical Media in a Tumbling Sorting Machine

Von der Fakultät Maschinenbau der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

by

Hashem Alkhaldi

geboren in Beirut (Libanon)

Hauptberichter:	Prof. Dr.–Ing. P. Eberhard
Mitberichter:	Prof. Dr. O. Abuzeid

Tag der Einreichung:	13. April 2007
Tag der mündlichen Prüfung:	19. June 2007

Institut für Technische und Numerische Mechanik
Universität Stuttgart

2007

Acknowledgements

It gives me great pleasure to take this opportunity to acknowledge my indebtedness to all those who have helped me in one way or another in completing the present work.

First and foremost, I would like to express my deepest thanks to my supervisor, Prof. Dr.-Ing. P. Eberhard, for his help, enthusiastic support, continuous encouragement and guidance during my research work. I am grateful to him for providing the opportunity to do this research under his supervision and his excellent monitoring of the study through numerous discussions and useful advice, without which this project would never have come to fruition. Furthermore, he is gratefully acknowledged for the real comfortable working environment and computer availability. I also want to thank Prof. Dr. O. Abuzeid for being the second reviewer of the thesis.

I owe special thanks to Prof. Dr.-Ing. W. Schiehlen for the helpful and enjoyable discussions I had with him. Reviewing always my annual reports and supporting me by DAAD is highly appreciated and gratefully acknowledged. All love and gratitude for our secretary Mrs. R. Prommersberger. They all provided me with a family-like atmosphere and gave me warm-hearted help whenever I needed.

I am greatly grateful to my present and former colleagues, particularly Dr.-Ing. A. Eiber, B. Muth, P. Ziegler, S. Ebrahimi, M. Ackermann and F. Fleißner, for their helpful and fruitful discussions, and for good time we spend together. Furthermore, I would like to to express my sincere thanks to all other colleagues at the ITM (Institute of Engineering and Computational Mechanics) who in one way or another helped me to accomplish this work.

I am greatly indebted to DAAD (German Academic Exchange Service) for their financial support during my Ph.D. work in Germany, and I would like to express my deepest appreciation and thanks to all members of DAAD for their assistance, advice and support. Many thanks are also due to Dr.-Ing. K. Friedrich from Allgaier-Werke GmbH, Uhingen for his assistance and cooperation.

Finally, I thank my wife for her continuous support and patience during the years I spent on my Ph.D. work and my mother for her encouragement and continuous support.

Stuttgart, April 2007 Hashem Alkhaldi

To my mother, my wife *Shahnaz*,
and my children *Aous*, *Rimaz* & *Rayan*
with love and gratitude

Contents

Zusammenfassung

Aufgrund ihrer weitverbreiteten Anwendung in industriellen und technologischen Prozessen sind granulare Medien als Inhalt dieser Arbeit von grossem wissenschaftlichen Interesse. Ein Scherpunkt dieser Arbeit ist, einen allgemeinen Überblick über einige bereits existierende Methoden zur Simulation granularer Medien zu geben. Die Verkürzung von Rechenzeiten durch die Verwendung paralleler Berechnungsansätze ist ein weiteres Thema dieser Arbeit. Die Erweiterung bestehender Algorithmen, die Parallelisierung eines bestehenden Simulationsprogramms, sowie die Untersuchung und Analyse einer realen industriellen Anwendung zum Sieben von Partikeln mit einer Taumelsiebmaschine mit verschiedenen Parameterkonfigurationen werden in dieser Arbeit vorgestellt. Diese Ziele werden durch einen guten Überblick über die dynamischen Betriebsparameter erreicht, welche die Effizienz der Taumelsiebmaschinen, d.h. den Trennprozess, beeinflussen.

Zu diesem Zweck gibt Kapitel 1 eine kurze Einführung in die in granularen Medien auftretenden Kontaktprobleme zusammen mit Erläuterungen zu einigen numerischen Algorithmen die in der sequentiellen und parallelen Simulation Verwendung finden. In diesem Kapitel wird ebenfalls ein allgemeiner und kurzer Überblick über einige Studien und die Einflussparameter, die die Partikelsiebtechnologie beeinflussen, gewährt.

In Kapitel 2 wird eine allgemeine Beschreibung der Probleme der Molekulardynamik gegeben und es werden die Grundeigenschaften granularer Medien erläutert. Einige der am häufigsten verwendeten Algorithmen und Modelle, z.B. die Diskrete Element Methode (DEM) und die Penalty-Methode mit Feder-Dämpfersystemen, werden in diesem Kapitel einführt. Verschiedene Strategien serieller und paralleler Kollisionserkennung und Kontaktkraftberechnung werden entwickelt, implementiert und untersucht. In diesem Kapitel werden auch die bei normaler und tangentialer Kollision auftretenden Effekte, sowie der Einfluss von Dämpfung und Adhesion auf die kollidierenden, runden Partikel erklärt. Einige Ansätze zur Beschleunigung der Simulation von Partikelsystemen durch die Sortierung mit geeigneten Sortierverfahren und die Nachbarschaftssuche mit Listen, z.B. der Verlet-Ansatz und verkettete lineare Listen werden verwendet und verglichen. Verschiedene Integrationsverfahren werden ebenfalls erläutert. Es wird beobachtet, dass Verlet-Integratoren effizient und genau sind und damit geeignet sind zur Integration der Bewegungsgleichungen granularer Systeme. Das Kapitel endet mit einer Beschreibung

der Struktur des verwendeten seriellen Molekulardynamikprograms MOLDYN und der das Simulationsproblem beschreibenden Eingabefiles.

Da themenverwandte Untersuchungen oft auf numerischer Simulation basieren, ist die Untersuchung numerischer Phänomene großer granularer Systeme mit Personalcomputern mit einer CPU eine Herausforderung. In Kapitel 3 wird die räumliche Gebietsunterteilung zur Erzeugung des parallelen Codes verwendet. Diese Methode gewährleistet Skalierbarkeit und gute Resultate besonders in Verbindung mit Lastverteilung. Eine Voraussetzung für den Erfolg numerischer Berechnungen ist der Zugang zu Computersystemen, die mächtig genug sind, um das zu untersuchende Problem zu berechnen. In diesem Kapitel werden bestehende sequentielle Algorithmen erweitert und verändert, um sie an moderne Hochleistungsrechner anzupassen. Die Bibliotheksfunktionen der Parallel Virtual Machine (PVM) werden für die Kommunikation zwischen Prozessoren in einem System mit verteiltem Speicher verwendet. Dieses Kapitel verdeutlicht auch den Zusammenhang zwischen dem Speedup, dem häufig verwendeten Messwert für Programmskalierbarkeit und der größe des Systems. Es wird beobachtet, dass die Leistung der Simulation sich mit steigender Partikelzahl verbessert. Der Grund liegt im Kommunikations- und Datenfluss, der bei steigender Partikelzahl effizienter wird. In einigen Fällen wurde superlineares Speedupverhalten beobachtet, was auf Cache-Effekte der einzelnen Prozessoren zurückzuführen sein könnte.

Als praktische, industrielle Anwendung granularer Untersuchungen wurde das Partikelsieben in Kapitel 4 betrachtet, welches eine essentielle Technologie zur Partikeltrennung in vielen industriellen Bereichen ist. Dieses Kapitel stellt ein numerisches Modell zur Untersuchung des Siebprozesses unter Verwendung der Diskrete Element Method vor, bei dem die Bewegung der einzelnen Partikel getrennt simuliert werden. Dynamische Parameter wie Partikelpositionen, Geschwindigkeiten und Orientierungen werden in jedem Simulationszeitschritt verfolgt. Das betrachtete Problem liegt in der Trennung runder Partikel verschiedener Größen mit Hilfe eines taumelnden vertikalen Zylinders, in den das Siebmaterial stetig zugeführt wird. Dieser Zylinder kann als glatter oder abgesetzter, mehrstufiger vertikaler Behälter beschaffen sein und wird als großes Reservoir für das Siebmaterial betrachtet. Die kleineren Partikel fallen normalerweise durch die Sieböffnungen, wohingegen die größeren Partikel zurückprallen und an konzentrischen Auslassöffnungen an der Zylinderwand austreten. Während des Siebvorgangs treten sowohl Partikel-Partikel-Wechselwirkungen als auch Partikel-Wand-Wechselwirkungen auf. Hierbei wird ein Feder-Dämpfer Penalty-Ansatz zur Berechnung der normalen Wechselwirkungskräfte und der Reibkräfte verwendet.

Als Folge der Kollisionen dissipiert kinetische Energie. Die Partikelverteilung, der Durchsatz der getrennten Partikel und die Effizienz des Trennvorgangs werden untersucht. Für bestimmte Geometriekonstellationen und Kontaktparameter wird der Partikelfluss, die Siebgüte und die Maschineneffizienz aufgezeichnet. Es werden Siebvorgänge in glatten

und gestuften Taumelzylindern untersucht. Sowohl für kontinuierliches Sieben als auch für das Sieben fester Chargengrößen ergibt sich eine ausgeprägte Abhängigkeit des Siebvorgangs von der Rotationsgeschwindigkeit der Trommel. Die Rotationsgeschwindigkeit sollte innerhalb bestimmter Grenzen festgelegt werden, um die Anzahl sortierter Partikel zu maximieren und den Durchsatz der verschiedenen Siebstufen zu optimieren. Zu hohe oder zu niedrige Geschwindigkeit führt zu einem schlechten Siebergebnis.

Darüberhinaus haben die Zuführrate, der Neigungswinkel und die Exzentrizität einen großen Einfluß auf die Effizienz der Maschine. Kleine Winkel zwischen 0.5° und 1° und Exzentrizitäten zwischen 25 und 50 mm sind zu empfehlen. Die Siebrauheit hat ebenfalls einen Einfluss auf die Anzahl der Partikel, die in der Siebtrommel verbleiben oder diese verlassen. Als optimaler Wert wird ein relativ geringer Reibwert vorgeschlagen. Desweiteren haben die Schwingungen der Trommel einen signifikanten Einfluss auf den Siebprozess. Schwingende Bewegungen der Siebtrommel führen zu besseren Ergebnissen als rotationsfreie Bewegungen oder Bewegungen mit kontinuierlicher Rotation. Bei gleichen Systemparametern liefert chargenweises Sieben bessere Ergebnisse als Sieben mit kontinuierlicher Zufuhr.

Eine Verbesserung der Genauigkeit der Simulation erfordert eine noch realistischere Implementierung der Kontaktkräfte und der zugeordneten Kontaktparameter des granularen Systems. Physikalische Kontakte innerhalb der Taumelsiebmaschine erfordern detailiertere Untersuchungen. Diese Parameter können aus speziellen Experimenten gewonnen werden. Zum besseren Verständnis des Siebevorgangs und im speziellen des Materialtransports zwischen den verschiedenen Siebstufen sind experimentelle Untersuchungen von Nöten.

Die Arbeit endet in Kapitel 5 mit einer Zusammenfassung der vorgestellten Betrachtungen und einem kurzen Überblick über zukünftige Erweiterungen.

Chapter 1

Introduction

1.1 Molecular dynamics, granular matter and parallel programming

The last decades have witnessed an enormous development in the research of granular media and particle simulation [91]. However, granular materials constitute the subject of vast literature, their research, beside possessing a long history, is currently active both in physics and engineering communities. Granular media studies are considered as being of great interest and are required in many engineering processes in different fields of industry. To improve the performance of such processes, a good understanding of the behavior of particle motion along with the increasing of the computers power are important and will contribute in the burgeoning of many different industrial applications of the granular technology.

The term *Molecular Dynamics* (MD) refers usually to computational techniques which use classical mechanics to analyze the structure and dynamics of molecular systems including polymers and macromolecules but are also applied to particulate materials. Contact phenomena are among the most interesting problems in molecular dynamics studies. Contact usually involves friction, which may only be neglected for simplicity in the case where frictional forces are small compared to the normal ones.

The complexity of a contact problem is due to at least three aspects. The *first* aspect is the nonlinear boundary condition at the contact region caused by the impenetrability constraint. The parameters of the contact region which include the sliding state, the frictional stress distribution, the shape and the size of the region, etc. are unknown before the analysis. The *second* aspect is the description of friction phenomena, which usually has no simple solution. The *third* aspect is the material and geometric nonlinearity. It is expensive and difficult to solve such a problem with geometric and physical nonlinearities.

Molecular dynamics simulation is a commonly used tool in physics and material science for modeling solids and liquids at the atomic level. In granular media often each particle is treated as a rigid spherical body and Newton's equations of motion are integrated to track the motion of each particle. The solution is obtained numerically, since the classical many-body problem is intractable, which means that the trajectories of the particles are obtained as a sequence of instantaneous positions and velocities at discrete intervals in time, i.e. *timesteps*. Interactions between grains are calculated with a contact algorithm that forbids interpenetration but allows separation, sliding and rolling with friction.

The mechanically correct description and simulation of contacts between many bodies is a very computation-time intensive topic. While by simple methods, a relatively low number of particles can be already computed with sufficient accuracy and acceptable computation times, there remain difficult and interesting problems as soon as elastic deformations, complicated particle geometries or a huge number of particles have to be considered. Tremendous improvements in computer power and computational methodology have accelerated the pace towards simulation of larger and larger systems, so that now simulations of millions of particles are possible. Such advances also enable researchers to obtain more information from their simulations.

Different parallelization techniques are developed in this field to raise up the efficiency of the simulations. Some of them are simple, e.g. the replicated data methods, and can be used to carry out molecular dynamics effectively, without the need for major changes from the approach used in serial codes. Others like spatial and domain decomposition methods are proposed as a path toward reducing inter-processor communication costs further to produce truly scalable simulation algorithms. A successful load balancing, i.e. each processor must have a roughly equal share of the work, and minimizing the ratio of communication cost to computational cost are two pointers to successful parallelization. In molecular simulation, both rise with the size of the system. However, computational costs, which usually depend on the number of pair interactions in the system, tend to rise quicker than communication costs as the size of the system increases. Consequently, most parallel algorithms are reasonably efficient for huge system of particles [118].

Solids can usually be processed in granular form. If measured by tons, it is the second-most-manipulated material by man behind water, especially coal and ordinary construction materials [21], therefore granular matter has been of central technological importance for a long time. Applications are enormously diverse: glass production, medicine and pharmaceuticals, maintaining railway ballasts, particle-classifying operations and many others. Nowadays, screening and segregation, in particular, are considered as technologically important demands for developing industrial operations which handle particulate materials. Since small reductions in energy consumption, increases in outputs and speeding up production represent substantial financial benefits for such processes, numerical simulation of these phenomena plays a significant role in this optimization procedure.

1.2 Literature survey and recent research

1.2.1 Classical molecular and contact dynamics

Contact phenomena are among the most important problems in engineering. The related studies already began hundreds of years ago. Investigating and analyzing this phenomenon can be enrolled under one of the three categories: *analytical*, *iterative* and *mathematical programming* methods. The *analytical* method might be referred back to 1881 by Hertz [44], who was a leading scientist in this area. He assumed that contacting bodies could be regarded as elastic half-spaces with small deformations. This method can be used in order to obtain a closed-form solution of the problem. In the *iterative* method, the fundamental principle is to take the contact boundary conditions into the finite element equations and then start looping until the contact conditions are satisfied. The third method is the *mathematical programming* in which the contact problem can be treated as the minimization of the systematic potential energy or other physical variables under some contact constraints.

Several methods of molecular dynamics were introduced by [43] to simulate large amounts of particles. Some of these methods are based on the exploitation of parallelization and metacomputing. Other approaches are more stochastic, e.g. the *Direct Simulation Monte Carlo* (DSMC), which simplify the calculation of collisions, positions and collision times. In [43] the performance of the various techniques was also compared. Some benchmarks were shown depending on the size of the system, the density of particles and the number of the processors.

The calculation and administration of the motion and the contacts of systems that are comprising many colliding bodies of round shape was investigated in [73]. Both, two dimensional and three dimensional cases are analyzed. Special attention is paid to the comparison of the efficiency of the employed algorithms with respect to calculation time. In order to model the behavior of many particles very efficiently, various methods from molecular dynamics are used. To reduce the high calculation time that is usually spent on collision detection, sophisticated sorting algorithms for the neighborhood search are required. This holds especially for large systems with many repeatedly colliding particles. The three methods, i.e. the *Verlet-Neighbor List* (VL), the *Linked Cell* (LC), and the *Linked Linear List* (LLL), are discussed and compared.

Different particle shapes are investigated by many researchers. The work in [84] proposed a new model for the description of complex granular particles and their interaction in molecular simulations of granular material in two dimensions. The grains are composed of triangles and are allowed to be convex or concave. Another new, computationally efficient model for the discrete element simulation of a certain class of non-round particles was described in [82]. The boundaries of the particles in this model are constructed

from the circular segments of different radii in such a way that connections between these segments are continuous. As such, the model does not permit the simulation of arbitrarily shaped particles, but it does allow a wide enough variety of shapes to assess the effects of non-round shapes in an efficient manner. A direct test of the models performance demonstrates that the model is much more efficient than other models for non-round particles currently available and is less than two times slower than models for the same number of round particles.

The *Discrete Element Method* (DEM) was used frequently for simulating different models in granular media for frictional, non-frictional and adhesive cases, see e.g. [72, 95]. This method is also used in [59] to examine the evolution of dense granular materials during deformation. The approach includes both normal and tangential forces at the contacts, since rotations of the particles are important due to friction. Several examples are presented, involving the formation of shear bands and the propagation of sound waves in non-cohesive granular packings. The ultimate goal is to obtain from such *microscopic* simulations the *macroscopic* constitutive laws that describe the material behavior in the framework of a continuum theory.

Other researches concentrates on studying the damping of particles during collision. The work e.g. in [63], concentrates on the investigation of damping phenomena during the arising of the particle-to-wall and particle-to-particle collisions under the vibrating motion of the structure. As a result of these collisions, the structure and the particles will exchange momentum and thus dissipate kinetic energy due to frictional and inelastic losses. Since the particle damping technology has been used successfully in many fields for vibration reduction, a discrete element method is used as a computational technique for particle damping modeling. In this study hundreds or thousands of particles as Hertz balls were considered and a discrete element model could describe the motions of these bodies and determine the energy dissipation.

1.2.2 Parallel computation

It is particularly suprising to know that the basics idea of *parallel computers* is not new. As early as 1842 Luigi Menabrea published an important article about the *'Analytical Engine'* which had been developed by the mathematician Charles Babbage, see [66]. He proposed computing numerical tables by means of parallel working mechanical calculators which have earned him a top spot in the history of early computing. As early as 1822 he speculated that a machine could be used to compute complex mathematical problems and calculate and correct errors in logarithm tables and astronomical charts.

In the 1980s, computer researchers believed computer performance was best improved by creating faster, more efficient processors. But this idea was challenged by the idea of clustering and parallel processing which, in essence, means linking together two or more

computers to work together on performing functions. The goal is to develop infrastructure so that end users do not need to know they are actually working on a cluster. Since the early 1990s there has been an increasing trend to move away from expensive and specialized propriety parallel supercomputers towards networks of workstations.

Basically, there are two main approaches for parallel programming. The first one is based on *implicit parallelism*. This approach has been followed by parallel languages and parallelizing compilers. The user does not specify, and thus cannot control, the scheduling of calculations and/or the placement of data. The second one relies on *explicit parallelism*. In this approach, the programmer is responsible for most of the parallelization effort such as task decomposition, mapping tasks to processors, and the communication structure. This approach is based on the assumption that the user is often the best judge of how parallelism can be exploited for a particular application. The use of explicit parallelism will obtain a better efficiency than parallel languages or compilers that use implicit parallelism, see e.g. [16].

Three different algorithms were discussed in [98] to speed up discrete-element simulations for granular matter. The *first* algorithm allows to determine neighborhood relations in polydisperse mixtures of particles of arbitrary shapes, either discs, ellipses, or polygons. The *second* algorithm allows to calculate the distance of two polygons in constant time, independent of the complexity of the shape of the polygons. This makes fast simulations of polygonal assemblies possible. The *third* method is a special type of parallelization technique, which is optimized for workstations with shared memory.

Two parallelization techniques were also discussed in [81] to implement the *Embedded Atom Method* (EAM) formalism for molecular dynamics on multiple-instruction/multiple-data (MIMD) parallel computers. The first method is the *atom-decomposition*, which is simple and suitable for small numbers of atoms. The second method is the *force-decomposition* which is considered as new and particularly appropriate for the EAM because all the computations are between pairs of atoms. Both methods have the advantage of not requiring any geometric information about the physical domain being simulated. They also presented timing results for the two parallel methods on a benchmark EAM problem and briefly indicated how the methods can be used in other kinds of materials in the MD simulations.

Other resarchers, see e.g. [95], found that the numerical simulation of granular flows, like many other particle-based methods, is computationally intensive for large-scale problems of industrial interest. Parallel computation has the potential to alleviate current computer-based limitations, allowing much larger granular systems with greater physical reality to be analyzed. A study of the implementation of a parallel algorithm is presented, together with performance measurements on a commodity cluster computer system. The results obtained validate not only the parallel algorithm, but also the potential role of such computer systems in industrial granular flow simulations.

Due to different factors related to the software and the hardware of the parallel environment, different *speedups* can be obtained for the parallel program. There is a hierarchy of terms of different speedups adapted from [69], e.g. the superlinear, linear superunitary, unitary, linear subunitary and sublinear speedups. These classifications depends on how much the speedup is scalable with the number of processors in use. The initial counter-reaction to the notion of superlinear speedup goes something like this: 'For a P-processor algorithm, simply execute the work of each processor on a single processor, and the time will obviously be no worse than P times greater. It will usually be less, because sources of parallel inefficiency are eliminated'. Faber et al. [27] have used this argument as a *proof* of the impossibility of superlinear speedup. The proof assumes fixed problem size, and that a single processor has all hardware necessary to duplicate the needs of the parallel algorithm.

A counter to this has been supplied in [77]. The idea here is that the serial processor has *loop overhead* in executing something k times, whereas the k-processor computer does not, permitting it to be more than k times faster. The idea assumes a loop is necessary to do something k times on a serial computer, clearly not true if program memory can store the straight-line code. Besides the latter effect, resarchers in [42] mention other apparent sources of superlinear speedup: *hidden memory latency*, *subdivision of system overhead*, and *randomized algorithms*. In the last case, independent processors traverse a solution space with better luck or less context switching cost than a single processor. Generally, historic explanations of superlinear speedup have turned out to be inefficiencies in the serial version caused by a sub-optimal program or by insufficient processor hardware.

Superlinear speedup that results from inefficiency in the serial algorithm is ephemeral and not particularly interesting. The superlinear speedup behavior could be due to the different speeds of memory inherent in distributed memory ensembles, and/or to the shift in time fraction spent on different-speed tasks, see [40].

1.2.3 Segregation and particle separation

Particle segregation is a very broad and complex phenomenon that can occur in various areas in industries which handle particulate materials. Although this very ancient technique may be dated back to thousands of years ago, a computational understanding of this has not yet been realized, due to the complicated size distribution and composition of industrial particulate solids, and comprehensive effect of particle motion under various operational parameters and screen configurations. The lack of advanced analytical and experimental techniques for the study of particulate systems has also hindered the progress in this subject area. As a result, most published information on sieve and screen performance has been empirical in nature, see [45, 56].

Particle separation phenomena are of great importance in granular media studies. Screen-

ers, classifiers, shakers and separators are used in a large number of industrial applications requiring separation and classification of powders or other bulk materials by particle size as well as separation of particles by density, magnetic properties or electrical characteristics. These machines are divided into different categories such as round and rectangular screeners, magnetic separators, electrostatic separators, rotary sifters, wet or concentrating tables, rake classifiers, classifying hydrocyclones, floatation systems and trammels.

The research work of Jansen and Glastonbury [45] in 1967/1968 in studying particle screening phenomena is among the earliest works in this field. They have tried to analyze the dynamics of screening processes and to study the factors that affect the screening performance. They have built their results on probability theories to understand the kinetics of sieving. The effect of the non-ideal aperture distribution of sieving on the sieve residue is studied in [46], where an algorithm for deducing an effective sieve residue from the rate of powder passage through a sieve is described. A detailed study of the sieving kinetics using batch sieving is described in [104] showing that the rate of sieving and its efficiency are determined by the numerical values of the sieving rate constants of each of the particle sizes in the feed mixture. Furthermore, they found that the near-mesh particles play a major role in the overall kinetic process and the presence of the oversized particles in the feed enhances the sieving rate dramatically. Continuous screening phenomena and comparisons with batch sieving are discussed in [4, 106]. The results showed that the two operations are comparable and the effect of the oversized particles is beneficial in speeding up the screening of near-mesh material.

The influence of some operating variables of separating sifters has been studied in [105]. Over a certain range of operating variables, the screening efficiency of two types of particles over a vibrating screen has been observed. The variables include the flow rate, deck angle, angular velocity and mesh size. The results show that the separation process is sensitive to the operating variables. Many other researchers have studied the radial and axial segregation process of granular mixtures in rotating cylinders, see e.g. [1, 54, 74, 119]. In [48] a constitutive model for the radial segregation flux in cascading layers of rotating cylinders is proposed. There, particle dynamics and Monte Carlo simulations for steady flows down an inclined plane are used for studying the density segregation.

Three decades ago, the discrete element method (DEM) that describes the motion of particles and models the behavior of dense solid assemblies in soil mechanics was proposed [20]. The DEM was adapted in [67] to be used for the analysis of the internal dynamics of tumbling mills. An elastic-perfectly plastic contact model was used, see [68]. Using this model requires only material parameters that may be obtained from standard tests. Some other researches studied volume filling fractions of the particles charged in the rotating drum of the separating machine, see [23]. Numerically the dynamics of the size segregation process of binary particle mixtures in rotating drums is studied. A direct dependency between the different particle size ratios and the final amount of the radial

segregation flow was marked in their study.

Analytical works, e.g. [73, 71, 111], consider particles with or without cohesive, adhesive, frictional forces between them during contact. The spring-dashpot model or penalty method uses different contact parameters which essentially embody contact properties, see e.g. [68]. None of these parameters are typical material properties and hence they are difficult to determine experimentally. In [55] a numerical model for studying a screening separation of granular mixture comprising two different sizes of particles on an inclined surface is presented. A two-dimensional transient model has been developed to calculate the particle motion on and through the screen using the DEM. They also discussed the influence of the feeding rate, the depth of the particle bed and the screen inclination on the screening efficiency, see [56]. Their analytical results were compared with some experimental studies.

1.3 Layout of the thesis

Due to their wide usage in industrial and technological processes, granular materials have captured great interest in this research. The aim of this study is to give a general view of some of the already existing methods of simulating granular media. Reducing the simulation time of large granular systems by introducing parallel computation strategies is another goal of this work. Further, a practical application of particle segregation technology, which is represented in modelling a real tumbling screening machine, is analyzed and investigated.

In Chapter 1, a brief literature survey of granular media studies and some of their investigation methods are presented. A quick overview of the some researches in the field of parallel programing and sorting technology is also put in hand.

In Chapter 2, a general description of the molecular dynamic problems is presented. The contact time calculation along with different Verlet integrators are introduced in this chapter. A detailed formulation of the *Discrete Element Method* (DEM) and penalty method of the spring-dashpot model are involved. Different strategies for a concurrent and distributed collision detection and contact force computation are developed, implemented and investigated. In this chapter, the effect of normal and tangential collision along with the influence of damping and adhesion of the colliding round particles are also explained. Some basic techniques for speeding up simulations of particulate systems by using some proper sorting algorithms and neighbor list computations, e.g. the *Verlet* approach and the *linked linear list* method, are used and compared. Finally, the chapter ends with a brief description of the computational structure of the used serial or sequential molecular dynamics code, i.e. *MOLDYN* and the necessary input files required for the code initialization.

Since the related studies are often based on numerical simulations, it becomes quite challenging to investigate computational phenomena of large granular systems using personal computers with single processor. In Chapter 3, the spatial decomposition method is basically used in building the parallel programing codes. Needless to say that the important factor which affects the success of the numerical procedure is how much one has access to a computer system which is powerful enough to handle the problem of interest. In this chapter, existing sequential algorithms [71] are extended and modified in such a way that modern high performance computers can be utilized for their parallel evaluation. The library functions of the *Parallel Virtual Machine* (PVM) are used to handle communication between processors in a distributed memory environment. For different system sizes and a different number of computing machines, the computational time and the *speedups*, which is the usually used measure of the program scalability, are drawn and analyzed.

As a practical engineering application of granular studies, particle screening, which is considered as an essential technology of particle separation in many industrial fields, is selected to be investigated in Chapter 4. This chapter presents a numerical model for studying the particle screening process using the discrete element method that considers the motion of each particle individually. Dynamical quantities like particle positions, velocities and orientations are tracked at each time step of the simulation. The particular problem of interest is the separation of round shaped particles of different sizes using a rotating tumbling vertical cylinder while the particulate material is continuously fed into its interior. This rotating cylinder can be designed as a uniform or stepped multi level oblique vertical vessel and is considered as a big reservoir for the mixture of particulate material. The finer particles usually fall through the sieve openings while the oversized particles are rebounded and ejected through outlets located around the machine body. Particle-particle and particle-boundary collisions will appear under the tumbling motion of the rotating structure. Herein, the penalty method, which employs spring-damper models, is applied to calculate the normal and frictional forces.

In this chapter, the efficiency of the sorting process is determined, e.g., by counting the number or measuring the mass of both right-sorted and undesired-gangue particles of the whole process at the different levels of the machine. Different parameters affect the machine performance, e.g., the machine speed, the feeding rate of particles, the barrel inclination angles, the shaft eccentricity, the barrel oscillation, the mesh clearance and the roughness of the sieve. Different simulations are conducted to study the effect of these parameters. Continuous feeding processes with different flow rates as well as batch sieving with a limited number of particles are analyzed and compared.

Finally, the thesis ends in Chapter 5 with a general summary of the presented work and a short overview of the proposed work in the future.

Chapter 2

Basics of Granular Matter and Molecular Dynamics

Molecular simulation and modeling methods are undergoing rapid development. They are increasingly important tools for fundamental and applied research in academia and industry in such diverse fields as design and material science. The Discrete Element Method (DEM) [5, 6, 88] describing a classical particle molecular system as a function of time has been used for several decades and successfully applied to understand and explain macro phenomena from micro structures.

The classical N-body problem lacks a general analytical solution, thus numerical solutions are needed. Solving the dynamics numerically and evaluating the interactions tends to be computationally expensive already for a few thousand particles [64]. Especially, the interactions are generally the computationally dominant part. For large scale Molecular Dynamics (MD) simulations, we do therefore not only require a powerful machine, but also new algorithmic techniques and parallelization schemes to solve the problem in reasonable time. With the introduction of novel computational algorithms (e.g., multiple time stepping integration schemes, fast electrostatic force algorithms, etc.) and large scale parallel computers, it became possible to study larger systems beyond billions of particles [11, 92]. Thus, with such problem sizes certain macroscopic properties of matter can be studied.

There is a proliferation of programs for MD and DEM [15, 22, 26, 57, 62, 75, 79, 90, 103, 112, 114], several of these are robust production codes; some with scalable parallel implementations. They cover common particulate and molecular dynamics problems and are excellent tools to perform simulations. However, many of the codes are legacy programs that are either poorly organized or extremely complex. One important factor is usually the large number of people that contributed to the writing of the codes and the lack of a strong coordination to enforce design, code organization, and programming guide-lines. Most MD applications also suffer from missing documentation that is needed

to understand both the design and implementation. Furthermore, some codes have a long history and were modified multiple times to solve different types of problems at different points in time [64].

2.1 Description of the contact problem

2.1.1 Verlet algorithm

One of the important parts of a particle dynamics programs is the time integration algorithm which is necessary to integrate the equations of motion of the interacting particles and find their trajectories, new positions and orientations. The integrator should generally be accurate, stable and easy to be implemented and coded. Time integration algorithms are based on finite difference methods, where time is discretized on a finite grid and the time step Δt being the distance between consecutive points on the grid. Knowing the positions and accelerations at a specified time and the positions from previous times, the integration scheme can give the new positions and some of their time derivatives at some later times of simulation.

Of course, these schemes are not exact and there is a kind of approximation where different types of errors are unavoidable but can be minimized. In particular, one can distinguish between truncation and round-off errors. Truncation errors are related to the accuracy of the finite difference method with respect to the true solution. Finite difference methods are usually based on a Taylor expansion truncated at some terms. These errors do not depend on the implementation, they are intrinsic to the algorithm. Round-off errors are related to errors associated to a particular implementation of the algorithm, for instance, to the finite number of digits used in computer arithmetics.

Both errors can be reduced by decreasing Δt which controls the fineness of the integration. Larger time step decreases the computation time. But too large time step will lead to instability and inaccuracy in the numerical integration [94, 116]. Verlet algorithms [113] assume that velocities and accelerations are constant over a given time step which is often set to $\Delta t = 10^{-5}$s in this study. A simple DEM algorithm will have to adopt a time step short enough to handle the fastest variables. For large Δt, truncation errors dominate, but they decrease quickly as Δt is decreased. For instance, the Verlet algorithm has a truncation error proportional to Δt^4 for each integration time step. Round-off errors decrease more slowly with decreasing Δt, and dominate in the small Δt limit. Using double precision helps to keep round-off errors at a minimum.

Two popular integration methods for molecular dynamics calculations are the Verlet algorithm and predictor-corrector algorithms. The Verlet algorithm is one of the most commonly used time integration algorithms in this field and it is used as the integrator

of the equations of motion in this study. The main idea in formulating this algorithm is to write two forward and backward expressions for the third-order Taylor expansions of the positions $\mathbf{r}(t)$. The forward expression of the Tayler series at $t + \Delta t$ is

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \dot{\mathbf{r}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{r}}(t)\Delta t^2 + \frac{1}{6}\dddot{\mathbf{r}}(t)\Delta t^3 + O(\Delta t^4) \ , \tag{2.1}$$

and the backward one at $t - \Delta t$ can be written as

$$\mathbf{r}(t - \Delta t) = \mathbf{r}(t) - \dot{\mathbf{r}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{r}}(t)\Delta t^2 - \frac{1}{6}\dddot{\mathbf{r}}(t)\Delta t^3 + O(\Delta t^4) \ , \tag{2.2}$$

where $\dot{\mathbf{r}}(t)$ and $\ddot{\mathbf{r}}(t)$ are the velocity and acceleration of the moving particle at time t which can be rewritten as $\mathbf{v}(t)$ and $\mathbf{a}(t)$, respectively. Summing and neglecting the higher order terms, the two Equations (2.1) and (2.2) together give

$$\mathbf{r}(t + \Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t - \Delta t) + \mathbf{a}(t)\Delta t^2 + O(\Delta t^4) \ . \tag{2.3}$$

As it appears from Equation (2.3) the Verlet algorithm uses positions and accelerations at time t and the old positions from time $t - \Delta t$ to calculate new positions at time $t + \Delta t$ without need to use velocities. The large popularity of this algorithm among molecular dynamics simulators is due to its simplicity in implementation, accuracy and stability.

This version of the Verlet algorithm does not require the velocities directly. Even though they are not needed for computing the trajectories, their knowledge is sometimes necessary to compute the kinetic energy, whose evaluation is necessary to test the conservation of the total energy of the granular system. These tests are very important to verify that a MD simulation is proceeding correctly. The velocities $\mathbf{v}(t)$ can be calculated from the old and new positions by using

$$\mathbf{v}(t) = \frac{\mathbf{r}(t + \Delta t) - \mathbf{r}(t - \Delta t)}{2\Delta t} \ . \tag{2.4}$$

Therefore, modifications to the original Verlet scheme have been proposed to overcome these difficulties and the velocities are handled somehow better. These schemes differ in what variables are stored in memory and at what times. One of these is the *leap-frog scheme* [83]. This algorithm defines velocities that are half a time step behind, or in front of, the current time step Δt. In this algorithm, the velocities are first calculated at time $t + \frac{1}{2}\Delta t$ as

$$\mathbf{v}(t + \frac{1}{2}\Delta t) = \mathbf{v}(t - \frac{1}{2}\Delta t) + \mathbf{a}(t)\Delta t \ . \tag{2.5}$$

The velocity in Equation (2.5) is used to calculate the positions $\mathbf{r}$, at the next time step $t + \Delta t$ as

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t + \frac{1}{2}\Delta t)\Delta t \ . \tag{2.6}$$

In this way, the velocities *leap* over the positions, then the positions *leap* over the velocities. The advantage of this algorithm is that the velocities are explicitly calculated, however, the disadvantage is that they are not calculated at the same time as the positions. The current velocities at time t can be approximated by

$$\mathbf{v}(t) = \frac{1}{2}\left(\mathbf{v}(t+\frac{1}{2}\Delta t) + \mathbf{v}(t-\frac{1}{2}\Delta t)\right) \ . \tag{2.7}$$

An even better implementation of the same basic algorithm is the so-called *velocity Verlet scheme*. This algorithm stores positions, velocities and accelerations at the same time t and calculates them again at $t+\Delta t$, see e.g. [108],

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 \ , \tag{2.8}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \frac{1}{2}\mathbf{a}(t+\Delta t)\Delta t \ , \tag{2.9}$$

and the velocity at $\mathbf{v}(t+\frac{1}{2}\Delta t)$ can be directly computed from

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t) + \frac{1}{2}\mathbf{a}(t)\Delta t \ . \tag{2.10}$$

The implementation of the velocity Verlet scheme is shown graphically in Fig. 2.1.

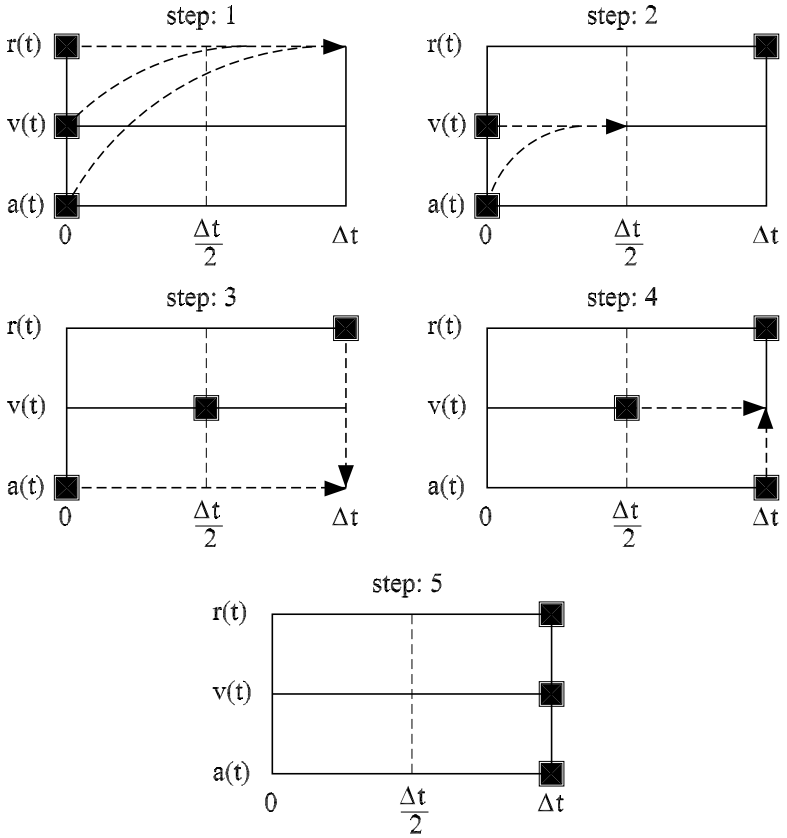


Figure 2.1: A graphical representation of the velocity Verlet algorithm

As an important extension of the MD method, other advanced versions of the Verlet algorithm have been generated, i.e. a time-reversible Verlet multiple-timestep algorithm. This scheme has the ability to tackle systems with multiple time scales: for example, particulate materials consisting of both heavy and light particles. This time reversibility $\mathbf{r}(t + \Delta t) \leftrightarrow \mathbf{r}(t - \Delta t)$ is considered as an advantage of Verlet algorithm.

2.1.2 Contact time calculation

The duration of contact between the colliding particles is now discussed. Since the size of time step Δt employed in the integration of the equations of motion should be carefully chosen according to the frequency of the oscillation, or rate of damping of a typical second-order differential equation of the simple damped harmonic oscillator of the form, see e.g. [58]

$$\ddot{\delta}_{ij} + 2\eta\dot{\delta}_{ij} + {\omega_0}^2\delta_{ij} = 0 \ , \tag{2.11}$$

where $\eta = \eta_0\omega_0$, δ_{ij} is the spring elongation which will represent later the overlap between the two particles i and j, 2η, η_0 and ω_0 are the damping parameter, the damping ratio and the undamped frequency of the system, respectively. Assuming the two rigid particles of masses m_i and m_j, see Fig. 2.2, are connected by spring-dashpot system of elastic spring constant k_p and viscous damping coefficient c_p, the equivalent mass of the two particles can be expressed as

$$\frac{1}{m_{ij}} = \frac{1}{m_i} + \frac{1}{m_j} \ , \tag{2.12}$$

from which we can write

$$m_{ij} = \frac{m_i\, m_j}{m_i + m_j} \ . \tag{2.13}$$

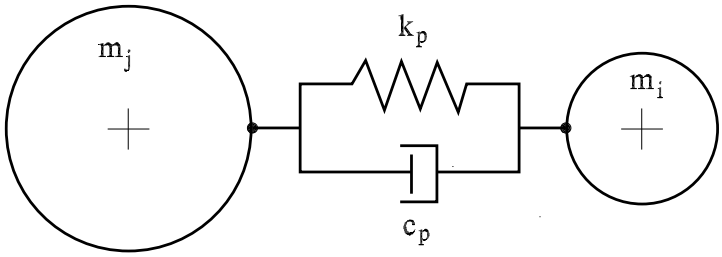


Figure 2.2: Spring-dashpot-mass-oscillator with two-end particles

The damping parameter and the undamped frequency can be written in terms of k_p, c_p and m_{ij} as

$$2\eta = \frac{c_p}{m_{ij}} \ , \qquad \omega_0 = \sqrt{\frac{k_p}{m_{ij}}} \ . \tag{2.14}$$

Using the collision theory of Hertz, see e.g. [99], the material constants k_p and c_p can be calculated. In order that the integrator is able to follow the system oscillation, the time

step size should be smaller than the reciprocal of the natural frequency Ω which can be given as

$$\Omega = \sqrt{{\omega_0}^2 - \eta^2} = \sqrt{\frac{k_p}{m_{ij}} - \frac{{c_p}^2}{4{m_{ij}}^2}} \ . \tag{2.15}$$

Assuming that the spring is initially unstretched $\delta_{ij}(0) = 0$ with initial relative velocity $\dot{\delta}_{ij}(0) = v_0$, the solution of Equation (2.11) is

$$\delta_{ij}(t) = \frac{v_0}{\Omega} \sin(\Omega t) e^{-\eta t} \ , \tag{2.16}$$

and the relative velocity is

$$\dot{\delta}_{ij}(t) = \frac{v_0}{\Omega} e^{-\eta t} \left(\Omega \cos(\Omega t) - \eta \sin(\Omega t)\right) \ . \tag{2.17}$$

The maximum overlap δ_{max} of the two colliding particles i and j will be reached at the maximum time when there is no further movement of them toward each other, i.e. $\dot{\delta}_{ij}(t_{max}) = 0$. Thus we can obtain

$$\delta_{max} = \frac{v_0}{\omega_0} e^{(-\eta/\Omega)\, \arcsin(\Omega/\omega_0)} \ . \tag{2.18}$$

From the natural frequency in Equation (2.15), the duration of contact between the two particles can be calculated as

$$\Omega = 2\pi f = \frac{2\pi}{T} \tag{2.19}$$

which gives

$$T = \frac{2\pi}{\Omega} \ . \tag{2.20}$$

As long as $\eta < \omega_0$ the contact time is considered to be the half time period of oscillation and can be expressed as

$$t_c = \frac{T}{2} = \frac{\pi}{\Omega} \ . \tag{2.21}$$

The calculation of the contact time is performed for all particle-particle contacts which appears in the beginning of our simulations. The time step Δt of the integration has to be chosen clearly smaller than a typical natural oscillation of a contact. A ratio of $1 : 20$ proved to give satisfying results. This means that the computational time step Δt is carefully selected to be at least less than the contact time t_c divided by 20 to be sure that it is small enough to capture all dynamical quantities of the particles, i.e. positions and velocities during contact

$$\Delta t < \frac{t_c}{20} \ . \tag{2.22}$$

The normal dissipation in a collision is characterized by the *coefficient of normal restitution* e_n, which is considered as a measure of the elasticity of the collision. This coefficient is defined as the ratio between the normal component of the relative velocity after $v_n^{(aft)}$ and before $v_n^{(bef)}$ the collision,

$$e_n = \left|\frac{v_n^{(aft)}}{v_n^{(bef)}}\right| \ , \qquad 0 \leq e_n \leq 1 \ . \tag{2.23}$$

A perfectly elastic collision (with no normal dissipation) has a coefficient of restitution of 1 as in two diamonds bouncing off each other. A perfectly plastic, or completely inelastic, collision has $e_n = 0$ as two lumps of clay that stick together without bouncing. From Equation (2.23) the coefficient of restitution can be calculated in terms of η and Ω as

$$e_n = e^{-\eta t_c} = e^{(-\pi\eta/\Omega)} \ . \tag{2.24}$$

Note that for the *linear* spring-dashpot interactions, the contact time and the coefficient of restitution are velocity independent. While for *nonlinear* forces, i.e. using nonlinear Hertz models of interactions, t_c and e_n become velocity dependent, see e.g. [61].

The calculation of the contact time between two colliding particles is clarified here through this small example. Assume two different size spherical glass balls of masses $m_i = 0.0383$g and $m_j = 0.09$g and radii $r_i = 1.5$mm and $r_j = 2$mm. The two masses are connected together with an elastic spring of stiffness $k_p = 7.36$N/m and damping element with a coefficient of $c_p = 8.96 \times 10^{-4} \frac{N}{m/s}$. The equivalent mass of the oscillating system is

$$m_{ij} = \frac{0.0383g \ 0.09g}{0.0383g + 0.09g} = 0.0269g = 2.69 \times 10^{-5} kg \ . \tag{2.25}$$

Using Equation (2.14), one can find the damping parameter η

$$\eta = \frac{8.96 \times 10^{-4} \frac{N}{m/s}}{2 \ (2.69 \times 10^{-5} kg)} = 16.65 \frac{1}{s} \ , \tag{2.26}$$

and the frequency of the elastic oscillator ω_0 as

$$\omega_0 = \sqrt{\frac{7.36 \frac{N}{m}}{2.69 \times 10^{-5} kg}} = 523.07 \frac{1}{s} \ . \tag{2.27}$$

The natural frequency can be directly calculated from Equation (2.15)

$$\Omega = \sqrt{(523.07)^2 - (16.65)^2} \frac{1}{s} = 522.8 \frac{1}{s} \ . \tag{2.28}$$

The duration of contact between both particles is therefore

$$t_c = \frac{\pi}{\Omega} = \frac{\pi}{522.8} \approx 0.006s \ . \tag{2.29}$$

Using Equation (2.22), the recommendable computational time step in this case will be

$$\Delta t < \frac{t_c}{20} = \frac{0.006}{20} s = 0.0003s = 3 \times 10^{-4} s \ . \tag{2.30}$$

The coefficient of restitution can be calculated by Equation (2.24) as

$$e_n = e^{-(\pi)(16.65)/(522.8)} = e^{-0.1} = 0.905 \ , \tag{2.31}$$

which denotes a largely elastic collision between the glass balls.

2.2 Contact forces calculation

2.2.1 Discrete element method

There are two main approaches for modelling granular materials, the *continuous* and *discrete* modelling approaches. The continuum-based model works on the macroscopic level where the description of the behavior of the granular material depends on the constitutive equations, whose parameters are usually measured experimentally like stress, strain, and other physical quantities describing the state of the system, see e.g. [52]. In this kind of models the granular structure of the material is idealized with a continuum of *material points*. Interactions between grains are calculated with a contact algorithm that forbids interpenetration, but allows separation, and sliding and rolling with friction. A separate contact detection step is not required, and the numerical cost for the contact model scales linearly with the number of grains. The corresponding field equations can be derived from the properties of a representative elementary volume (REV) in the vicinity of the material point. Other methods like the finite-element-method (FEM) or the boundary-element-method (BEM) are used to investigate the deformable bodies with peculiar contact dynamics, see e.g. [24, 78].

An alternative straightforward approach to model granular material is the so called particle dynamics or discrete element method (DEM). DEM is a numerical technique pioneered by Cundall [18] for problems in rock mechanics where the continuity between the separate elements does not exist. This technique deals with the granular materials on the microscopic level where the material is assumed to be composed of distinct grains that interact with each other. Furthermore, it is capable of handling a wide range of material behavior, inter-body interaction force laws and arbitrary geometries.

The power of DEM in simulating the real processes and in identifying the mechanisms is evident. Thus it is a useful tool in understanding the physics of the processes and problems. The continuum-based models can simulate deformation mechanisms in jointed and particulate media. DEM clearly can do the same in a much simpler manner. Because DEM can monitor internal stresses and contact behavior unobtrusively, it is significant for use in understanding fundamental particulate material behavior. In addition, since external stresses and stress paths can be controlled precisely it can be used for developing and validating constitutive relationships of any particulate materials such as soil, rock, grain or ceramic powder by using appropriate particle properties, sizes and shapes.

DEM is one of the most frequently-used tools for explaining the experimentally observed facts from a more fundamental approach [101]. Using a micro-structural approach is very useful to find the macroscopic state variables in terms of micro-variables such as particle displacements, contact forces and local interactions. To do this, there are different averaging strategies in the literature to transfer from discontinuous models to a continuum

description, see e.g. [51].

DEM can be applied on two or three-dimensional dense systems of particles. Calculation of the interaction between the particles is a very time consuming part of the simulation. The computing power nowadays offers the possibility to consider even more realistic three-dimensional systems with spheres, ellipsoids and blocks to simulate particulate materials, which was prohibitively complex and expensive a few years ago. However, due to the explicit nature of the algorithm, it is necessary to use a very small time step of simulation to guarantee numerical stability and accuracy [101].

Many researchers are attempting to expose the advent of the effective use of the supercomputer and parallel computing by modifying the discrete element simulation codes. Although both have a significant importance, some consensus is emerging on the use of a parallel processing machine to simulate particulate materials using DEM rather than vector processing machines [19]. It may be expected that these studies will allow a major step forward to the more realistic simulations of particulate solids and a better understanding of the constitutive behavior of particulate materials.

There are many practical applications for the DEM where many useful results may be obtained by applying this numerical approach including: mixing and segregation in industrial blending systems, solving three-dimensional impact dynamics problems and vibrating feeders, soil-structure interaction problems, rock and ice mechanics, simulating gravity flow of bulk solid materials in mines, bulk material transportation, fracture mechanics and large deformation problems and many others.

DEM takes into consideration that every single particle is accounted for, and the contacts are handled then by solving Newton's equations of motion. This approach is applied in this study to analyze the short-range interaction molecular dynamics between neighboring particles inside the domain of the granular environment. The typical long-range forces which have an effect on the far-distance particles even without physical contact like electro-static forces, will not be considered in this study. If the material is dry and non-cohesive, the only interactions between grains are friction and repulsion [35], otherwise the attractive adhesion forces will appear. In this case, the normal contact forces modelled by the penalty method and the transverse tangential forces using Coulomb friction are implemented.

As a summary, Fig. 2.3 presents the DEM procedure. The starting conditions are the initial positions of particles and their initial velocities. Using DEM and soft particle contact, the contact forces and moments are calculated. Integrating the equations of motion will give the new velocities, positions and orientations in the next time step. This cycle has to be repeated for all time steps during simulation.

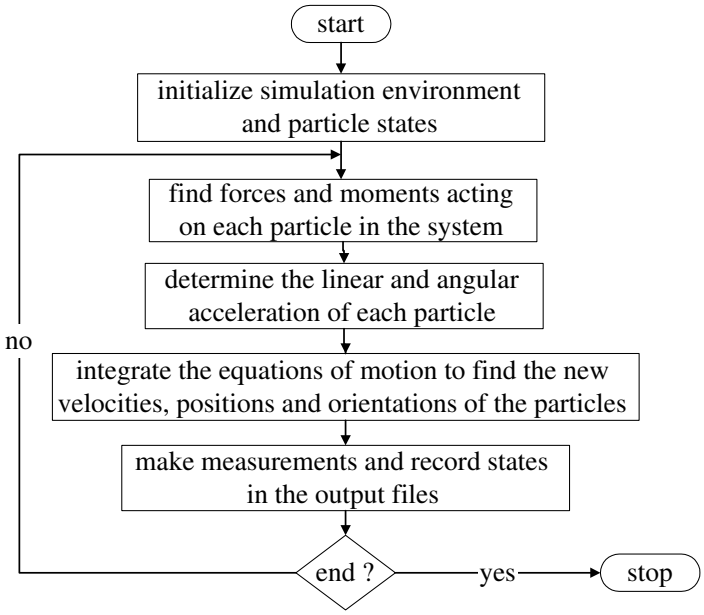


Figure 2.3: Flowchart of DEM

2.2.2 Penalty approach of spring-dashpot model

1. **Particle-to-particle contact**
   In a spatial multibody system, each free body has six degrees of freedom in translation and rotation, see [97] and [100]. In order to analyze the system, the discrete element method is used. In this method, the motion of each single particle is considered individually. Particle positions, velocities and orientations are computed at each time step of simulation. The equations of motion of the rigid body i is governed by six differential equations

$$m_i \mathbf{a}_i = \mathbf{F}_i \ , \qquad \mathbf{I}_i \boldsymbol{\alpha}_i = \mathbf{M}_i \ , \tag{2.32}$$

   where m_i and $\mathbf{I}_i$ are the mass and the inertia tensor of particle i, $\mathbf{F}_i$ and $\mathbf{M_i}$ are the force and torque vectors, $\mathbf{a}_i$ and $\boldsymbol{\alpha}_i$ are the linear and angular accelerations. Considering two bodies i and j in an N particle system, see Fig. 2.4a, the force $\mathbf{F}_i$ and torque $\mathbf{M}_i$ acting on particle i can be calculated as

$$\mathbf{F}_i = \sum_{j=1,\ j\neq i}^{N} \mathbf{F}_{ij} + m_i \mathbf{g} \ , \tag{2.33}$$

$$\mathbf{M}_i = \sum_{j=1,\ j\neq i}^{N} \mathbf{M}_{ij} = \sum_{j=1,\ j\neq i}^{N} \mathbf{r}_i \times \mathbf{F}_{ij} \ , \qquad i = 1, ..., N \ , \tag{2.34}$$

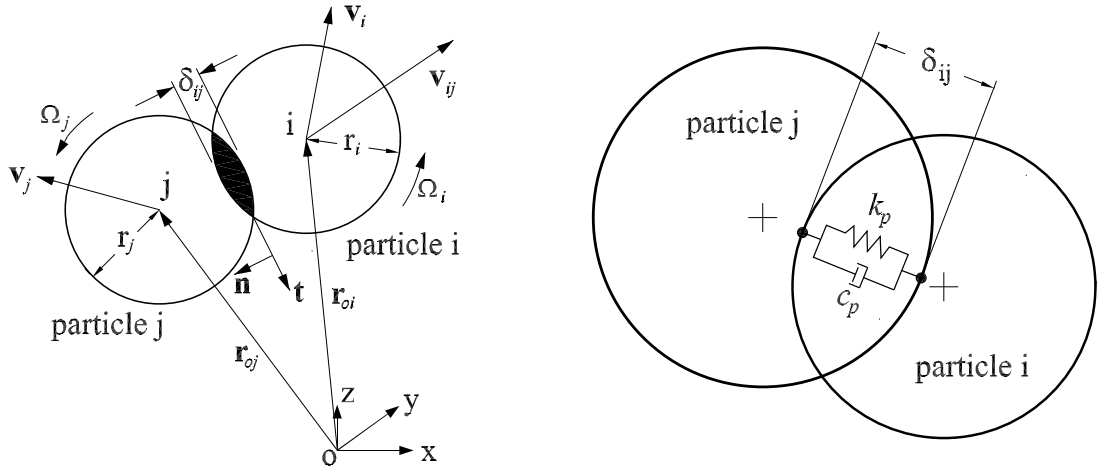
   where $\mathbf{r}_i$ is the vector from the center to a point on the surface of particle i and $\mathbf{g}$ is the gravity vector.

   The contact calculations are based on the soft-particle model which leads to a deterministic simulation where the state of each particle in the system and all particle

interactions are determined using physical laws. Applying the penalty method, we can determine the normal and frictional forces between the colliding particles, see Fig. 2.4b. This model assumes that the contact forces result from an unphysical overlap between the bodies in contact, see [2] and [25]. As a result of collisions, the particles will dissipate energy due to the normal and frictional contact losses. The total force between the two particle i and j is

$$\mathbf{F}_{ij} = \mathbf{F}_{ij}^n + \mathbf{F}_{ij}^t \ , \tag{2.35}$$

where $\mathbf{F}_{ij}^n$ and $\mathbf{F}_{ij}^t$ are the normal and tangential components of $\mathbf{F}_{ij}$.



(a) vector description of particle interaction (b) spring-damper model

Figure 2.4: Particles in overlap

• Normal contact forces

The contact between a pair of bodies is considered to be distinct single-point contact. The particles are considered to be rigid, however the contacts are deformable. The normal force $\mathbf{F}_{ij}^n$ between two colliding particles i and j is the summation of two parts, the elastic repulsive force of the spring and dissipative viscous force of the damping element. This force can be formulated as

$$\mathbf{F}_{ij}^n = (k_p \delta_{ij} + c_p \dot{\delta}_{ij})\mathbf{n} \ . \tag{2.36}$$

Defining the virtual overlap δ_{ij} of the particles and the normalized line of centers of the two particles $\mathbf{n}$ as follows

$$\delta_{ij} = (r_i + r_j) - (\mathbf{r}_{oi} - \mathbf{r}_{oj})^T \mathbf{n} \ , \qquad \mathbf{n} = \frac{\mathbf{r}_{oi} - \mathbf{r}_{oj}}{|\mathbf{r}_{oi} - \mathbf{r}_{oj}|} \ , \tag{2.37}$$

k_p, c_p are the spring stiffness and damping coefficient of the penalty approach, $\mathbf{n}$ is the normal unit vector between the centers of the two particles, δ_{ij} and $\dot{\delta}_{ij}$ are the

overlap and the relative velocity in the normal direction between the two colliding particles i and j.

With a little mathematical manipulation of dissipative energy expressions [49, 53], where the dissipated energy ΔT in the impact of two spheres may be expressed in terms of the coefficient of restitution and the relative approach velocity as

$$\Delta T = \frac{1}{2}\frac{m_i m_j}{m_i + m_j}\ (v_n^{(bef)})^2(1 - e_n^2)\ , \tag{2.38}$$

the damping coefficient c_p can be expressed in terms of the coefficient of normal restitution e_n as

$$c_p = \mu_h \delta_{ij}^{3/2}\ , \qquad \mu_h = \frac{3k_p(1 - e_n^2)}{4v_n^{(bef)}}\ , \tag{2.39}$$

where μ_h is the hysteresis damping factor. If the damping is too large, one can calculate the damping coefficient using the expression

$$c_p = 2\ \sqrt{\frac{m_i m_j}{m_i + m_j}\ k_p} = 2\ \sqrt{m_{ij} k_p}\ . \tag{2.40}$$

Instead of the pre-mentioned linear model of viscous dissipation, an alternative way to introduce dissipation to the system is to use different forces for loading and unloading [93]. This approach takes into consideration the permanent, plastic deformation during a typical contact. In this approach the *loading* spring constant k_l is usually smaller than the *unloading* spring constant k_{un}, i.e. the unloading path lies below the loading one $k_l < k_{un}$ and, thus, the energy lost is proportional to the amount of deformation of the particles. For this case the velocity-dependent damping $c_p\dot{\delta}_{ij}$ will be dropped from this model. Therefore, the contact force in the normal direction will have the form

$$F_{ij} = F_{ij}^n = \begin{cases} k_l \delta_{ij} & \text{for loading case ,} \\ k_{un}(\delta_{ij} - \delta_0) & \text{for un/reloading case ,} \end{cases} \tag{2.41}$$

where δ_0 is the finite penetration at which the contact force vanishes during unloading, see Fig. 2.5a. This amount of overlapping can be calculated by applying the force continuity at the maximum pentration as

$$k_l \delta_{max} = k_{un}(\delta_{max} - \delta_0)\ . \tag{2.42}$$

The dissipated energy may be identified as the surface area within the triangular region in Fig. 2.5a which will lead to a so-called momentum restitution coefficient ϵ_m that can be defined as

$$\epsilon_m = \sqrt{k_l / k_{un}}\ , \tag{2.43}$$

where $0 < \epsilon_m < 1$. Therefore, Equation (2.42) can be rearranged and written again as

$$\delta_0 = (1 - \epsilon_m^2)\delta_{max} \ . \tag{2.44}$$

During the initial loading the force increases linearly with slope k_l with respect to the penetration δ until the maximum overlap δ_{max} is reached. This overlap is the maximum penetration that could be reached at which the relative velocity between the two colliding particles is zero. After this point the two particles start to move apart from each other in the unloading process. Unloading follows down the second linear line of slope k_{un} until the force vanishes after a time t_c while crossing the x-axis at a finite penetration δ_0, see Equation (2.44). Since there is plastic deformation, the overlapping will not return back to zero as the force vanishes, but to δ_0.

The time t_c can be calculated referring to Equation (2.21) as the sum of the half contact duration of particles with either stiffness k_l and k_{un} and can be expressed as [58]

$$t_c = \frac{\pi}{2}\left(\sqrt{\frac{m_{ij}}{k_l}} + \sqrt{\frac{m_{ij}}{k_{un}}}\right) \ . \tag{2.45}$$

While the two particles are not completely separated ($\delta = \delta_0 \neq 0$), it could happen that they are opposed to an external collision from one or more of the surrounding particles. This collision will apply an additional external force to the original particles and let the bodies get closer again due to the *reloading* effect. Reloading for $0 < \delta_r < \delta_0$, where δ_r is the overlap after which the reloading starts, will then take place with gradient k_{un} along the linear line of the force $k_{un}(\delta - \delta_r)$ until the original loading curve is reached. Further reloading will follow up then the original loading path with a force $k_l\delta$ until the maximum deformation.

The missing need of including any arbitrary damping c_p and the direct analytical prediction of the parameters ϵ_m and t_c is considered as a great advantage of this model. Furthermore, this model in hand can be easily extended to include the *adhesion* effect during contact which occurs when the attractive forces are applied instead of repulsive ones, i.e. when

$$\mathbf{F}_i^T \mathbf{n}_{ij} < 0 \ , \tag{2.46}$$

where n_{ij} is the normal unit vector from the centers of particle i to particle j. The force law including adhesion can be reformulated to appear as

$$F_{ij} = F_{ij}^n = \begin{cases} k_l\delta_{ij} & \text{for loading case ,} \\ k_{un}(\delta_{ij} - \delta_0) & \text{for un/reloading case ,} \\ k_{ad}\delta_{ij} & \text{for adhesion (unloading) case ,} \end{cases} \tag{2.47}$$

where k_{ad} is the slope of the adhesive line which should be negative $k_{ad} < 0$ in case of adhesion, see Fig. 2.5b. However, if the achieved attractive force is not strong

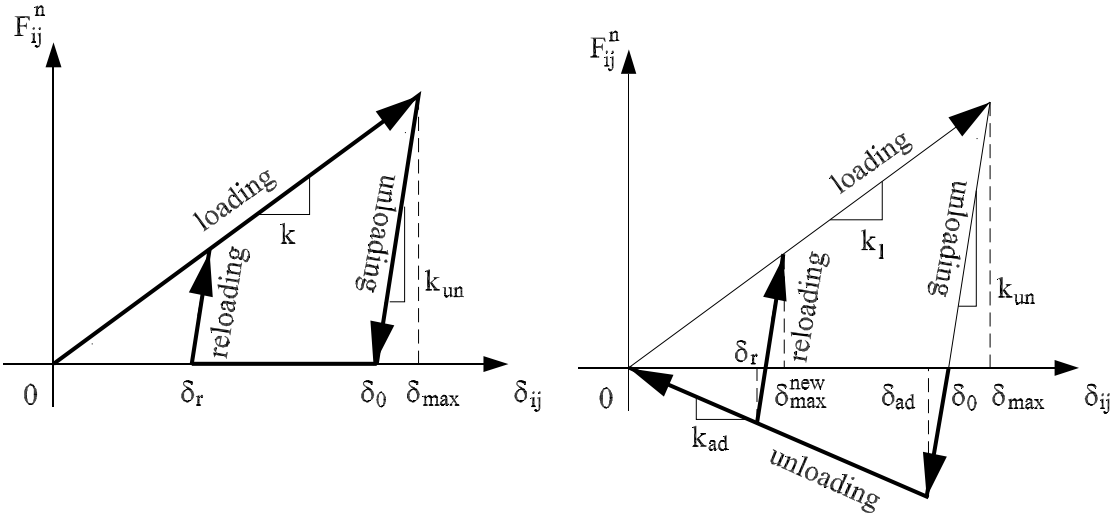
enough to change the direction of motion of the particles, the attractive force is increasing, see [72]. The maximum value δ_{ad} of the attractive force is reached, when the unloading path of slope k_{un} is crossing the decreasing adhesive line of gradient k_{ad}. Applying the similarity of triangles, one can write

$$\frac{\delta_0 - \delta_{ad}}{k_{ad}\delta_{ad}} = \frac{\delta_{max} - \delta_0}{k_l \delta_{max}} \ , \tag{2.48}$$

and the deformation δ_{ad} is

$$\delta_{ad} = \frac{(1-\epsilon_m^2)k_l}{k_l + k_{ad}\epsilon_m^2}\delta_{max} \ . \tag{2.49}$$

Further unloading forces $k_{ad}\delta_{ij}$ are calculated along a negative-gradient line which limits the maximum possible attractive forces between the particles. The reloading effect can start again at any point in between along this third line, i.e. $0 < \delta_r < \delta_{ad}$ and the force increases with the positive gradient k_{un} until reaching the new value of δ_{max}^{new} on the original loading curve. Otherwise, the reloading could not happen and the force keeps decreasing along the adhesive line until $\delta_{ij} = 0$, the case that the two particles do not overlap anymore.



(a) repulsive force model (b) attractive force model

Figure 2.5: Schematic diagram of the loading and un/reloading hysteresis loop of contact

• Tangential contact forces

In this part, the implementation of the tangential forces is described. The tangential forces are active at contacts where the relative tangential velocity of the particles is not zero. The normal and tangential components of the relative velocity $\mathbf{v}_{ij}$ of the two particles in the normal and tangential directions, $\mathbf{n}$ and $\mathbf{t}$, are

$$v_{ij}^n = \dot{\delta}_{ij} = (\mathbf{v}_i - \mathbf{v}_j)^T \mathbf{n} \ , \tag{2.50}$$

$$v_{ij}^t = (\mathbf{v}_i - \mathbf{v}_j)^T \mathbf{t} + \Omega_i r_i + \Omega_j r_j \ , \tag{2.51}$$

$$\mathbf{t} = \frac{\mathbf{v}_{ij}^t}{v_{ij}^t} \ , \tag{2.52}$$

where Ω_i and Ω_j denote the angular velocities of particles i and j, respectively and they appeared in Fig. 2.4a. If the tangential velocity in the shearing plane of contact is equal to zero in the beginning of a contact, the contact is head on or *normal*, otherwise it is shearing or *tangential*. Here the tangential force $\mathbf{F}_{ij}^t$ is connected to the normal force by the Coulomb laws of friction, namely

$$F_{ij}^t = \begin{cases} F_{\text{static}}^t \leq \mu_s F_{ij}^n & , \ v_{ij}^t = 0 \ , \\ F_{\text{dynamic}}^t = \mu_d F_{ij}^n & , \ v_{ij}^t \neq 0 \ , \end{cases} \tag{2.53}$$

where μ_s and μ_d are the static and dynamic friction coefficients, respectively. The '$\leq$' sign that appears in Equation (2.53) means that in the case of static friction, F_{ij}^t is just compensating the unknown external shearing force F_{ij}^{ext} applied to the contact, so that $v_{ij}^t = 0$. If $F_{ij}^{ext} > \mu_s F_{ij}^n$, one enters the dynamic friction regime where $F_{ij}^t = \mu_d F_{ij}^n$ applies, see e.g. [96]. The discontinuity at zero velocity is considered as a disadvantage of the Coulomb law.

In general, the static friction is always greater than the dynamical sliding friction, i.e. $\mu_s > \mu_d$. For simplicity, the coefficient of static and sliding friction might be assumed equal. Therefore, the tangential shearing force can be expressed in terms of Coulomb's law as

$$\mathbf{F}_{ij}^t = -\mu_d |F_{ij}^n| \mathbf{t} \ . \tag{2.54}$$

By direct substitution of Equations (2.36) and (2.54) into Equation (2.35), the contact force that describes this model, see Fig. 2.6a, can be written as

$$\mathbf{F}_{ij} = (k_p \delta_{ij} + c_p \dot{\delta}_{ij}) \mathbf{n} - \mu_d |F_{ij}^n| \mathbf{t} \ . \tag{2.55}$$

Substitution of $\mathbf{F}_{ij}$ in Equations (2.33) and (2.34) will yield the resultant contact force and torque acting on particle i due to the particle-particle interaction.

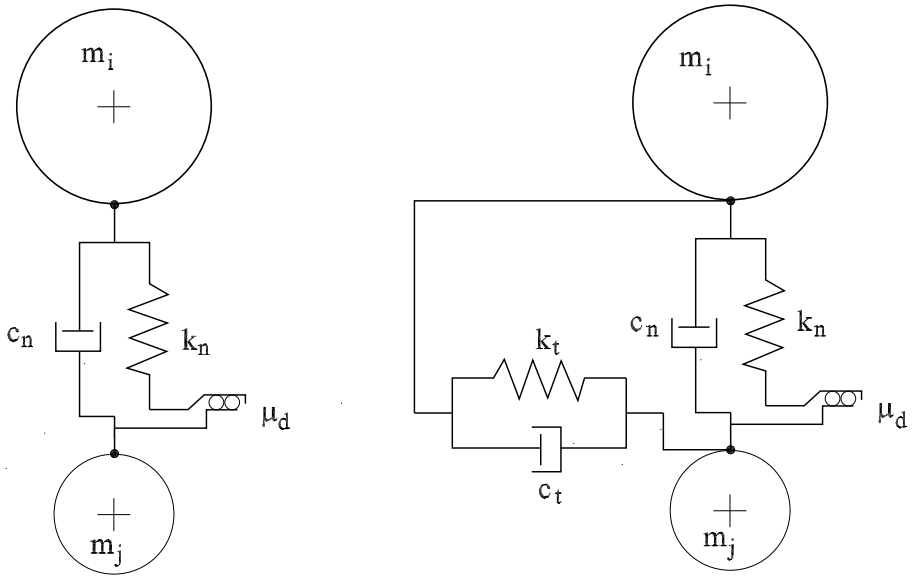
Since Coulomb friction is a discontinuous force model, some research works made some adjustments to the model to avoid this in the simulations, see e.g. [20, 61]. These research works introduce the shear damping to find the viscous friction force. This force which is proportional to the tangential velocity between the two colliding particles takes the form

$$\mathbf{F}_{ij}^t = -c_t \mathbf{v}_{ij}^t \ , \tag{2.56}$$

where c_t is the shearing damping constant of the contact. Some other models consider the both Coulomb and the viscous effects in determining the tangential force [63]. In this model, the penalty method of spring-damper model is applied twice in the normal and transverse direction of contact, see Fig. 2.6b. The contact model is described by a set of force-displacement relationships

$$f_n = k_n \delta_n + c_n \dot{\delta}_n \ , \tag{2.57}$$

$$f_t = \mu_d f_n + k_t \delta_t + c_t \dot{\delta}_t \tag{2.58}$$



(a) penalty model in normal direction (b) penalty model in both directions

Figure 2.6: Spring-dashpot contact model in normal and/or tangential direction with a shearing frictional force using Coulomb's law in the tangential plane of contact

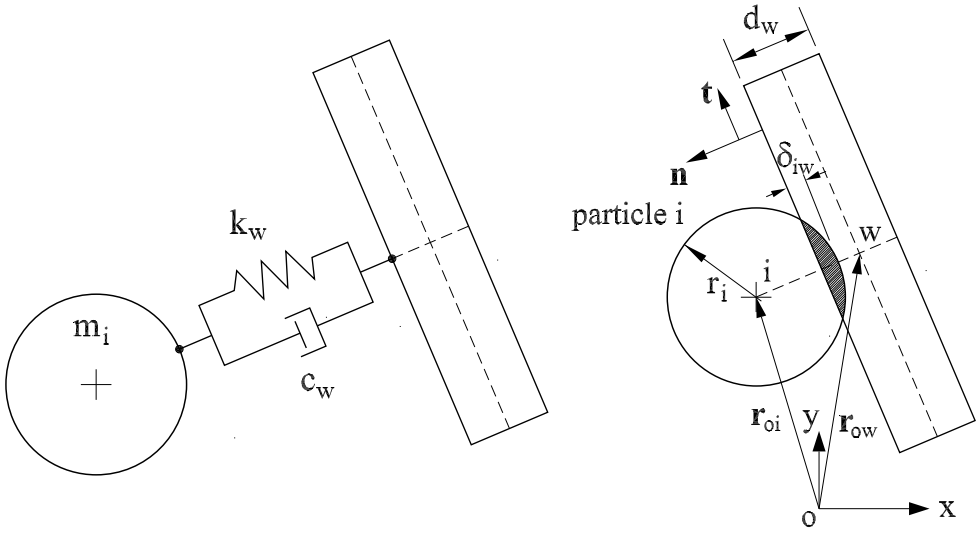
for the normal and tangential contact forces f_n and f_t respectively, where δ_n and δ_t are the contact distances (overlaps) in the normal and tangential directions, $\dot{\delta}_n$ and $\dot{\delta}_t$ are the relative velocities in these directions which have been given in Equations (2.50) and (2.51). Starting from the time t_0 at which the contact was first established, the distance δ_t over which the tangential spring is stretched is

$$\delta_t = \int_{t_0}^{t} v_{ij}^t(\tau) d\tau \ . \tag{2.59}$$

2. **Particle-to-wall contact**
The penalty principle of spring-damper model is also applied here in finding the forces between the particles and the walls, see Fig. 2.7a. Similarly as we did in the particle-particle contact, the particle-wall contact needs to know the amount of overlapping with the wall during contact which can be given as

$$\delta_{iw} = (r_i + \frac{d_w}{2}) - (\mathbf{r}_{oi} - \mathbf{r}_{ow})^T \mathbf{n} \ , \qquad \mathbf{n} = \frac{\mathbf{r}_{oi} - \mathbf{r}_{ow}}{|\mathbf{r}_{oi} - \mathbf{r}_{ow}|} \ , \tag{2.60}$$

where δ_{iw} is the overlap between the particle i and the wall, d_w is the wall thickness, $\mathbf{r}_{oi}$ and $\mathbf{r}_{ow}$ are the position vectors of the center point of ball i and the wall w, respectively, and $\mathbf{n}$ is the normal unit vector to the wall surface in the direction of reflection, i.e. the normal unit vector from the wall to the center of the particle i, see Fig. 2.7b. As well as the particle becomes in touch with the wall surface, an opposite contact force is produced. This normal force represents the summation of



(a) spring-dashpot model (b) vector description of contact

Figure 2.7: Particle-to-wall interaction

the elastic spring and viscous damping forces assuming the linear Hertz model of contact and is given by

$$\mathbf{F}_{iw}^{n} = (k_w \delta_{iw} + c_w \dot{\delta}_{iw})\mathbf{n} \ , \tag{2.61}$$

where k_w, c_w are the spring stiffness and damping coefficient with the wall and $\dot{\delta}_{iw}$ denotes the relative velocity between the particle i and the wall. In case of moving or vibrating walls, the relative normal velocity $\dot{\delta}_{iw}^{n}$ will take the form

$$\dot{\delta}_{iw}^{n} = \dot{\delta}_{iw}\mathbf{n} = \mathbf{v}_i^n - \mathbf{v}_w^n \ , \tag{2.62}$$

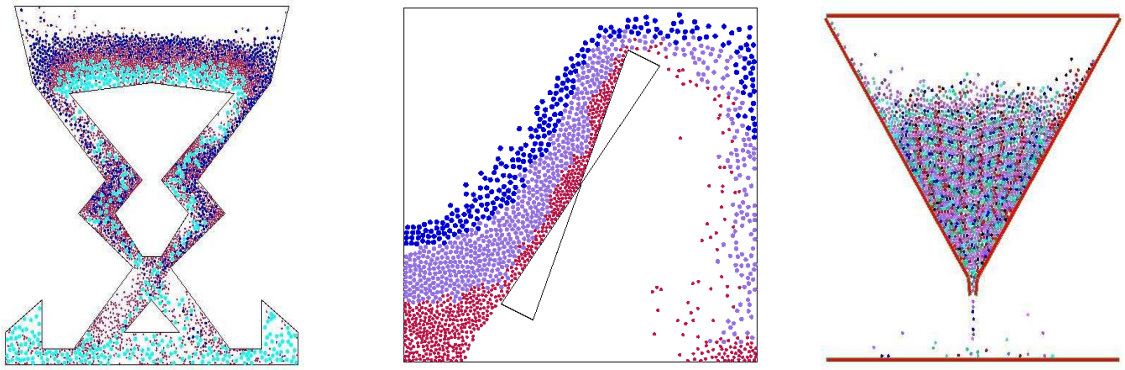
where $\mathbf{v}_i^n$ and $\mathbf{v}_w^n$ are the normal velocities of particle i and the wall, respectively. Otherwise, the wall is stationary ($\mathbf{v}_w^n = 0$) and the relative velocity is equal then the particle velocity itself in the normal direction.

The amount of viscous damping during contact is controlled by the coefficient c_w. The energy dissipation will increase as c_w becomes larger and the kinetic energy will be reduced after contact due to velocity reduction the case which will drive the system to damp faster. The shearing force can be taken also into consideration in the tangential plane of contact. Using the simplest form of Coulomb's law, the friction force $\mathbf{F}_{iw}^{t}$ with the wall can take the form

$$\mathbf{F}_{iw}^{t} = -\mu_d |F_{iw}^{n}| \mathbf{t} \ , \tag{2.63}$$

which points in the opposite direction of the shearing velocity in the tangential plane of contact. Increasing the friction coefficient will help in damping the particles in the tangential direction. The adhesion effect between particles, which will be discussed later, has also a great influence on the overall damping of the colliding particles. Different examples with different number of particles and boundary conditions are

performed, see Fig. 2.8. Some of them shows the particles contact with fixed and/or rotating walls for different levels of damping.



(a) zig-zag pattern (3893) (b) mixing process (1567) (c) outflowing hopper (1911)

Figure 2.8: Particle-wall contact with fixed and movable walls

2.3 Sorting algorithms and neighbor list computations

Since our work is focusing on studying the contact forces of particular materials of a granular medium, the simulations are restricted to particles within some small region surrounding the original particle i. This is typically implemented by regarding special regions of the system using a certain *surrounding distance* beyond which the neighborhood calculations and particle interactions are ignored. In a short-range force of MD simulation, the vast majority of computation time is spent in evaluating the contact force between colliding bodies and updating the neighboring lists, see Equation (2.33). In most cases, the time integration typically requires only not more than 7-10% of the total time of computation [79]. Evaluating the force sums efficiently requires knowing which particles are within the surrounding distance at every timestep. These particles are identified as *neighbors*. The key is to minimize the number of neighboring particles that must be checked for possible interactions since calculations performed on particles outside their surrounding distances will be wasted computation.

There are some basic techniques used to speed up simulations of particulate systems. Representatives of these techniques are the *Verlet* approach and the *linked linear list* approach. In this case, the aim is to construct, as efficiently as possible, a list of potential collisions between pairs of particles. The Verlet approach depends on the particles within the cutoff distance while linked linear list approach mainly uses the bounding boxes constructed around each particle. As the bounding boxes of two particles intersect, the

particles are assumed to be neighbors.

For each method the neighboring particles are stored in a neighbor data-structure after the pre-sorting has been done. Once the neighbor list is built, examining it for possible interactions is much faster than checking all particle combinations in the system. Since the update of these list can be optimized, the essential calculation operation for collision detection can be reduced from $O(N^2)$ to an order proportional to the size of the system, i.e. $O(N)$, where N is the number of particles in the system.

2.3.1 Verlet approach

In 1967, Verlet originally proposed the idea of using the neighbor lists and he is considered a pioneer in this field [113]. In order to reduce the computational effort, Verlet thought in a way to avoid to do calculations for all particles in the system. Particles which are at a large distance from each other do not interact. To utilize this property, the Verlet neighbor lists (VL) are built [5, 88].

As shown in Fig. 2.9 a circular domain (spheres in the spatial case) is chosen around each particle i in the system. The radius of these circles is considered as the surrounding Verlet radius of the Verlet approach outside which all neighborhood calculations are ignored. This radius of the Verlet circles r_v is usually chosen as five times the maximum particle radius r_{max} in the system, i.e.

$$r_v = 5r_{max} \ , \text{ or in this case } \quad r_v = 5r_1 \ . \tag{2.64}$$

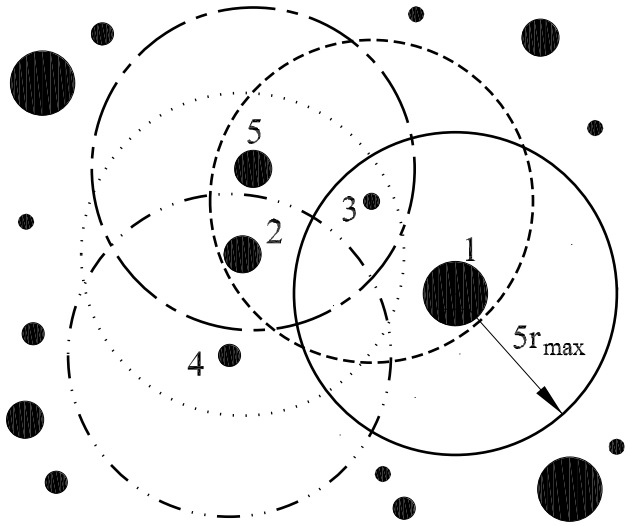


Figure 2.9: Verlet particle circles with radius $r_v = 5r_{max}$

The radius of the Verlet circle can be extended to cover wider zones around the center particle. This optimal extension depends on the density of the whole system, the particle

radii distribution and on the speed of the particles in the particular simulation. Increasing the cutoff radius r_v will increase the number of neighboring particles in the list and reduce the frequency for which the list will need to be updated. On the other hand, the number of force calculation operations and detection along with the contact calculation time will increase.

Particles inside these enclosing circles are considered as neighbors to the original particle i in the center of these Verlet circles. The neighbor list contains all the particles which are nearly within the cutoff radius of each particle. To avoid double counting in the force summation only neighbors where $j > i$ are stored. In other words, particles of lower numbers than the particular particle have not to be checked as no pair needs to be tested twice.

To identify the particles which belong to each Verlet circle, Fig. 2.10 shows the domains of these circles separately with their members inside. As shown in this figure, e.g. the central particle 1 has particle 3 as a neighbor while central particle 3 has the neighbors 1, 2 and 5 within its domain and so on.

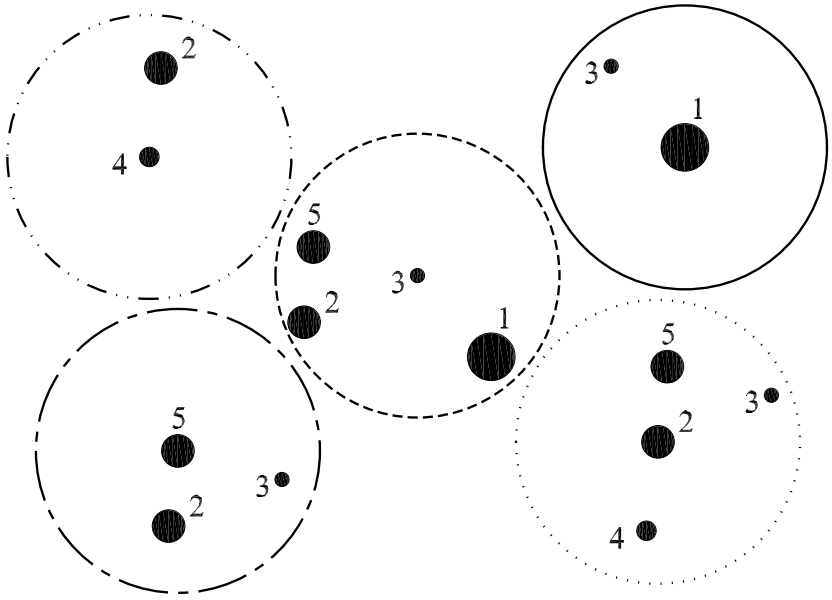


Figure 2.10: Verlet circles depicted separately with their members inside

Examining all pair separations in the granular system of N particles is computationally expensive. To determine if contact occurs between any two particles, contact could be

checked between all possible particle pairs. Suppose that

$$\begin{aligned}
&\text{particle (1) needs } \rightarrow \text{ } (N-1) \text{ contact checks ,} \\
&\text{particle (2) needs } \rightarrow \text{ } (N-2) \text{ contact checks ,} \\
&\text{particle (3) needs } \rightarrow \text{ } (N-3) \text{ contact checks ,} \\
&\ldots \\
&\ldots \\
&\text{particle } (N-3) \text{ needs } \rightarrow \text{ (3) contact checks ,} \\
&\text{particle } (N-2) \text{ needs } \rightarrow \text{ (2) contact checks ,} \\
&\text{particle } (N-1) \text{ needs } \rightarrow \text{ (1) contact check .}
\end{aligned} \tag{2.65}$$

Therefore, the total number of contact checks is

$$\begin{aligned}
\sum_{i=1}^{N-1}(N-i) &= (N-1)+(N-2)+(N-3)+...+3+2+1 \\
&= \frac{1}{2}N(N-1) \ ,
\end{aligned} \tag{2.66}$$

which shows that the number of necessary arithmetic operations is of order $O(N^2)$. Some advantages result from the use of the lists of nearby pairs of particles and hence, the speed of the simulation program is improved. Updating the lists is necessary to be done every several time steps and they can be used for a few time steps without reconstruction. Since in practice, the particles move just a small distance at each time step, a large fraction of the neighbors remains the same during this time step. Therefore, it seems wasteful to update the list at every time step of simulation. The update frequency depends on the velocity of the particles, the density of the granulars and the size of the Verlet circles r_v around the particles. Refreshing the neighbor list e.g. every e.g. 20th time step could be acceptable for relatively dense systems of granular media.

The example in Fig. 2.11 shows the neighbor lists of five particles (N=5). The contact detection and force calculation has to be done on the particle pairs stored in the list. To be neighbors is not necessary to be in contact. Using Verlet lists will reduce the number of contact checks from ten pairs in the original system to five pairs in Verlet neighbor contact detection scheme.

2.3.2 Linked linear list approach

The linked linear list approach (LLL) is a frequently-used and efficient searching approach in determining the neighbor particles especially for polydisperse granular systems. Using this approach, one can reduce the time consumption from $O(N^2)$ to $O(N)$ what will accelerate the simulation.

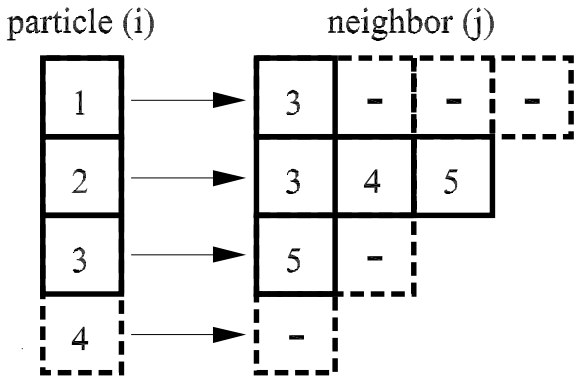


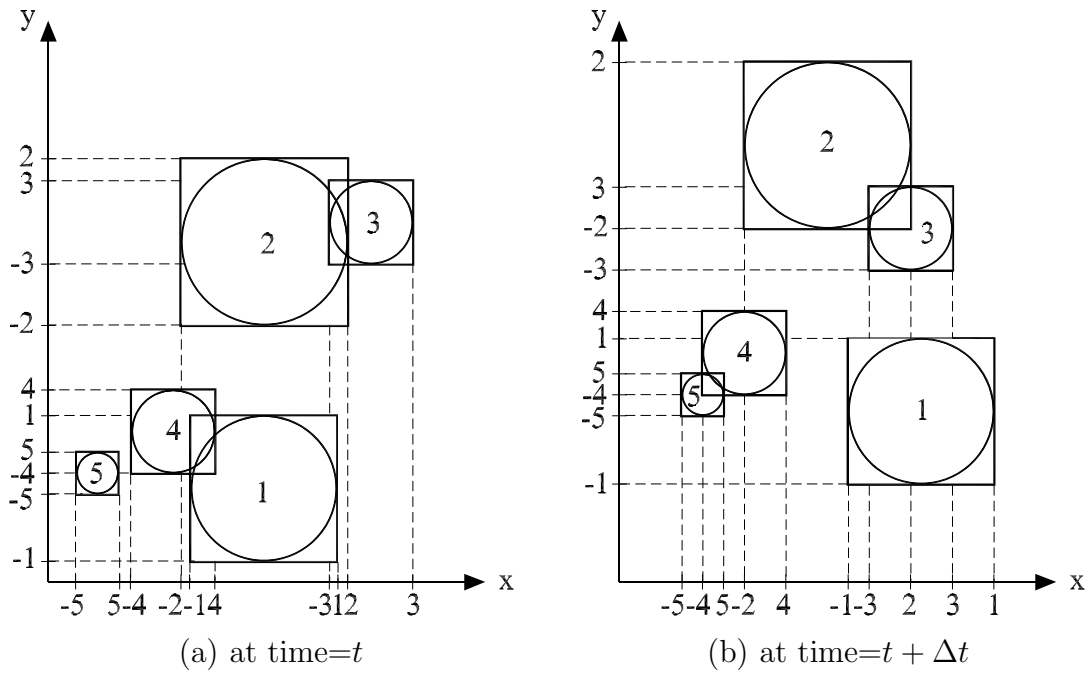
Figure 2.11: Searching algorithm / particle storage in the Verlet-neighbor lists for a system of $N = 5$ particles

The idea of using this method is to enclose the particles with *bounding boxes* in such a way that the particles fit inside these tangential boxes. The edges of the bounding box are aligned parallel to the system axes $x, y,$ and z, see Fig. 2.12. Finding the neighbors is independent on the particle shape since the particle will be enclosed by a bounding box. The disparity in the particle sizes will just lead to different sizes of the surrounding boxes and there polydisperse as well as momodisperse systems can be simulated efficiently.

• Creating linear lists

After the bounding boxes are laid around the particles, they are projected orthogonally to the coordinate axes of the system. The beginning and ending limits of the bounding boxes should be marked and stored in sequence. For particle i let the negative sign $-i$ denote the starting extreme and the positive sign denotes the ending extreme of the bounding box, i.e. $(-i \; +i)$, see Fig. 2.12. Two lists have to be created for the two dimensional case while three lists are required in the case of 3D. Since the list has to store the beginning and ending points of the particle, the length of each list is equal to twice the number of particles in the system, i.e. $2N$. The overlapping between the projections of the particles i and j along a certain axis occurs if the starting $-i$, ending $+i$ or both of particle i is in between the starting $-j$ and ending $+j$ of particle j. If the two particles do overlap along one axes, it is not necessary to have an actual overlapping in the space. The potential spatial overlap occurs when the particles overlapped over all individual axes of the system.

Figure 2.12a shows five particles surrounded by their bounding boxes. The state of these particles is captured at two successive time steps, i.e. t and $t + \Delta t$ and their projections are depicted in Fig. 2.13. Since 2D case, the limits of the bounding boxes are stored at the time t as $[-5 \; +5 \; -4 \; -2 \; -1 \; +4 \; -3 \; +1 \; +2 \; +3]$ along the x direction and $[-1 \; -5 \; -4 \; +5 \; +1 \; +4 \; -2 \; -3 \; +3 \; +2]$ along the y direction. From the sequence of these two lists, it can observed that the particles of the group {{1,2},{1,3},{1,4},{2,3},{2,4}} overlap together over the x-axis while the particles of the group {{1,4},{1,5},{2,3},{4,5}}



(a) at time=t (b) at time=$t + \Delta t$

Figure 2.12: Bounding boxes with starting and ending limits

overlap over the y-axis. The actual intersection between the bounding boxes is represented in the intersection between these two groups of particle pairs which will appear in the two pairs 1/4 and 2/3.

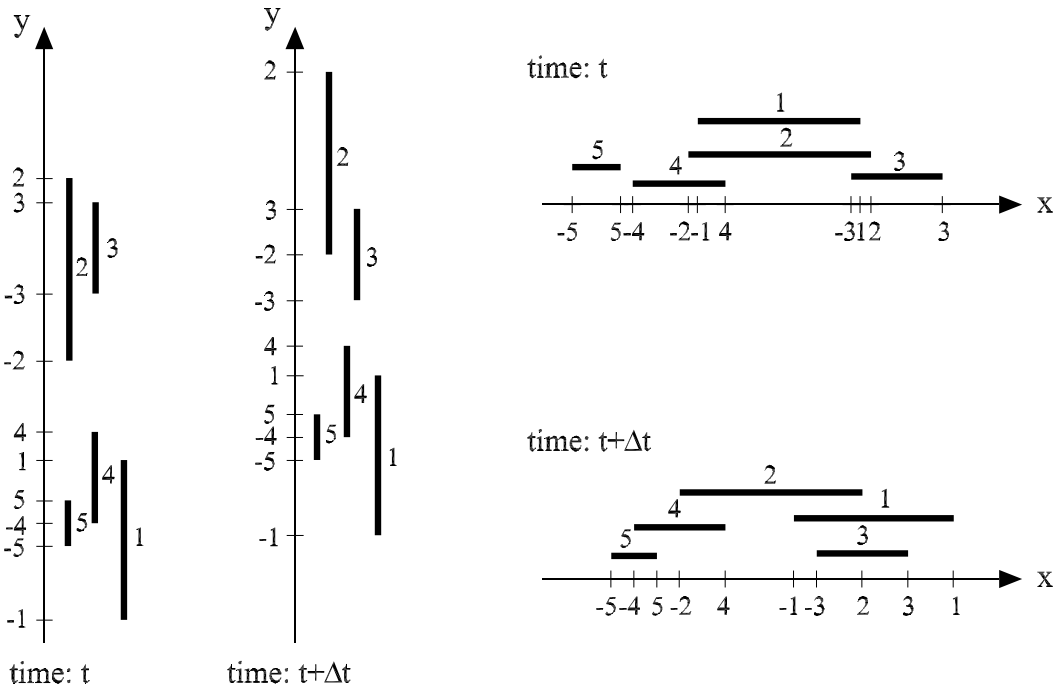


Figure 2.13: Projection of the bounding boxes on the coordinate axes at two successive times

• **Updating linear lists**

Particles collide and change their locations frequently during simulation. This change will create a dynamic list that can be updated at each time step of computation. But, although these lists have to be frequently updated, the essential computational time can be reduced to be proportional to the size of the granular system, i.e. $O(N)$. Since slightly changes happen on the locations of the colliding particles during one time step, there is no need to damage the old neighbor list and create another new one from scratch. In this case, updating the old list with very few changes of particle location is the proper way to save time and speed up the simulation. That corresponds to resort again the nearly sorted list. The changes are usually some permutations where the begin or end of one particle jumps over the begin or end of the next one.

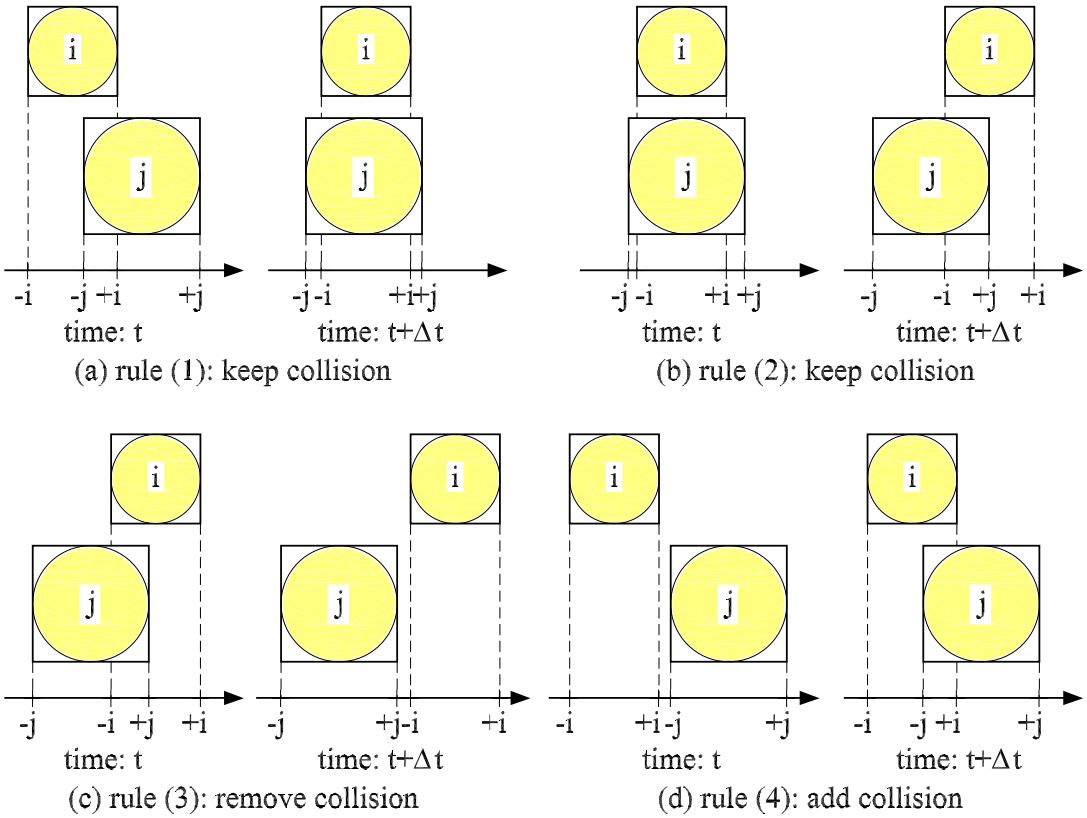


Figure 2.14: Exchange rules for updating the linked linear list at two successive times of simulation

The update can be simply done by traveling through the lists sequentially and checking for any permutations. If the order of extreme ends of two particles is unchanged, the collision status will remain the same. In other words, if $[-i \ -j \ +i \ +j]$ is the status at time t and $[-i \ +j \ -i \ +j]$ is the status at $t + \Delta t$, the two particle i and j will keep on overlapping. During resorting, some tests depending on some specified *exchange rules* have to be performed for every particle exchange [98], see Table 2.1. A schematic

depiction is shown in Fig. 2.14 for these four rules. that can be summarized as

► **rule (1)**: The beginnings of the two particles are switched, which means that the bounding boxes were in overlap at time t and keep in overlap along the specified axis at time $t + \Delta t$, i.e. $[-\underline{i}\ -\underline{j}\ +i\ +j] \Rightarrow [-\underline{j}\ -\underline{i}\ +i\ +j]$, where exchanged particle-ends are denoted by the *underline* character, see Fig. 2.14a.

► **rule (2)**: The ends of the two particles are switched, which means that the bounding boxes were in overlap at time t and keep in overlap along the specified axis at time $t+\Delta t$, i.e. $[-j\ -i\ +\underline{i}\ +\underline{j}] \Rightarrow [-j\ -i\ +\underline{j}\ +\underline{i}]$, see Fig. 2.14b.

► **rule (3)**: The beginning of particle i and the near ending of particle j are switched, which means that the bounding boxes were in overlap at time t and the overlap has been removed from the specified axis at time $t+\Delta t$, i.e. $[-j\ -\underline{i}\ +\underline{j}\ +i] \Rightarrow [-j\ +\underline{j}\ -\underline{i}\ +i]$, see Fig. 2.14c.

► **rule (4)**: The ending of particle i and the following beginning of particle j are switched, which means that the bounding boxes were not in overlap at time t and become to be in overlap along the specified axis at time $t+\Delta t$, i.e. $[-i\ +\underline{i}\ -\underline{j}\ +j] \Rightarrow [-i\ -\underline{j}\ +\underline{i}\ +j]$, see Fig. 2.14d.

Table 2.1: Rules for updating and resorting the list of the bounding boxes in one dimension. The negative sign (-) denotes the beginning edge of the bounding box while the positive sign (+) denoted its ending one

Rule no.	Order at t	Order at $t + \Delta t$	Intersection state
(1)	$-i\ -j$	$-j\ -i$	keep overlap
(2)	$+i\ +j$	$+j\ +i$	keep overlap
(3)	$-i\ +j$	$+j\ -i$	remove overlap
(4)	$+i\ -j$	$-j\ +i$	add overlap

For the example which was shown in Fig. 2.13, the three different cases of adding, keeping and removing overlap between the bounding boxes of the particle pairs are summarized in Table 2.2 and in Fig. 2.15 at two successive times of simulation.

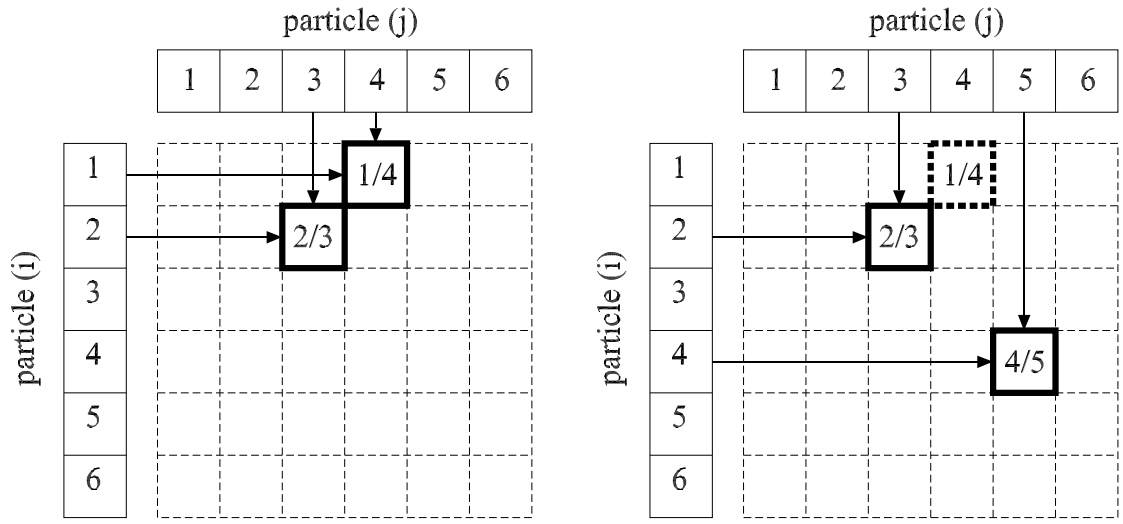
• Comparing VL and LLL approaches

A granular system of 4336 circular particles is studied. The system of particles is polydisperse, i.e. there are different sizes of particle diameters, and consists of 2278 particle with radius 10mm, 1716 particles with radius 15mm and 342 particles with radius 20mm. Each group of these particles is distributed homogeneously inside a circular container at the beginning of the simulation, see Fig. 2.16. As shown in this figure the particles fall under the gravity effect from the middle of the container and hit its bottom. Due to the large impact, these particles start moving upwards from right and left in a round motion with the wall profile until they meet together at the top of the container where they fall

Table 2.2: Application of updating the list of the bounding boxes on the example of three particle pairs shown in Fig. 2.12 and Fig. 2.13

Particle pair	Current state at t		
	x-axis	y-axis	Actual intersection
1/4	overlap	overlap	exist
2/3	overlap	overlap	exist
4/5	no overlap	overlap	not exist

Particle pair	Updating at $t + \Delta t$		
	x-axis	y-axis	Actual intersection
1/4	remove overlap	keep overlap	not exist
2/3	keep overlap	keep overlap	exist
4/5	add overlap	keep overlap	exist

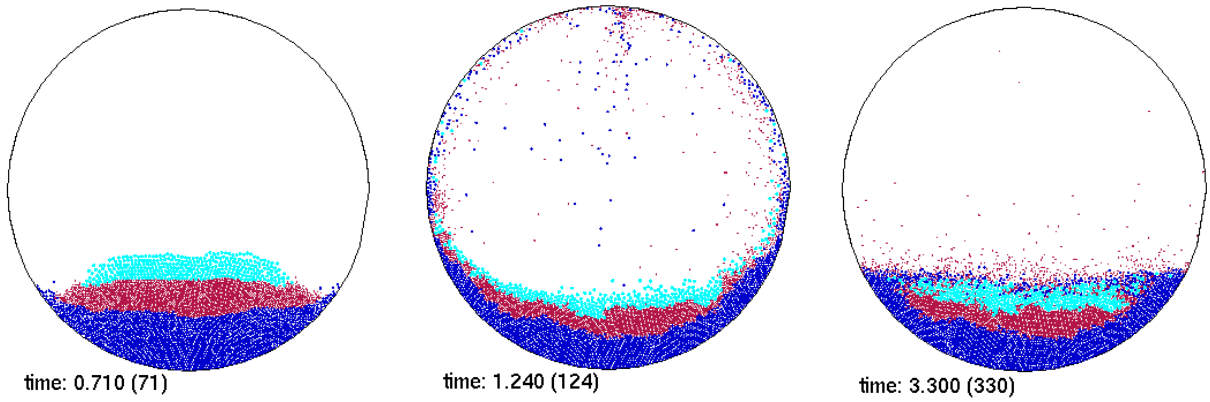


(a) at time=t (b) at time=$t + \Delta t$

Figure 2.15: Storage of colliding bounding box pairs / linked linear list

again as showers until settle down and damp in the bottom of the tank. These particles are divided into five different groups and simulated using the pre-mentioned approaches of the Verlet and the linked linear lists.

The simulations are run for 4 seconds with $\Delta t = 10^{-5}$s using spring stiffness $k_p = 720N/m$ and coefficient of restitution $e_n = 0.2$. For each test the corresponding computational time is observed. The results of the computational time with respect to the number of particles are shown in Fig. 2.17. From this curve, it is observed that the linked linear approach is much faster than Verlet method. The situation becomes worse by using the Verlet method



(a) at $t = 0.71$ s. (b) at $t = 1.24$ s. (c) at $t = 3.30$ s.

Figure 2.16: Snapshots of the simulation of a polydisperse mixture of 4336 circular particles of three sizes at different simulation times

in simulating large systems while it can be used efficiently in dealing with small systems, for more details see [73].

Most of the computation time is spent in creating, updating and sorting the neighbor lists and some other in finding forces and contact detection, while a very small portion of this time is required for solving and integrating the system equations. In the example shown in Fig. 2.16, the collision and force calculation take 82.75% and the updating lists needs 12.7% while the integration and other operations consumes 2.46 % from the overall computation time. For this reason, parallel computational technology becomes a main and important demand to speed up and solve granular media problems.

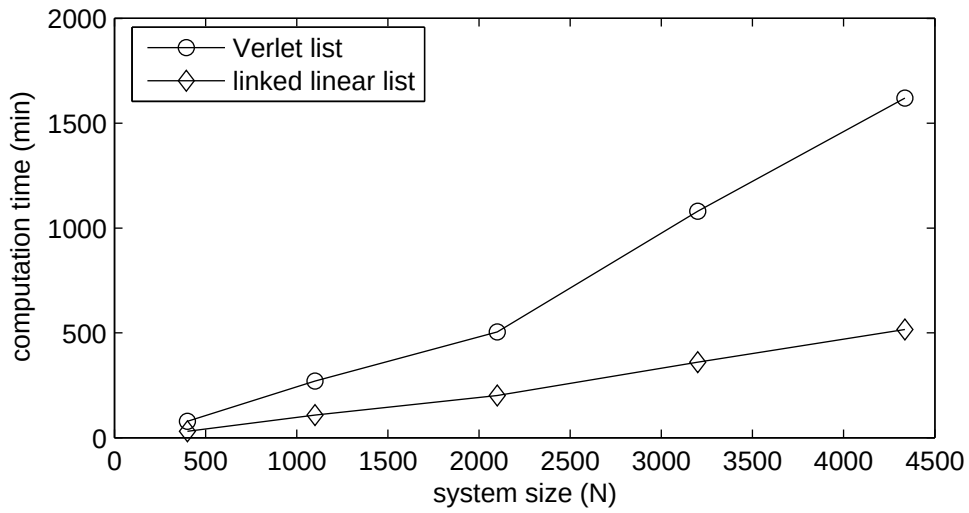


Figure 2.17: Comparing the computational time of VL and LLL methods for different number of particles

2.4 Computational structure of the serial MD code

In this part, we will explain the structure of the molecular dynamics program MOLDYN, originally written by Muth [71]. A general overview about different subrotines will be briefly presented without much details. Only some special parts which are necessary to understand the general communication along with the input and output files will be taken into consideration.

2.4.1 Code initialization

MOLDYN basically consists of different consecutive parts connected to form the general structure of the program, see Fig. 2.18. Variables, vectors, matrices and structures were defined, allocated and set to zero at the very beginning of the program. As an initialization of the code, all required data will be read from the input files and stored in the corresponding data structures, see Appendix A.1. These files are divided into three different categories, the *control-input-file* whose contents is the information about the starting time t_0 and ending time t_{end} of the simulation, the simulation time step Δt, the gravity acceleration, the particles density ρ, the static μ_s and dynamic μ_d friction coefficients, the spring stiffnesses for loading k_l, unloading k_{un} and adhesion k_{ad} contact cases, the damping constants between particles c_p and with the walls c_w and/or the normal coefficient of restitution e_n, the *particle-input-file* which has all information about the particles in the system, e.g. the initial positions, velocities, orientations, radii and the particle state if it is movable or stationary and finally the *wall-input-file* which supplies information about the number of walls, the location of the center point of the walls and the direction of the positive norms toward where the particles should be reflected. It is also possible to define the walls as a segments so that the norm direction of the wall segment ab is quite opposite to the norm direction of the wall segment ba, where a and b are the end points of the wall segment-line ab.

2.4.2 Description of the program

As a small necessary test of the defined simulation time step, the `contact-time` subroutine checks if Δt is small enough comparable with the smallest contact time t_c of the colliding particles in the system, see Equation (2.22), otherwise the program will be interrupted. The mass of the particles can be calculated from the density and the volume of each particle i, i.e. $m_i = \rho_i V_i$, where $V_i = (4/3)\pi r_i^3$ is the volume of the spherical particle of radius r_i. The mass of the particles is very important to estimate the contact time t_c and to determine the particle acceleration, i.e. $\mathbf{a}_i = \mathbf{F}_i/m_i$, see Equation (2.33).

In this stage, the main time loop starts. MOLDYN offers the choice to select between

either the `Verlet-approach` or the `linked-linear-lists-approach`. If the Verlet-approach is in use, the `Verlet-circle` subroutine is then responsible to determine and calculate the verlet circles around the particles as five times the size of the largest particle in the system, see Equation (2.64), otherwise the *linked-linear-lists-approach* will be activated. In this method the location of the bounding boxes should be determined as they are projected on the system axes. These calculations will be performed in the `bounding-box-calculations` subroutine where the particle ends have to be sequentially stored in lists along the different axes of the system.

The neighboring particles are now known and the particle pairs have occupied specific locations in the storage lists. But to be a neighbor is not necessary to be in contact. Therefore, the `collision-fine-test` will determine if these pairs are in contact or not by calculating the overlap between them, see Equation (2.37). If the overlap exists, a flag is passed to the `particle-particle-contact` subroutine to start force calculations. The force calculations are represented in finding the particle-penalty forces of spring and dashpot elements in the normal and tangential directions. In this subroutine different contact modes of loading, unloading, adhesive and frictional forces are taken into consideration, see Equation (2.47).

The computed particle-particle forces are stored and kept to be added later to the wall contact forces. The `particle-wall-contact` subroutine is responsible to calculate the normal and frictional tangential forces by applying the soft particle model and the simple linear Coulomb approach, see Equation (2.61) and Equation (2.63). This force component with the walls is added to the pre-calculated component between particles to get the overall force acting on every particle i in the system. Since the total force acts on particle i is now known, the acceleration $\mathbf{a}_i$ of the particle is directly calculated by dividing it by mass. In this stage the equations of motion of all particles are ready to be solved. The `integration-scheme` subroutine offers three different choices of Verlet integrators to be used, i.e. original Verlet, velocity Verlet and leap-frog Verlet, see Section (2.1.1). Using the current acceleration at time t and the current and old particle position at time $t - \Delta t$, the original Verlet integrator can explicitly solve the particle equations of motion and get the new positions without even need to use the particle velocity, see Equation (2.3).

Once the new positions and orientations are known, the new state should be recorded on specific output files using the `record-state-and-visualization` subroutine. These files are read by some animation browsers, e.g. ANIM and XBALLS [12, 60], in order to be visualized. Monitoring the particle collision during simulation is very useful to judge if the simulation proceeds in the right way, otherwise it should be terminated. Calculating the total energy dissipation of the system during simulation is also another indicator about the accuracy and acceptability of the contact calculation results. Since the motion of the particles is very small within two successive time steps, the presentation of the output data will be very slow during animation. Therefore, a large amount of this data will not

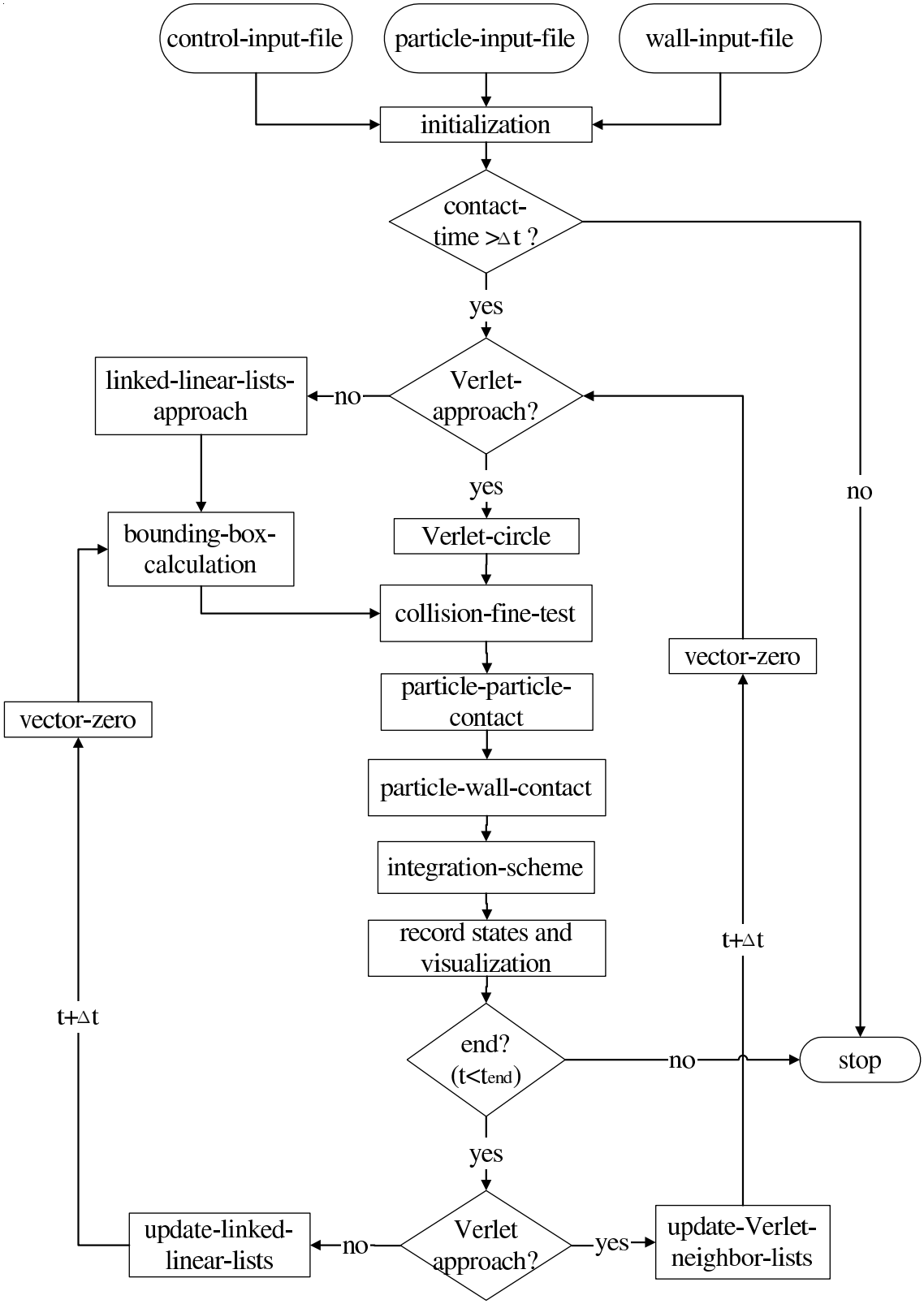


Figure 2.18: Flowchart of the program MOLDYN [71]

be saved but just the data at every certain output time steps, i.e. the simulation time step could be 10^{-5}s while the output time step be like 10^{-3}s. The output time step should be carefully selected to make the animation more fast and not to create any annoying discontinuity in the particle motion during visualization in the same time.

At the end of every time step, the program check for the end time t_{end}. If $t \geq t_{end}$, the program will be terminated and all variables, vectors, matrices and structures will be deallocated. If $t < t_{end}$, the program will proceed to the next time step and all neighboring lists will be updated. Refreshing neighboring lists by adding or removing collisions depends on which neighboring sorting approach is used, i.e. `update-Verlet-neighbor-list` or `update-linked-linear-lists` subroutines. The collision detection and lists updating are the most time consuming part in the simulation. Usually, the old lists will not be lost since it will be used in the next time step. Since these lists are mostly or nearly sorted from the previous time step, these subroutines just run over them to add or remove the new collisions wherever needed.

The MOLDYN is very flexible program in use and can support different choices in solving planar and spatial granular system. The subroutines are written in general form to cover the two and three dimensional problems. Moreover, this program can solve not only the spherical or round-shaped particles, but also the general shape of polygonal particles with concave and convex collisions. The variety and availability in using different damping and adhesion effects and applying different models of frictional approaches give it a great importance in solving efficiently different systems in granular media.

Chapter 3

Parallelizing Molecular Dynamics Using Spatial Decomposition

Computer simulations of technological relevance became possible only in the past decades due to the enormous advances in computer technology. This progress in computer technology contributed much to the scientific interest in granular matter as a subject of research in physics and engineering [85].

The discrete element approach is widely applied for simulation of large, complex, irregular and data-parallel many-particle systems in many areas in physics and engineering and is considered as a useful deterministic technique of granular simulation. The mechanically correct description and simulation of contacts between mechanical bodies still belongs to the most challenging and computation-time intensive questions. While, by simple methods, a relatively low number of particles can be already computed with sufficient accuracy and acceptable computation times, there remain difficult and interesting problems as soon as elastic deformations, complicated particle geometries or a huge number of particles have to be considered.

In recent years there has been considerable interest in devising parallel discrete element algorithms, see e.g. [38, 41, 102, 110]. The natural parallelism in these algorithms is, that the force calculations and the trajectory updates can be done simultaneously for all particles in the system. In this part of the work, existing contact algorithms are extended and modified in such a way that modern high performance computers can be utilized for their parallel evaluation. Appropriate load distribution schemes and different strategies for a concurrent and distributed collision detection and contact force computation have to be developed, implemented and investigated.

Concerning engineering applications we note that an important factor which affects the success of the numerical procedure is how much one has access to a computer system which is powerful enough to handle the problem of interest. Therefore, we will do our

simulation using parallel high performance computers, which are developing rapidly in the ongoing computer revolution. So, the available current algorithms need to be redesigned to suit the new architectures of modern parallel computers. These modified algorithms have to be run on different machines simultaneously and the results will be collected by a master one.

3.1 High performance computers

High performance computers (HPCs) are computers that can perform multiple, complex digital operations within seconds. The term *high performance computing* often refers to the use of parallel computers (sometimes supercomputers) and computer clusters, that is, computing systems comprised of multiple processors linked together in a single system with commercially available interconnects. This is in contrast to mainframe computers, which are traditionally monolithic in nature.

High performance computing can be performed in a number of ways. The scope and effect of HPC technologies depends on several factors, e.g. , how the computing capability is structured, how the computing capability is used, and what is the desired result of the computing capability. The first high-speed computers developed in the late 1950s and the early 1960s, such as the IBM 704 and the IBM STRETCH, were single computers performing a series of sequential, arithmetic functions (one operation per one instruction in a sequential order). In the late 1960s and into the 1970s, improvements in computing memory capacity and speed led to developments in which repetitive operations could be undertaken per instruction. An HPC which performs one operation per one instruction uses scalar speed; an HPC which performs repetitive operations per single instruction uses vector speed, see [65]. There are two main types of HPCs: vector computers and parallel computers.

• Vector computers
This type of computers as used since the 1970s is conceptually a parallel computer that can be classified as SIMD machine (single instruction, multiple data). They are very efficient for processing vectorial data and pipelining, i.e. data-parallel operations, see e.g. [85, 87]. The computing application, the number of processors needed, the time, cost, and other factors all play a role in how HPCs are used.

• Parallel computers
A parallel computer - as used in our work- is simply a collection of processors, interconnected in a certain fashion to allow coordination of their activities and exchange of data. Traditionally, software which is written for *serial* computation is run on a single computer having a single Central Processing Unit (CPU). As the problem is broken into a discrete series of instructions, they are executed one after another in such a way that only one

instruction may execute at any moment in time, see Fig. 3.1.

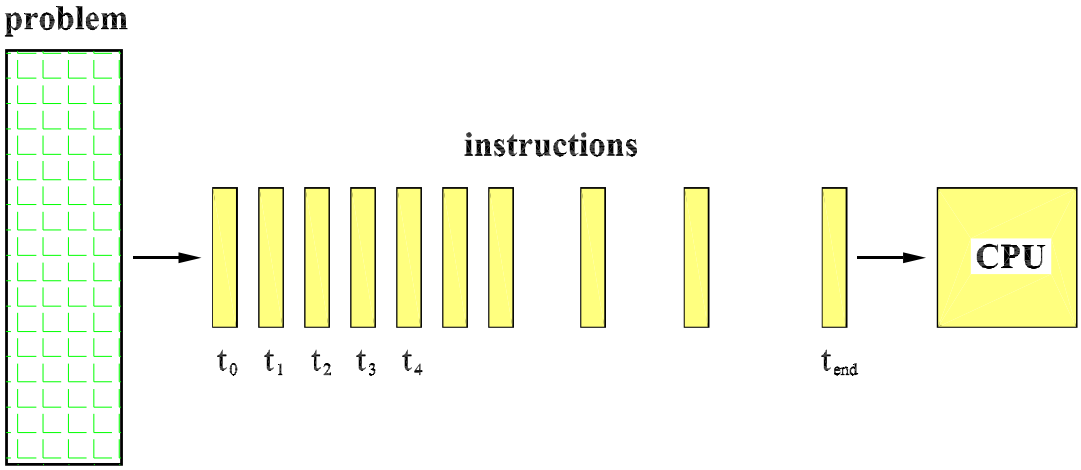


Figure 3.1: Graphical representation of serial computation

On the other hand, *parallel* computing is the simultaneous use of multiple compute resources to solve a computational problem. The problem is usually broken into parts that can be solved concurrently using multiple CPUs. Each part of the problem is further broken down to a series of instructions which are executed simultaneously on different CPUs, see Fig. 3.2. The compute resources can include a single computer with multiple processors, an arbitrary number of computers connected by a network or a combination of both.

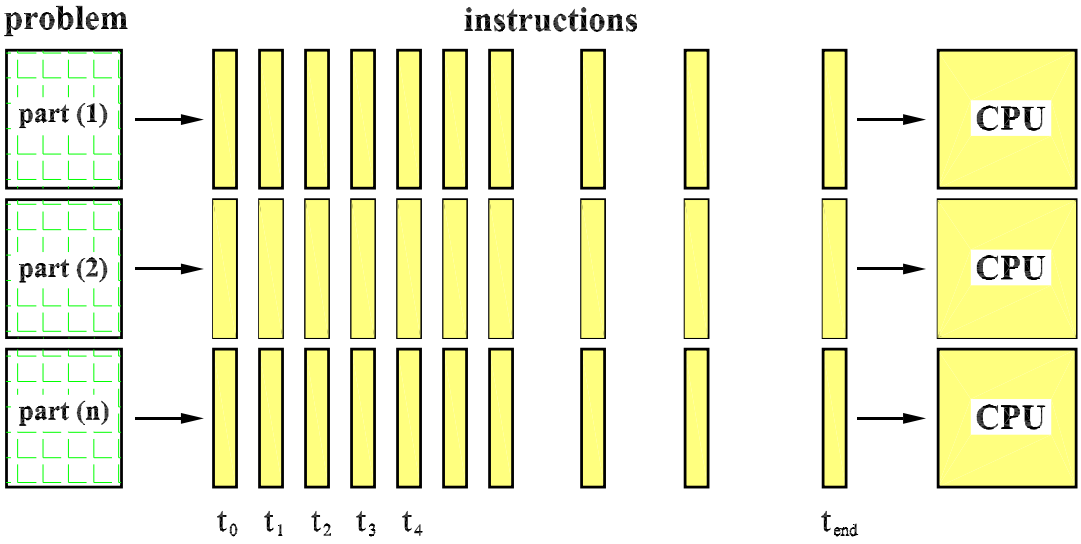


Figure 3.2: Graphical representation of parallel computation

There are different ways to classify parallel computers. One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy [31]. Flynn's taxonomy dis-

tinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction and Data. Each of these dimensions can have only one of two possible states: Single or Multiple. Let *SI* denote the Single Instruction Stream, *MI* is the Multiple Instruction Stream, *SD* is the Single Data Stream and *MD* is the Multiple Data Stream, the Table 3.1 below defines the four possible classifications according to Flynn, see e.g. [10].

Table 3.1: Classification matrix of the parallel computers according to Flynn's taxonomy

Data/ Instruction	**SD**	**MD**
SI	**SISD** Single Instruction, Single Data	**SIMD** Single Instruction, Multiple Data
MI	**MISD** Multiple Instruction, Single Data	**MIMD** Multiple Instruction, Multiple Data

Parallel computers are fundamentally divided into two main types according to their memory architecture: the shared memory parallel computers and the distributed memory parallel computers.

3.1.1 Shared memory architecture

The single computer with multiple internal processors is known as a *Shared Memory Multiprocessor*. This multiprocessor is a natural extension of the single processor of the conventional computer and has the ability to access all memory as global address space, see Fig. 3.3. This means that any processor can readily have access to any memory location without any need for copying data from one memory to another.

In a shared memory scheme, the multiple processors can operate independently but share the same memory resources. Altering values at a given memory location should be done carefully since cached copies of such variables also have to be updated for any processor using that data and so the changes in a memory location effected by one processor are visible to all other processors. Based upon the memory access times and the identity of the different processors, shared memory computers can be divided into two main classes: the *Uniform Memory Access* (UMA) and the *Non-Uniform Memory Access* (NUMA). The major disadvantages of the shared memory multiprocessor are in the difficulty of hardware implementation that can achieve fast access to all shared memory locations, the responsibility of the programmer for synchronization constructs that insure correct access of global memory and the high cost and complexity to design and produce shared memory machines with an ever increasing numbers of processors.

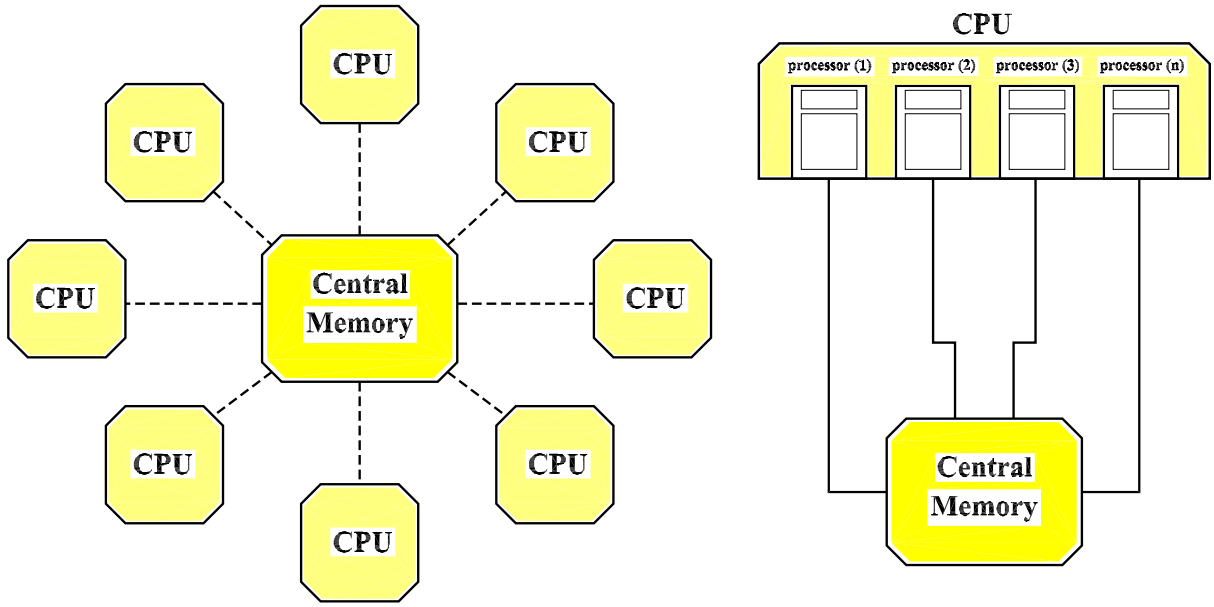


Figure 3.3: Shared memory architecture

3.1.2 Distributed memory architecture

The set of computers interconnected through a network is known as a *Distributed Memory Multicomputer* or message passing multicomputer. This memory type consists of connecting the inter-processor memory of the independent computers via an communication network as shown in Fig. 3.4. In this scheme, processors have their own local memory and so operate independently. The memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.

Inter-processor communication is achieved through sending messages explicitly from each computer to another using message passing libraries. It is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is very essential and it is likewise the programmer's responsibility. The distributed memory multicomputer will physically scale easier than a shared memory multiprocessor, i.e. it can more easily be extended by adding more computers to the network.

The most compelling reason for using message passing multicomputers is in its direct applicability to existing computer networks. Scalability between the memory and the number of processors is an advantage of the message passing multicomputer. Furthermore, each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherency.

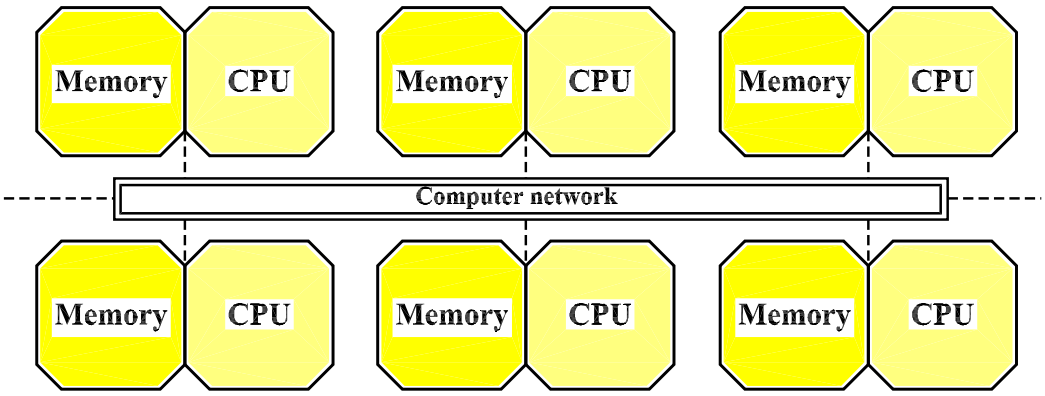


Figure 3.4: Distributed memory multicomputer

3.2 Parallel virtual machine (PVM)

Parallel Virtual Machine (PVM) as well as Message Passing Interface (MPI) are libraries that provide functions to handle communication between processors in a distributed memory environment. PVM is a software package that permits a heterogeneous collection of serial, parallel and vector computers which are connected to a network to appear as one large computing resource. The function calls which are provided by the communication library can be added to a serial program in order to convert it to a parallel program, often with only a few modifications. PVM programs may be compiled to run in parallel on multiple nodes of parallel computers as well as on a cluster of workstations connected over a network.

In a distributed memory machine, a process is created for each processor taking part in the calculation. The creation of processes can be done within the code by using PVM. In other cases, the processes can be created with a separate program that starts the processes on all the processors specified by the user. All the processes have the same code, which has to be modified so that each processor knows exactly what part of the code to execute. This is usually done based on the process identification number. Since PVM as well as MPI have a message-passing interface, i.e. a communication library, it is the responsibility of a software developer to initialize the communication environment, implement the data exchanges, synchronize the execution, and terminate the communication environment.

There are three main parallel models which are usually used in parallel computations [89], these models are

• **Crowd computing model**
This model consists of a collection of closely related processes, typically executing the same code, performing calculations on different portions of the workload and usually involving the periodic exchange of intermediate results. This computing model can be classified into

▶ **Master-slave model** that has a separate control program, i.e. the master, which is responsible for spawning, initialization, collection and display of results. The slave programs perform the actual computations on the workload allocated either by the master or by themselves.

▶ **Node-to-node model** where multiple instances of a single program execute, with one process taking over the noncomputational responsibilities as well as contributing to the calculation itself.

• Tree computing model
The processes of this model are spawned (usually dynamically as the computation grows) in a tree-like manner establishing a tree-like parent-child relationship using different combinatorial searching methods, e.g. *branch-and-bound* algorithms [7], *alpha-beta* search [29], and recursive *divide-and-conquer* algorithms [70].

• Hybrid computing model
A hybrid or combination model possesses an arbitrary spawning structure in a way that at any point in the application execution the process relationship may resemble an arbitrary and changing graph. Hybrid processors usually combine two traditionally separate types of computational devices on a single chip, see e.g. [34].

Starting PVM programming, one or more sequential programs containing embedded calls to the PVM library have to be written. Each program corresponds to a task making up the application. The programs are compiled for each architecture in the host pool and the resulting object files are placed at a location accessible from machines in the host pool. An application is executed when the user starts the master or initiating task from a machine within the host pool. The *master* process subsequently starts other PVM tasks, eventually there are a number of active tasks to compute and communicate to solve the problem. Tasks may interact through explicit message-passing. Once the tasks are finished they and the *master* task disassociate themselves from PVM by exiting from the PVM system.

Since the application is executed on multiple processors, the application needs to be divided into parts, which are then distributed to the processors. The purpose of load balancing is to divide the workload into optimal proportions with respect to the sizes and the capabilities of the processors. There are two main methods for decomposing a problem into smaller tasks to be performed in parallel: *functional* and *domain decompositions*. In functional decomposition the problem is decomposed into different tasks, which can be distributed to multiple processors for simultaneous execution while in domain decomposition the problem's data domain is partitioned and distributed to multiple processors for simultaneous execution.

3.3 Computational structure of the parallel MD code

A spatial decomposition method for short-range parallel direct simulations (will be discussed later) can shorten the computation times. In this method a part of the physical simulation domain is assigned to each processor. This processor computes the forces on particles in its domain, determines forces from neighboring domains, and updates the positions and velocities of all particles within its box at each time step. Each processor tracks particles as they enter and exit its sub-domain. As the simulation progresses, processors exchange particles as they move from one sub-domain to another. In this case, particles are reassigned to new processors as they move through the physical domain.

PVM is based on the message-passing computing model of parallel programming. Messages are passed between tasks over the connecting networks. User's tasks are able to initiate and terminate other tasks, send and receive data, and synchronize with one another using a library of message passing routines. Tasks are dynamic, i.e. can be started or killed during the execution of a program, even the configuration of the virtual machine can be dynamically configured.

In our work, the existing serial code was rearranged and modified to work in parallel using a message passing library, see Fig. 3.5. The parallel code is divided into three main parts, i.e. master, slave and control-output programs.

3.3.1 Master program

This part is responsible for starting the different slaves and control-output programs and for distributing the original data to them. It collects also the output data for each simulation time step and redistributes them to the different slaves according to the fixed predefined physical boundaries assigned to the processors.

In the master program, we first get the task ID of the master by calling the PVM function *pvm_mytid()* which enrolls the process in PVM. The slave program finds the task ID of the master by calling the function *pvm_parent()*. The PVM system assigns each process a unique integer called its task ID. The task ID helps in identifying the process with which is needed to communicate. In order to write a parallel program, tasks must be executed on different processors by creating the slave processes using *pvm_spawn()*. As an example of the message passing between processors, the master in Fig. 3.5 creates the two slaves, i.e. slave(1) and slave(2), and the control-output program. The master informs these programs through message 11 and 25 by the essential information needed for initialization about particles positions, velocities, radii, densities, stiffness and damping parameters and some others.

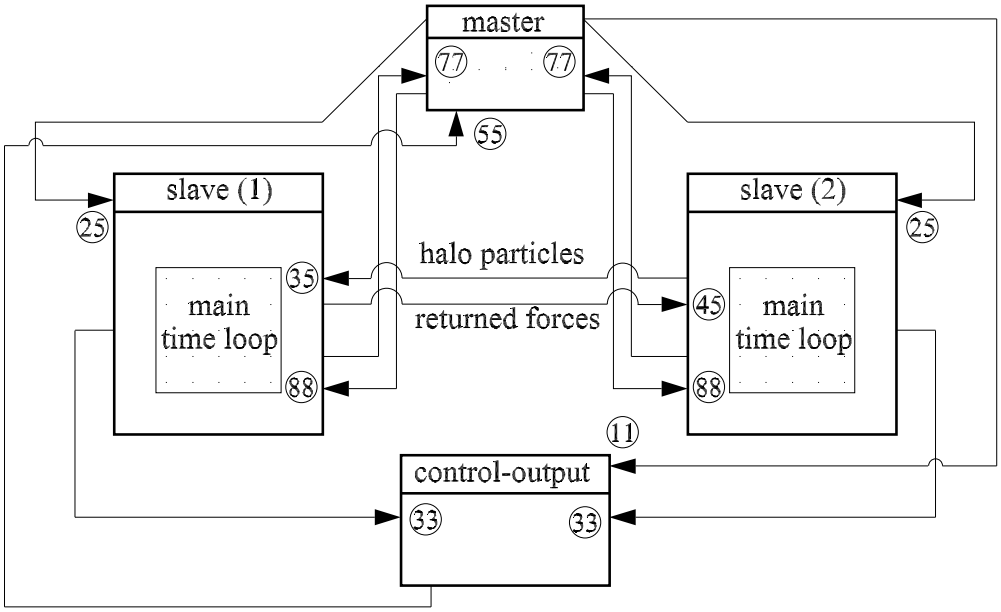


Figure 3.5: Message exchange between master, slaves and control-output codes

3.3.2 Slave programs

The slaves, which are generated according to the selected number of processors, receive the data of the new particle positions from the master program at each simulation time step. The neighborhood-list is generated and the contact forces are then computed. The positions and velocities of all particles are updated and sent directly to the master program in order to redistribute them again at the next time step. Another copy of this updated data is sent also to the control-output program to collect and arrange them in the suitable form of the final output file required for animation. The slaves should also communicate with each other and exchange the data belonging to those particles, which lie in the so-called *halo region* at the domain boundaries.

A message in PVM consists of basically two parts, the data and a tag that identifies the type of the message. To send a message, it is first needed to initialize the send buffer. This is done by calling the *pvm_initsend()* function. Once the buffer has been initialized, we need to put data into the buffer and encode it. So the function *pvm_pkstr()*, *pvm_pkint()* and *pvm_pkdouble()* are used to *pack* these different types of data into the buffer. Once the data is packed, the function *pvm_send()* is called to send the message. The first argument of this function is the ID of the process to which the message is to be sent and the second argument is the message tag. *pvm_mcast()* is a similar function and is useful when the same message should be sent to a set of tasks. It does the same as *pvm_send()*, except it takes an array of tids instead of just one.

Once the data is sent to the slave from the master or from another slave, the slave will process it by calling *pvm_recv()* to receive the coming data. The arguments of this function are the task ID from which the message is expected and the tag of the expected message.

If the desired message has not yet been sent, this function waits and does not return. Thus, in effect, the master is now waiting for the slave to process the data. Once the message arrives, the data is still in the receive buffer. It needs to be *unpacked*, i.e. decoded to get the original message. This decoding is done by the *pvm_upkstr()* function.

In Fig. 3.5 the adjacent slaves exchange information about the location of the halo particles through message 35 and receive back the returned forces by message 45. Each slave integrates its equations of motions and determines the new location of its home particles. These new locations are sent by message 77 to the master in order to be redistributed to the different slaves through message 88. A copy of the new positions is also sent to the control-output program by message 33 in order to be prepared for visualization. The control-output program informs the master through message 55 if there are no further data comes in order to finish and stop the simulation.

Before the PVM program exits, it must tell the PVM system that it is leaving the PVM system so that resources occupied by the process can be released. This is done by calling the *pvm_exit()* function. Many other PVM functions which are used for data communication within the network environment can be found in detail in the PVM reference manual [36].

3.3.3 Control-output program for visualization

This program collects the data, which is sent from the different slaves at each output time step during simulation. These raw results are arranged in the final output file shape for animation. It would be impossible to follow particles motion by reading the output text files and acquiring these information especially for large systems. For a better imagination and to know how the particles move and collide during simulation, visualization of raw data is considered as the important interface for interaction between the user and the computers.

3.4 Parallelized domain decomposition strategies

There are many different strategies normally used to decompose granular systems in parallel computations. These approaches play a significant role in maintaining on certain level of communication/computation balance of parallel environment. Furthermore, each of them solves the problem with different orders of time complexity depending on the size of the system. The spatial decomposition method, which is used in our parallel simulations, is a well-known method in this field. As a general overview on other methods, the two approaches of the *Replicated Data Method* and the *Hierarchical Tree Decomposition Method* are briefly discussed here.

3.4.1 Replicated data method

This method is also called *particle decomposition method* and is one of several ways to achieve parallelization in granular simulations. Furthermore, it is relatively simple to program and is reasonably efficient. Its name derives from the replication of the configuration data on each node of a parallel computer, i.e. the arrays defining the particle coordinates $\mathbf{r}_i$, velocities $\mathbf{v}_i$ and forces $\mathbf{f}_i$, for all N particles $\{i : i = 1, ..., N\}$ in the simulated system, are reproduced on every processing node.

For a system of N particles and P processors, each of the P processors is assigned a group of $N_k = N/P$ particles at the beginning of the simulation, where N_k is the number of particles belongs to processor k and P is the number of processors. Particles in a group do not need to have any special spatial relationship to each other. A processor will compute forces on only its N_k particles and will update their positions, velocities and orientations for the duration of the simulation no matter where they move in the physical domain of the global system.

In this strategy most of the forces computations and the integration of the equations of motion can be shared easily and equally between nodes. As the number of processors P increases, the computation time goes down. But since we still have to communicate the same amount of information which is proportional to the number of particles N, it does not scale. However, this strategy can be expensive in memory and has a high communication overhead, but overall it has proven to be successful over a wide range of applications. These issues are expored in more detail [86, 102].

As a representation of the computational work involved in this algorithm and due to short-range forces, an $N \times N$ sparse force matrix $\mathbf{F}$ can be created [80]. The element of F_{ij} represents the force on particle i due to particle j. The symmetries in the forces $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$ being computed reduce the computational effort. In two-body forces, for example, particle i and its neighbors j forces are updated for both particles in pair when $i < j$ and ignored when $i > j$. Therefore, one computation is sufficient for each pair of particles.

The vectors $\mathbf{r}$ and $\mathbf{f}$ of length N store the position and total force on each particle. For a 3D simulation, $\mathbf{r}_i$ would store the three coordinates of particle i. With these definitions, the replicated data algorithm assigns each processor a sub-block of $\mathbf{F}$ which consists of N_k rows of the matrix. Processor p_k computes matrix elements in the $\mathbf{F}_k$ sub-block of rows, where $0 \leq k \leq P$. It also assigns the corresponding sub-vectors of length N_k denoted by $\mathbf{r}_k$ and $\mathbf{f}_k$. To compute all the elements in $\mathbf{F}_k$, processor p_k needs the positions of many particles owned by other processors. This implies that at every timestep each processor must receive updated particle positions from all the other processors, an operation called *all-to-all* communication. Various algorithms have been developed for performing this operation efficiently on different parallel machines and architectures, see e.g. [33].

The replicated data method shows a relatively low speedup compared with the other methods and is restricted since the distributed memory machines often do not possess enough memory on a processing node to hold all of the data for a large job. When the goal is to simulate an extremely large system and also to take advantage of a larger number of processors, a different approach is needed, e.g. spatial decomposition which will be discussed later.

3.4.2 Hierarchical tree decomposition method

The classical N-body problem simulates the evolution of a system of N particles, where the force exerted on each one arises due to its interaction with all the other bodies in the system. The simulation proceeds over time and the net force acts on each body is computed at every timestep. If all pairwise forces are computed directly, this requires $O(N^2)$ operations at each timestep. Hierarchical tree-based methods have been developed to reduce the complexity. What we need is a supporting data structure to subdivide space, see e.g. [30]. The *octtree* is used in 3-dimensional problems while the *quadtree* is used in 2-dimensional ones. As a brief description of the planar case, the quadtree begins with a square in the plane; this is the root of the quadtree. This large square can be broken into four smaller squares of half the perimeter and a quarter the area each; these are the four children of the root. Each child can in turn be broken into four subsquares to get its children, and so on, see Fig. 3.6.

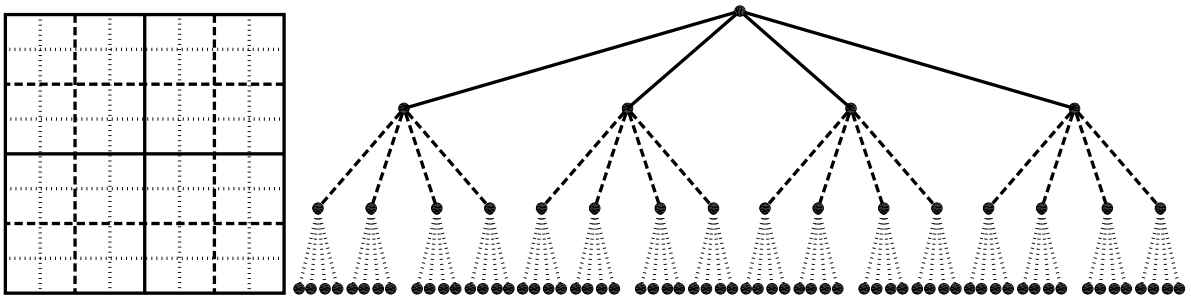


Figure 3.6: A complete quadtree with four levels, each line type represents a new generation of children

There are several approaches that have looked at parallel implementations of these tree-based methods, e.g. the *Barnes-Hut algorithm* (BHA) [9] and *Greengard's Fast Multipole Algorithm* (FMA) [37]. The primary difference between these algorithms is that the BHA algorithm computes particle-cell interactions, whereas the FMA computes cell-cell interactions, thereby reducing its complexity. The Barnes-Hut algorithm solves the problem with time complexity of $O(NlogN)$ while the FMA approximates the solution with bounded error in time $O(N)$.

The Barnes-Hut algorithm solves the problem cleverly by using a divide-and-conquer approach. It is very popular, as it is not too complex to implement, and tends to have fast execution. It uses a quadtree for representing the particles inside the two-dimensional space (or octtree for the corresponding three-dimensional case). Each node in the tree represents a cell enclosing a certain area within the space. A quadtree is constructed by recursively subdividing the root node of the tree, which represents the whole 2D space containing all the particles, into four nodes representing four sub-cells with equal sizes, until each sub-cell has at most one particle. Each cell contains the total mass and the position of the center of mass of all the particles in the subtree below.

After a quadtree (or octtree) is constructed, the tree is traversed from its root once per particle to compute the net force acting on it. The tricky point here for improving performance is that at each step of traversal, if the cell represented by that node is *well separated* from the particle, we can consider that the forces acted on the particle resulting from the particles inside that cell come from a single point of source. In that case, we just use the center of mass approximation to compute the force on the particle due to the entire subtree under that cell. Otherwise, if the cell is close enough, each of its subcells has to be visited. A cell is considered to be well separated from a particle if its size, divided by its distance of its center of mass from the particle, is smaller than a certain parameter called theshold, which controls the accuracy of the approximation, see e.g. [8].

On the other hand, the Fast Multipole Algorithm is a linear-time algorithm. It performs a computation over a hierarchically decomposed space, and has distinct computations for *near* and *far* bodies. Similar to that of the Barnes-Hut algorithm, FMA also uses an octtree for spatial 3-dimensional case where the root encompasses the entire space with each of its children encompassing equal-size octants of the space, repeatedly down to the leaves. After the tree is built, it has a top-down phase in which the local expansion of the parent cell is shifted to the center of each child, and added to the multipole expansions of the cell in the childs's interaction list to form its local expansion. Finally, the local expansions at the leaf cell, along with direct interactions with particles in neighboring cells gives the local force on each particle. The number of terms in the multipole expansions controls the accuracy of the algorithm, see [14, 76].

3.5 Spatial decomposition method (SDM)

In 1995, a classication of the decomposition techniques was proposed by Wilson [117], namely:

► **Geometric decomposition:** The problem domain is broken up into smaller domains and each process executes the algorithm on each part of it.

► **Iterative decomposition:** Some applications are based on loop execution where each iteration can be done in an independent way. This approach is implemented through a

central queue of tasks, and thus corresponds to the task-farming paradigm.
▶ **Recursive decomposition:** This strategy starts by breaking the original problem into several subproblems and solving these in a parallel way. It clearly corresponds to a divide and conquer approach.
▶ **Speculative decomposition:** Some problems can use a speculative decomposition approach, i.e. k solution techniques are tried simultaneously, and $k-1$ of them are thrown away as soon as the first one returns a plausible answer. In some cases this could result optimistically in a shorter overall execution time.
▶ **Functional decomposition:** The application is broken down into many distinct phases, where each phase executes a dierent algorithm within the same problem. The most used topology is the process pipelining, see [16].

The spatial decomposition method which is also called *geometric decomposition method* [28] will be discussed in detail in this section since it is used in our parallel simulations. In this method, to each processor a portion of the physical simulation domain is assigned which is considered as a sub-division of the overall workload of the system. The processor boundaries remain fixed in space as particles moves through them. Each processor computes only the forces on particles in its sub-domain. As the simulation progresses processors exchange particles as they move from one sub-domain to another.

3.5.1 Qualitative overview

The homogeneity in particle distribution along with the expected shape of particles movement plays a significant role in differentiating between these different patterns. The general headlines of the SDM can be described in the following steps:
• **step(1):** The physical volume is divided into a regular grid which are parallel slices in the case of one dimensional decomposition, see Fig. 3.7.
• **step(2):** Each grid cell is assigned to a processor. This processor is responsible for performing the force calculations and state updates for all particles (nominally) within the cell.
• **step(3):** Force computation requires state information for some particles owned by other processors, i.e. the particles located in so-called halo regions at the borders of the processors. These are acquired by a communication phase between slaves at the start of each computational step.
• **step(4):** Particles will occasionally drift across processor boundaries. These processors remain the responsibility of the original parent processor during the basic (communicate, update) cycle outlines in steps 2 and 3. Reassignment of particles to processors according to the cell boundaries is done periodically by the master program.

For neighbor-list computations, each processor computes neighbor list for its local particles using the same algorithm as in the sequential code. The decision to recompute the

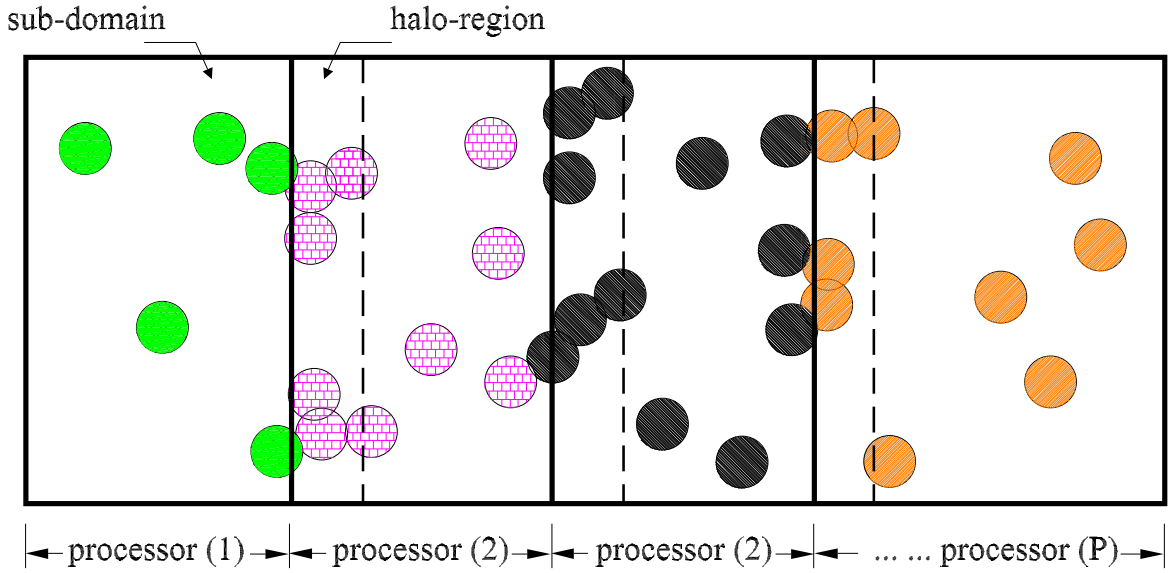


Figure 3.7: Spatial decomposition of the particles domain into parallel vertical slices, each processor is responsible for one of the sub-domains

neighbor list is taken differently in the two implementations. In parallel implementation the neighbor list is recomputed based only on changes in positions of local particles. Also, if a neighboring processor recomputes its neighbor list due to local changes, then particles within a distance of the halo region are sorted, and the neighbor list recomputed. This is needed to ensure that neighboring processors have a consistent view of their particles. When a neighbor list computation is required due to local changes, a processor informs the neighbors of this, along with the message that sends the boundary data, so that the neighbors too can recompute their lists. This scheme assumes that particles stored in each processor span a range of at least the halo region width $2r_{max}$, in order to ensure correctness.

3.5.2 Message communication pattern

Since force calculations and position updating are done locally, at each time step every processor needs to communicate with neighboring processors the updated positions of its halo particles which are located on their boundaries. In this case each processor acquires all information necessary for force computation in each step. PVM provides two modes of communication - *blocking* and *non-blocking*, see e.g. [32].

In the blocking mode of communication the send operation does not complete until the buffer is empty and the receive operation does not return until buffer is full. Whereas the non-blocking communication operations return immediately with request handles that can be waited upon and queried. Thus the use of non-blocking communication provides an opportunity to overlap computation with communication. Using this scheme could

sometimes create some improvements in the communication cost.

The sending and receiving is only related to the particles which lie near a physical boundary of the processors, i.e. the particles which belong to one processor and have the possibility to be in contact with the other particles in the next domain of the other processor. Let us define the particles which belong to processor p_k and lie inside its physical domain as the *home particles* of p_k. On the other hand, let the particles which belong to processor p_k and specifically lie inside the area of its halo region h_k be defined as the *immigrant particles* of p_k. These particles will travel or immigrate to the neighboring processor p_{k-1}, where $k \neq 1$. This immigrant particles of processor p_k will be received by processor p_{k+1} and considered as *visiting particles* to this host processor, where $k \neq P$. The sending-receiving loop of processor p_k is depicted in Fig. 3.8.

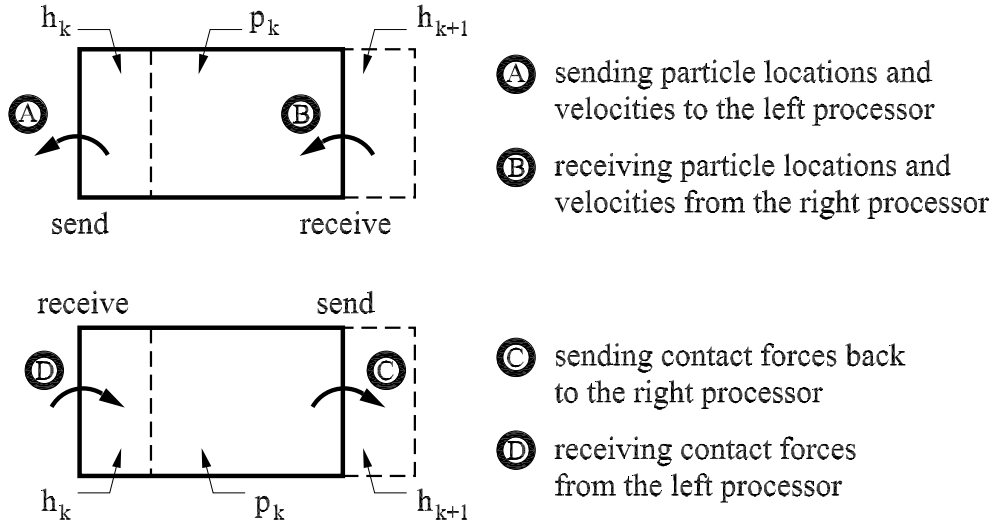


Figure 3.8: Message communication pattern of processor p_k. Each processor sends and receives attributes of buffer data to and from left and right neighboring processors

In the communication pattern described above, several attributes, i.e. positions, velocities, particle radii, particle densities, ...etc., of buffer particles need to be sent to the neighboring processors. Sending individual messages for each array is not desirable because it will cause significant message startup cost irrespective of the amount of data to be sent. Instead, several arrays can be packed together in each communication step, so that we can send just one long message instead of several smaller ones.

3.5.3 Mathematical formulation

By subdividing the physical volume among processors into P domains, where P is the number of processors, the one dimensional decomposition method of parallel slices can be geometrically described as shown in Fig. 3.9. The kth processor p_k is responsible for

particles whose x-coordinates lie in the range of its local domain

$$\sum_{i=0}^{k-1} L_i \leq x < L_k + \sum_{i=0}^{k-1} L_i \ , \tag{3.1}$$

where $0 \leq k \leq P$ and $L_0 = 0$. In this case the values p_k are not equally spaced along the overall width L and each has a local width L_k. Adjusting this width during simulation leads to dynamic processor boundaries which is useful for the load balance of the problem. Dynamical load balance nearly maintains the same number of particles assigned to each processor and, therefore, an almost equal computation load distribution among the slaves.

On the other hand, when the particles distribution is nearly homogeneous along the x-direction and hence the size of movements of the particles between any two adjacent processors is almost identical, the processors are assumed to be equally spaced of fixed-defined boundaries and identical widths L_k, i.e.

$$L_i = \frac{L}{P} \ , \tag{3.2}$$

and therefore Equation (3.1) becomes

$$(k-1)\frac{L}{P} \leq x < k\frac{L}{P} \ . \tag{3.3}$$

Similarly, each processor p_k has a so-called *halo-region* h_k, $k \neq 1$, at its boundaries, where h_2, h_3, ..., h_P are the halo regions of the processors p_2, p_3, ..., p_P, respectively, see Fig. 3.9. The minimum width of the halo-region is usually selected to be as twice as the maximum radius of the particles in the system $2r_{max}$. This selection is to maintain on all contact possibilities of those particles on the borders and belong to different processors. The width of the halo-regions b_k can be assumed to be identical, i.e.

$$b_i = 2r_{max}, \quad 2 \leq i \leq P \ . \tag{3.4}$$

Therefore, the particle is assumed to belong to the halo-region h_k of processor p_k if the x-coordinate of its center lies in the range of the halo-region extremes, i.e.

$$(k-1)\frac{L}{P} \leq x < (k-1)\frac{L}{P} + 2r_{max} \ . \tag{3.5}$$

Its always true that $M_k \leq N_k$, where N_k is the local number of particles of processor p_k and M_k is the number of particles belongs to the halo-region h_k; $k \neq 1$. Assuming an equally particle distributed scheme, $N_k = N/P$ for $1 \leq k \leq P$.

Message communication and exchanging data between different processors of the slaves and the master can be summarized in the following steps:

• **step(1):** For each processor p_k, where $1 \leq k \leq P$, construct the matrix $\mathbf{H}$ of the individual *home* particles, i.e. the particles with $(k-1)L/P \leq r_x^k < k\ L/P$

$$\mathbf{H}^k = \Big[\ \mathbf{g}_h^{k^T} \quad \mathbf{r}_x^{k^T} \quad \mathbf{r}_y^{k^T} \quad \mathbf{r}_z^{k^T} \quad \mathbf{v}_x^{k^T} \quad \mathbf{v}_y^{k^T} \quad \mathbf{v}_z^{k^T} \quad \mathbf{r}_{x,old}^{k^T} \quad \mathbf{r}_{y,old}^{k^T} \quad \mathbf{r}_{z,old}^{k^T} \quad \mathbf{r}^{k^T} \quad \boldsymbol{\rho}^{k^T} \ \Big]_{N_k \times 12}^{T} , \tag{3.6}$$

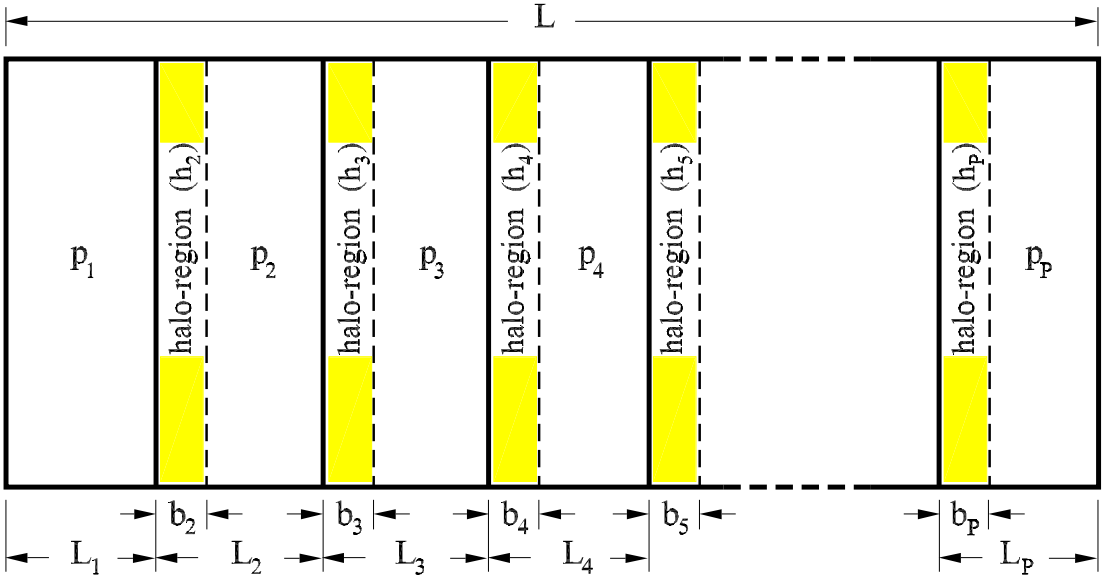


Figure 3.9: Geometrical description of the processors dimensions in a slice spatial decomposition method

where $\mathbf{g}_h$ is the vector of the global indices of home particles, $\mathbf{r}_x$, $\mathbf{r}_y$ and $\mathbf{r}_z$ are the vector of particles positions along the three coordinate axes x, y and z, respectively, $\mathbf{r}_{x,old}$, $\mathbf{r}_{y,old}$ and $\mathbf{r}_{z,old}$ are the vectors of old-particle positions along those axes, $\mathbf{v}_x$, $\mathbf{v}_y$ and $\mathbf{v}_z$ are the vector of particles velocities along the main coordinate axes, $\mathbf{r}$ is the vector of particles radii and $\boldsymbol{\rho}$ is the vector of particles mass density.

• **step(2):** Construct the matrix $\mathbf{I}$ of the *immigrant* particles for each processor k, where $k \neq 1$, as

$$\mathbf{I}^k = \left[\begin{array}{cccccccccccc} \mathbf{g}_i^{k^T} & \mathbf{r}_x^{k^T} & \mathbf{r}_y^{k^T} & \mathbf{r}_z^{k^T} & \mathbf{v}_x^{k^T} & \mathbf{v}_y^{k^T} & \mathbf{v}_z^{k^T} & \mathbf{r}_{x,old}^{k^T} & \mathbf{r}_{y,old}^{k^T} & \mathbf{r}_{z,old}^{k^T} & \mathbf{r}^{k^T} & \boldsymbol{\rho}^{k^T} \end{array} \right]^T_{M_k \times 12}, \quad (3.7)$$

where $\mathbf{g}_i$ is the vector of the global indices of the immigrant particles. These particles are located inside the halo region h_k of processor k, i.e. the particles with $(k-1)L/P \leq r_x^k < (k-1)L/P + 2r_{max}$, where r_{max} is the maximum particle radius in the system. These immigrant particles which belong to processor p_k and are stored in the matrix $\mathbf{I}^k$ should be sent in a buffer to the left neighboring processor p_{k-1} using PVM library commands. This particle exchange will lead to an extra communication cost relative to the sequential programming.

• **step(3):** This buffer which holds data of the matrix $\mathbf{I}^k$ of the immigrant particles from processor p_k will be received by processor p_{k-1} and stored in a matrix allocated for receiving the *visiting* particles $\mathbf{V}^{k-1}$, i.e.

$$\mathbf{V}^k = \mathbf{I}^{k+1} \quad \text{and} \quad \mathbf{V}^{k-1} = \mathbf{I}^k \ , \quad (3.8)$$

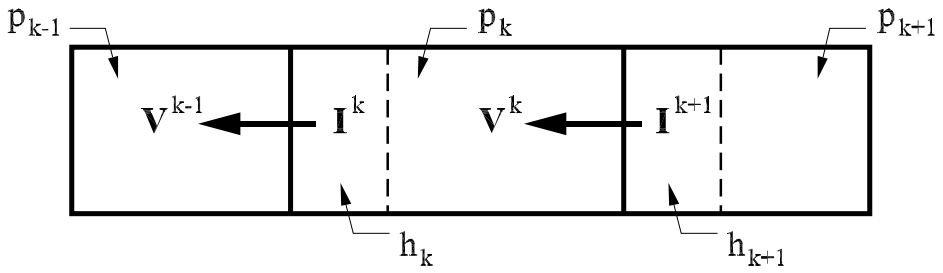


Figure 3.10: Sending halo-particles (locations, velocities,...etc) to the neighboring processors; sending the particles data in **I** and receiving it in **V**

for $k \neq 1$, see Fig. 3.10.

• **step(4):** Now processor p_k is ready to simulate its home particles of $\mathbf{H}^k$ together with the visiting particles of $\mathbf{V}^k$. These two matrices can be merged together in a new matrix $\mathbf{E}^k$ as

$$\mathbf{E}^k = \begin{bmatrix} \mathbf{H}^k \\ \mathbf{V}^k \end{bmatrix} = \begin{bmatrix} \mathbf{H}^k \\ \mathbf{I}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{H}^{k^T} & \mathbf{I}^{k+1^T} \end{bmatrix}^T . \tag{3.9}$$

The size of this matrix is extended to be the sum of the sizes of both entire matrices $\mathbf{H}^k$ and $\mathbf{I}^{k+1}$, i.e. $(N_k + M_{k+1}) \times 12$.

• **step(5):** Each processor p_k computes the overall contact forces acting on each particle of its merged home-visiting matrix $\mathbf{E}^k$. These particles are of two kinds, i.e. home particles belong to the processor p_k itself and visiting particles coming from the neighboring processor p_{k+1}. It should be noticed that these visiting particles are also considered as home particles with respect to the processor p_{k+1}. In this case and to avoid replication of force calculations between the *visiting-visiting* (v-v) particles, these calculations are done once in their home processor p_{k+1} and not in the host processor p_k. This strategy will ensure reduction in the additional computation cost. On the other hand, the host processor p_k does the force calculations only between its *home-home* (h-h) particles and *home-visiting* (h-v) particles.

The contact forces acting on the *home* particles of processor p_k due to h-h and h-v interactions are stored the matrix $\mathbf{P}_h^k$ as

$$\mathbf{P}_h^k = \begin{bmatrix} \mathbf{g}_h^{k^T} & \mathbf{f}_x^{k^T} & \mathbf{f}_y^{k^T} & \mathbf{f}_z^{k^T} \end{bmatrix}_{N_k \times 4}^T , \tag{3.10}$$

where $\mathbf{f}_x$, $\mathbf{f}_y$ and $\mathbf{f}_z$ are the vector of the contact forces of home particles along the three coordinate axes x, y and z, respectively.

• **step(6):** On the other hand, the contact forces acting on the *visiting* particles in processor p_k, $k \neq P$, due to h-v interactions are stored in the matrix $\mathbf{P}_v^k$ as

$$\mathbf{P}_v^k = \begin{bmatrix} \mathbf{g}_i^{k+1^T} & \mathbf{s}_x^{k^T} & \mathbf{s}_y^{k^T} & \mathbf{s}_z^{k^T} \end{bmatrix}_{M_{k+1} \times 4}^T , \tag{3.11}$$

where $\mathbf{s}_x$, $\mathbf{s}_y$ and $\mathbf{s}_z$ are the vector of the contact forces of visiting particles along the main coordinate axes x, y and z, respectively.

• **step(7):** This force matrix $\mathbf{P}_v^k$ which stores the contact forces between the home particles of processor p_k and the visiting particles of processor p_{k+1} should be returned back in a reverse sending process to its original home processor p_{k+1}. Processor p_k will receive this force matrix and store it in $\mathbf{G}_v^{k+1}$, i.e.

$$\mathbf{G}_v^{k+1} = \mathbf{P}_v^k = \left[\begin{array}{cccc} \mathbf{g}_i^{k+1^T} & \mathbf{s}_x^{k^T} & \mathbf{s}_y^{k^T} & \mathbf{s}_z^{k^T} \end{array} \right]^T_{M_{k+1}\times 4} , \tag{3.12}$$

for $k \neq P$, and similarly

$$\mathbf{G}_v^{k} = \mathbf{P}_v^{k-1} = \left[\begin{array}{cccc} \mathbf{g}_i^{k^T} & \mathbf{s}_x^{k^T} & \mathbf{s}_y^{k^T} & \mathbf{s}_z^{k^T} \end{array} \right]^T_{M_{k}\times 4} . \tag{3.13}$$

It should be recognized that step(7) is quite opposite to step(3). In step(3), the buffer is sent in a forward direction to processor p_k holding a data about some dynamical and geometrical information of the immigrant particles, while in this step the direction of the sending data is reversed. The buffer holds back the information about the contact forces of these particles but in the backward direction, compare Fig. 3.10 and Fig. 3.11.

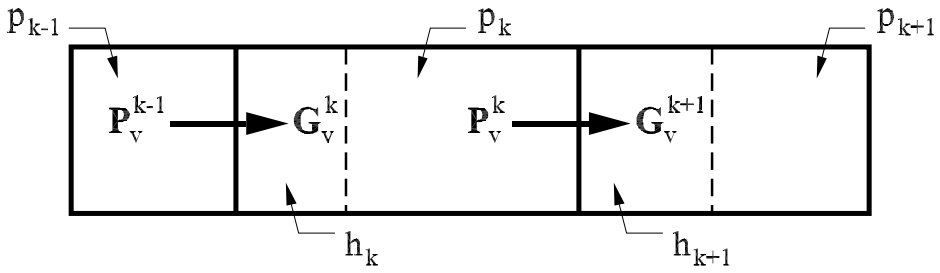


Figure 3.11: Receiving back the halo-particle forces from the neighboring processors; sending the force data in $\mathbf{P}_v$ and receiving it in $\mathbf{G}_v$

• **step(8):** Each processor p_k has now in hand the contact forces acting on its own particles inside its physical domain. These forces are summed up in the overall contact force matrix $\mathbf{F}_h^k$ taking into consideration that the forces which are summed together are only for the particles of identical global indices, i.e.

$$\mathbf{F}_h^k = \begin{cases} \mathbf{P}_h^k + \mathbf{G}_v^k & \text{if} \quad \mathbf{g}_h^k(j) = \mathbf{g}_i^k(l), \quad \forall\ 1 \leq j \leq N_k, \\ & \qquad\qquad\qquad\quad \forall\ 1 \leq l \leq M_k, \\ \mathbf{P}_h^k & \text{else}, \end{cases} \tag{3.14}$$

where j and l are the local particle indices of the home and immigrant particles, respectively. The addition of $\mathbf{P}$ and $\mathbf{G}$ in Equation (3.14) is only for the identical particles which have the same global indices in the both matrices. Matrix $\mathbf{F}_h^k$ has a size of $N_k \times 4$.

• **step(9):** Since the contact forces applied on the particles are computed, the corresponding accelerations are directly found by dividing each of the forces of Equation (3.14)

by the mass of the particle. Each processor p_k will solve the equations of motion of its home particles by integrating them explicitly using the Verlet approach which was previously discussed in Section 2.1.1. This integration will update the data of the old particles situtation and give their new positions, velocities and orientations.

• **step(10)**: Save the output data coming from all slaves and arrange them in the proper requested form of the animation output files in which they have to be sorted globally according to their identities. This sorting process along with some file arrangements are done at each time step through a control-output program to be ready for visualization.

• **step(11)**: As the master receives the data about the new particle locations from all slaves, it will directly generate the new updated matrices as in Equations (3.6) and (3.7) in order to proceed further in a new simulation time step. Starting again in a loop from step(1), it will be recognized that all vectors and matrices belong to the slaves have to be *dynamically* allocated due to the variety in the number of particles simulated by slaves at each time step.

Since in the following example we deal with particles which are almost homogeneously distributed along the width of the domain, the parallel slice decomposition is quite acceptable to divide the physical domain of the system. Especially it does not need a lot of dynamic load balancing between processors. Increasing number of processors in the parallel environment will increase communication but decrease computation between particles where they have to be optimized.

In order to show how the pre-mentioned steps are applied to construct the different necessary vectors and matrices and send them in a communication scheme between processors, a small quantitative example of ten particles is created in Fig. 3.12. The data matrices of the home particles of the three processors p_1, p_2 and p_3 can be built using Equation (3.6) as

$$\mathbf{H}^1 = \begin{bmatrix} 0 & x_0 & z_0 & .. & .. \\ 3 & x_1 & z_1 & .. & .. \\ 5 & x_2 & z_2 & .. & .. \end{bmatrix}^1_{3\times 12}, \quad \mathbf{H}^2 = \begin{bmatrix} 2 & x_0 & z_0 & .. & .. \\ 6 & x_1 & z_1 & .. & .. \\ 7 & x_2 & z_2 & .. & .. \end{bmatrix}^2_{3\times 12}, \text{ and}$$

$$\mathbf{H}^3 = \begin{bmatrix} 1 & x_0 & z_0 & .. & .. \\ 4 & x_1 & z_1 & .. & .. \\ 8 & x_2 & z_2 & .. & .. \\ 9 & x_3 & z_3 & .. & .. \end{bmatrix}^3_{4\times 12}. \tag{3.15}$$

The immigrant particles in the halo-regions of processors p_1 and p_2 are stored in the matrix $\mathbf{I}$ of Equation (3.7) as

$$\mathbf{I}^2 = \begin{bmatrix} 7 & x_2 & z_2 & .. & .. \end{bmatrix}^2_{1\times 12}, \quad \mathbf{I}^3 = \begin{bmatrix} 4 & x_1 & z_1 & .. & .. \\ 8 & x_2 & z_2 & .. & .. \end{bmatrix}^3_{2\times 12}. \tag{3.16}$$

Accordingly, the visiting particles of the host processors p_2 and p_3 are stored in the matrix $\mathbf{V}$ of Equation (3.8) as

$$\mathbf{V}^1 = \mathbf{I}^2 = \begin{bmatrix} 7 & x_2 & z_2 & .. & .. \end{bmatrix}^2_{1\times 12}, \quad \mathbf{V}^2 = \mathbf{I}^3 = \begin{bmatrix} 4 & x_1 & z_1 & .. & .. \\ 8 & x_2 & z_2 & .. & .. \end{bmatrix}^3_{2\times 12}. \tag{3.17}$$

The *home* particles in Equation (3.15) and the *visiting* particles in Equation (3.17) are merged in one matrix $\mathbf{E}$ as showed in Equation (3.9)

$$\mathbf{E}^1 = \begin{bmatrix} 0 & x_0 & z_0 & .. & .. \\ 3 & x_1 & z_1 & .. & .. \\ 5 & x_2 & z_2 & .. & .. \\ 7 & x_3 & z_3 & .. & .. \end{bmatrix}^1_{4\times 12}, \quad \mathbf{E}^2 = \begin{bmatrix} 2 & x_0 & z_0 & .. & .. \\ 6 & x_1 & z_1 & .. & .. \\ 7 & x_2 & z_2 & .. & .. \\ 4 & x_3 & z_3 & .. & .. \\ 8 & x_4 & z_4 & .. & .. \end{bmatrix}^2_{5\times 12}, \text{ and}$$

$$\mathbf{E}^3 = \begin{bmatrix} 1 & x_0 & z_0 & .. & .. \\ 4 & x_1 & z_1 & .. & .. \\ 8 & x_2 & z_2 & .. & .. \\ 9 & x_3 & z_3 & .. & .. \end{bmatrix}^3_{4\times 12}, \tag{3.18}$$

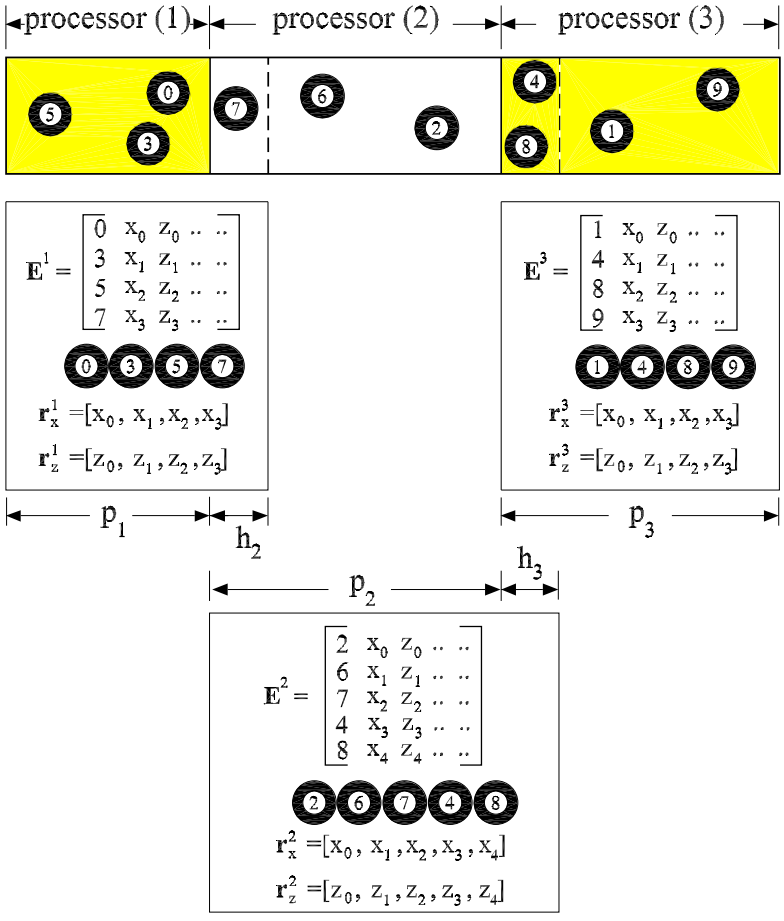


Figure 3.12: Particle immigration between processors using spatial decomposition method

where $x_3^1 = x_2^2$, $z_3^1 = z_2^2$, $x_3^2 = x_1^3$, $z_3^2 = z_1^3$, $x_4^2 = x_2^3$ and $z_4^2 = z_2^3$.

According to Equation (3.10), the contact forces acting on the home particles of the three processors due to h-h and h-v interactions are

$$\mathbf{P}_h^1 = \begin{bmatrix} 0 & f_{x0} & f_{z0} & .. & .. \\ 3 & f_{x1} & f_{z1} & .. & .. \\ 5 & f_{x2} & f_{z2} & .. & .. \end{bmatrix}^1_{3\times 12}, \ \mathbf{P}_h^2 = \begin{bmatrix} 2 & f_{x0} & f_{z0} & .. & .. \\ 6 & f_{x1} & f_{z1} & .. & .. \\ 7 & f_{x2} & f_{z2} & .. & .. \end{bmatrix}^2_{3\times 12}, \text{ and}$$

$$\mathbf{P}_h^3 = \begin{bmatrix} 1 & f_{x0} & f_{z0} & .. & .. \\ 4 & f_{x1} & f_{z1} & .. & .. \\ 8 & f_{x2} & f_{z2} & .. & .. \\ 9 & f_{x3} & f_{z3} & .. & .. \end{bmatrix}^3_{4\times 12}. \tag{3.19}$$

Using Equations (3.12) and (3.13), the returning force matrix $\mathbf{G}_v$ of the visiting particles which will be received by processors p_2 and p_3 is

$$\mathbf{G}_v^2 = \mathbf{P}_v^1 = \begin{bmatrix} 7 & s_{x0} & s_{z0} & .. & .. \end{bmatrix}^2_{1\times 12}, \ \mathbf{G}_v^3 = \mathbf{P}_v^2 = \begin{bmatrix} 4 & s_{x0} & s_{z0} & .. & .. \\ 8 & s_{x1} & s_{z1} & .. & .. \end{bmatrix}^3_{2\times 12}. \tag{3.20}$$

The resulting overall contact forces that act on the home particles of each processor are computed in matrix $\mathbf{F}_h$ according to Equation (3.14) as

$$\mathbf{F}_h^1 = \begin{bmatrix} 0 & f_{x0} & f_{z0} & .. & .. \\ 3 & f_{x1} & f_{z1} & .. & .. \\ 5 & f_{x2} & f_{z2} & .. & .. \end{bmatrix}^1_{4\times 12}, \ \mathbf{F}_h^2 = \begin{bmatrix} 2 & f_{x0} & f_{z0} & .. & .. \\ 6 & f_{x1} & f_{z1} & .. & .. \\ 7 & (f_{x2}+s_{x0}) & (f_{z2}+s_{z0}) & .. & .. \end{bmatrix}^2_{3\times 12}, \text{ and}$$

$$\mathbf{F}_h^3 = \begin{bmatrix} 1 & f_{x0} & f_{z0} & .. & .. \\ 4 & (f_{x1}+s_{x0})) & (f_{z1}+s_{z0}) & .. & .. \\ 8 & (f_{x2}+s_{x1})) & (f_{z2}+s_{z1})) & .. & .. \\ 9 & f_{x3} & f_{z3} & .. & .. \end{bmatrix}^3_{4\times 12}. \tag{3.21}$$

This force of Equation (3.21) will be explicitly integrated to extract the new positions and velocities of the particles.

3.6 Simulation results

In general, parallel applications are much more complex than corresponding serial applications. Not only there are multiple instruction streams executing at the same time, but also the data flowing between them. The primary intent of parallel programming is to decrease execution wall-clock time.

Due to the need of data replication and to the overheads associated with parallel support libraries and subsystems, the amount of memory required can be greater for parallel codes

than for serial ones. However, adding more computers to the parallel environment will generally increase the performance of the parallel coding in a so-called *speedup* of the parallel program. Furthermore, the hardware factors play a significant role in scalability, e.g. memory-CPU bus bandwidth, communications network bandwidth, amount of memory available on any given machine and the processor clock speed.

In our work, a Linux-Cluster of about 25 Pentium 4 computers, speed of $2.8 \div 3$ GHz, cache-memory of $0.5 \div 2$ MByte and RAM of $1 \div 2$ GByte is used. Different problem sizes of various number of particles are simulated on different processors. These processors could belong to a single computing machine of a single CPU with a main memory as shown in Fig. 3.3, or belong to different machines each of which has a single or multiple processors. In our computations, parallel simulations are performed either on multi-processes of a single machine or on processors of multiple machines in a distributed memory system, see Section 3.1.2.

Performance is of paramount importance in parallel programming. Measuring the performance of a parallel program is a way to assess how well and how efficient our efforts have been at dividing the application into modules cooperating with each other in parallel. The most visible metric of performance is the *execution time*. By measuring how long the parallel program needs to run to solve our problem, we can directly measure its effectiveness. To find out how much better our program does on the parallel machine, compared to a program running on only one processor, taking the ratio of the two is a natural solution. This measure is called the *speedup*, and is associated with the number of processors in the machine. So, the speedup is a measure of how much faster the program runs on the parallel machine than it does on a serial machine. It is important, of course, that the program with the serial time be measured on the same hardware we are using to run the parallel implementation. Therefore, the speedup is simply defined as the serial execution time of a sequential code over the parallel processing time, as

$$\text{speedup } \tau = \frac{\text{serial execution time}}{\text{parallel execution time}} = \frac{t_s}{t_p} \ , \tag{3.22}$$

where t_s is execution time of the sequential program and not that of the parallel program of *one* processor t_1. The time t_1 is not comparable with t_s since the parallel program running with $P = 1$ may have much overhead and therefore $t_1 > t_s$.

Table 3.2 shows the execution times and the speedup τ of the parallel code of two numbers of particles over different number of processors. The speedup curves, which describe the relation between the speedup and the number of processors, are plotted in Fig. 3.13a. From these curves, it can be recognized that the job can be simulated faster as it is distributed to a large number of processors regardless if they are on a single or multiple machines. The algorithm may have inherent limits to scalability, i.e. at some point, adding more resources causes performance to decrease. Most parallel solutions demonstrate this characteristic at some point. From the speedup curves, it can be also recognized that

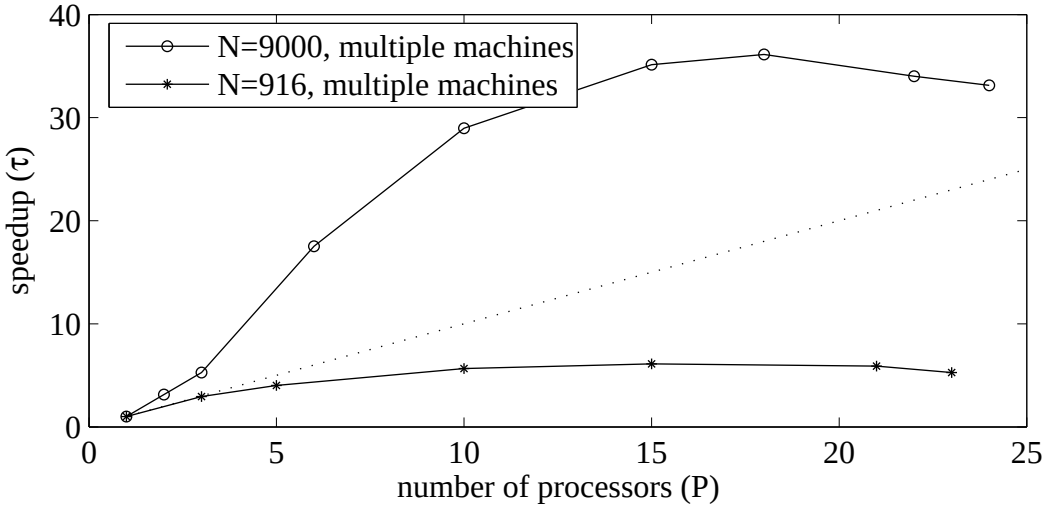
the speedup increases as the number of processors increases until the maximum optimum value is reached. This optimum value is a value after which the increment of the number of processing machines will contribute nothing in improving the speed of the parallel simulation. This happens due to that communication time between processors is highly increased with respect to the computation time. In this computations the value reached is about $P = 20$ processors.

Table 3.2: The computation wall-clock time and the corresponding speedup of two different system sizes of 916 and 9000 particles simulated on processors of different computing machines for a simulation time of 0.1 s and simulation time step $\Delta t = 10^{-5}$ s.

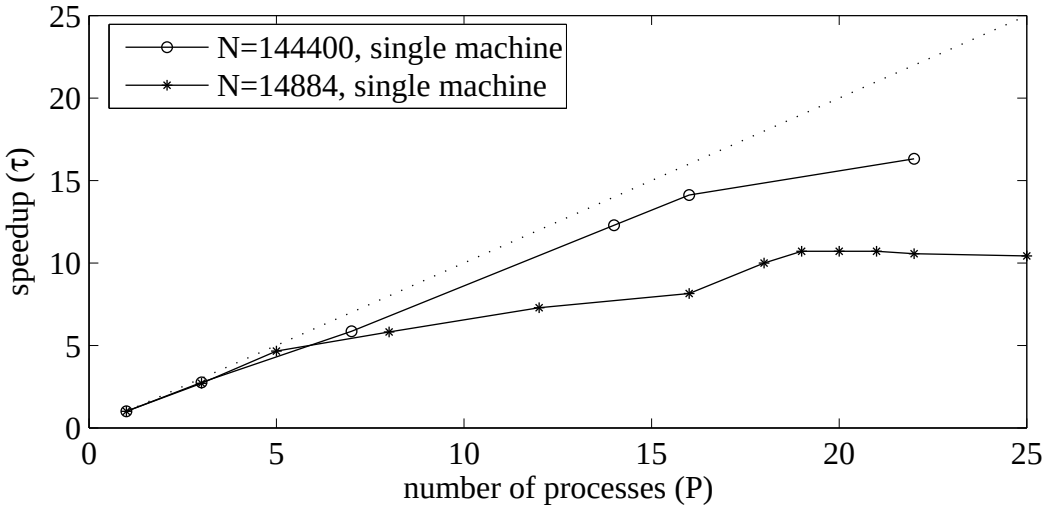
number of processors	N=916 particles		N=9000 particles	
	time (h)	speedup τ	time (h)	speedup τ
1	15.3	1	6.43	1
2	–	–	2.05	3.1
3	5.2	2.9	1.22	5.3
5	3.8	4.0	–	–
6	–	–	0.37	17.5
10	2.7	5.7	0.22	29.0
15	2.5	6.1	0.18	35.1
18	–	–	0.18	36.1
21	2.6	5.9	–	–
22	–	–	0.19	34.0
23	2.9	5.3	–	–
24	–	–	0.20	33.1

On the other hand, the computations show a usual *slowdown* and *superlinear* behavior in the speedup of parallel algorithm. The superlinear improvement means that a program on k processors is more than k times faster than the equivalent uniprocessor. An important factor for the superlinear speedup is that when the whole problem is divided into several small problems, each processor get a smaller amount of data so that the cache hit ratio could become higher with the smaller data size, which will lead to higher computation speed. After the number of processors reaches a certain level that the entire dataset of a subdomain problem can be contained in the cache, further increase of machine size will not improve the computation speed but only increases the percentage of communication time, which will lower the efficiency, see e.g. [39, 77].

The computation measurements which is represented in Table 3.3 are performed on a system of $N = 14884$ particles. The parallel simulations show that the simulations can be performed in a time less than that of the sequential simulation. Moreover, it can be recognized that the job can be simulated faster as it is distributed to a larger number of



(a) speedups measured on multiple machines



(b) speedups measured on a single machine

Figure 3.13: The speedup of parallel simulation of different problem sizes of 916, 9000, 14884 and 144400 particles measured on different processors of single and multiple computing machines. The superlinear speedup behavior appears over the 45°-ideal line

processors regardless if they are on a single or multiple machines. Performance of parallel computations can be measured by several ways, they mainly are

• **execution-time curve**: This curve presents the execution time required for parallel compution for different number of processors, see Fig. 3.14a. This figure shows the usual and the superlinear behavior above and below the ideal computation line, respectively. This kind of figures give some insight, but it is not the best representation of the results.

• **speedup curve**: This curve is a direct representation of Equation (3.22) which gives much more insight and makes the increment of the speed much easier to be readable, see Fig. 3.14b. The speedup is normally bounded in a slowdown speedup within the interval $0 \leq \tau \leq P$. For a superlinear behavior $\tau > P$. In this figure, the superlinear behavior

Table 3.3: Computation wall-clock time and corresponding speedup of 14884 particles simulated on processors located on the single and different computing machines for a simulation time of 0.3 s and simulation time step $\Delta t = 10^{-5}$ s.

number of processors	single computing machine		multiple computing machines	
	time (h)	speedup τ	time (h)	speedup τ
1	75	1	75	1
3	27.8	2.7	10.5	7.1
5	16.1	4.7	3.8	19.7
6	–	–	3.0	25.0
8	12.9	5.8	2.5	30.0
10	–	–	1.9	39.5
12	10.3	7.3	1.6	46.9
16	9.2	8.2	1.5	50.0
18	7.5	10.0	1.3	57.3
19	7.0	10.7	1.4	54.4
20	7.0	10.7	1.5	51.7
21	7.0	10.7	–	–
22	7.1	10.6	1.5	50.7
25	7.2	10.4	–	–

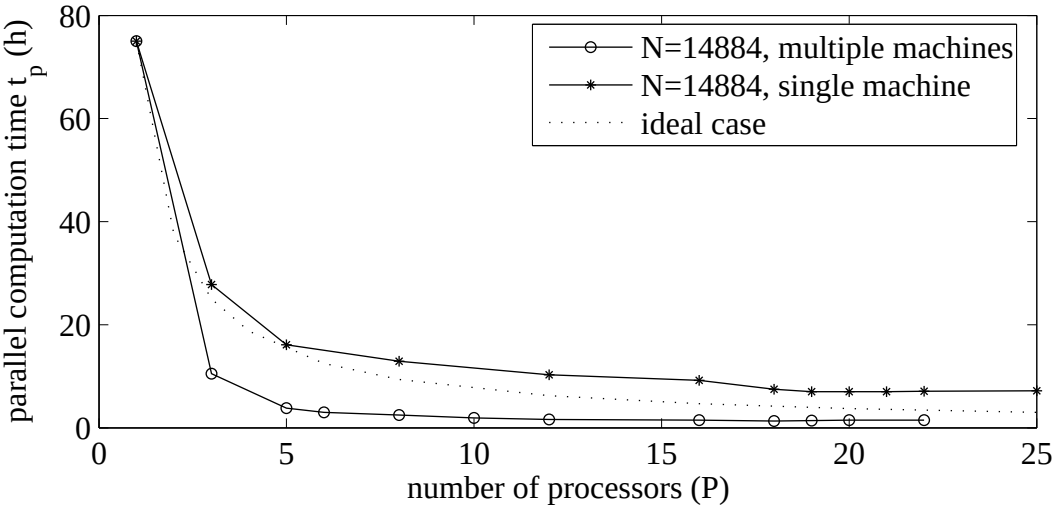
in the speedup τ appears again when the simulation is done over distributed machines of different processors and individual cache memories which will contribute in decreasing the communication cost effect and improving the performance.

• **efficiency curve**: The efficiency η of a parallel program is defined as the fraction of the total computing power that is usefully employed. Using Equation (3.22), the efficiency of the parallel program η is defined as

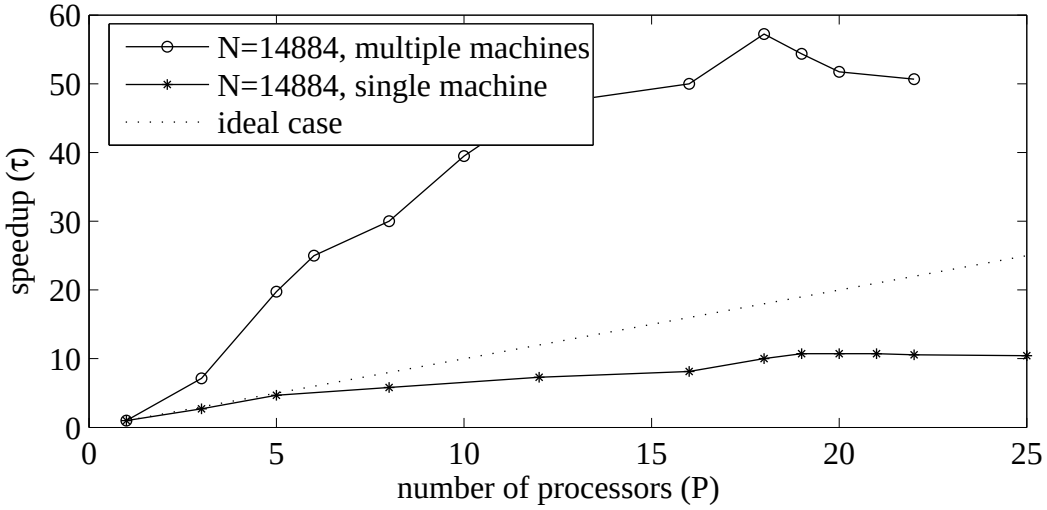
$$\eta = \frac{\tau}{P} = \frac{t_s}{P t_p} \,, \tag{3.23}$$

where P is the total number of processors. This efficiency is normally bounded in a normal speedup within the interval $0 \leq \eta \leq 1$, while $\eta > 1$ is for the superlinear behavior. The parallel efficiency η first increases to above 100 % and then decrease after a certain number of processors, see Fig. 3.14c. This is supported by the testing case of $P > 15$ processors where each processor gets only a very small amount of data within its subdomain, and communication cost becomes substantial compared to computation time.

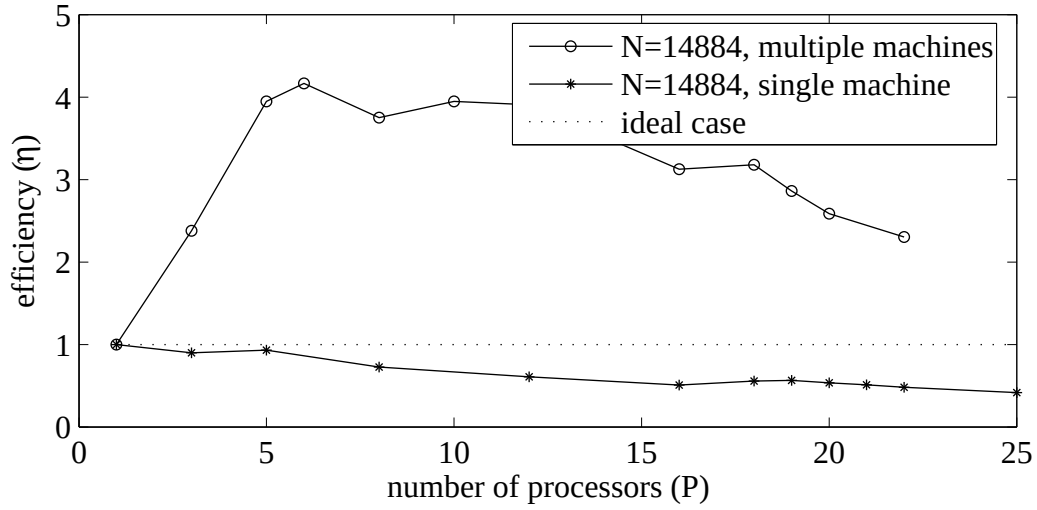
• **normalized cost**: It is also called the *inefficiency* and is defined as the ratio between the time of the parallel program and the time of a perfectly parallelized version of the sequential program. The normalized cost ν can be written as the reciprocal of the



(a) computation-time curves



(b) speedup curves



(c) efficiency curves

Figure 3.14: Parallel computation-time, speedup and efficiency curves of parallel simulation of N=14884 particles measured on different processors of single and multiple computing machines

efficiency

$$\nu = \frac{Pt_p}{t_s} = \frac{P}{\tau} = \frac{1}{\eta} \, . \tag{3.24}$$

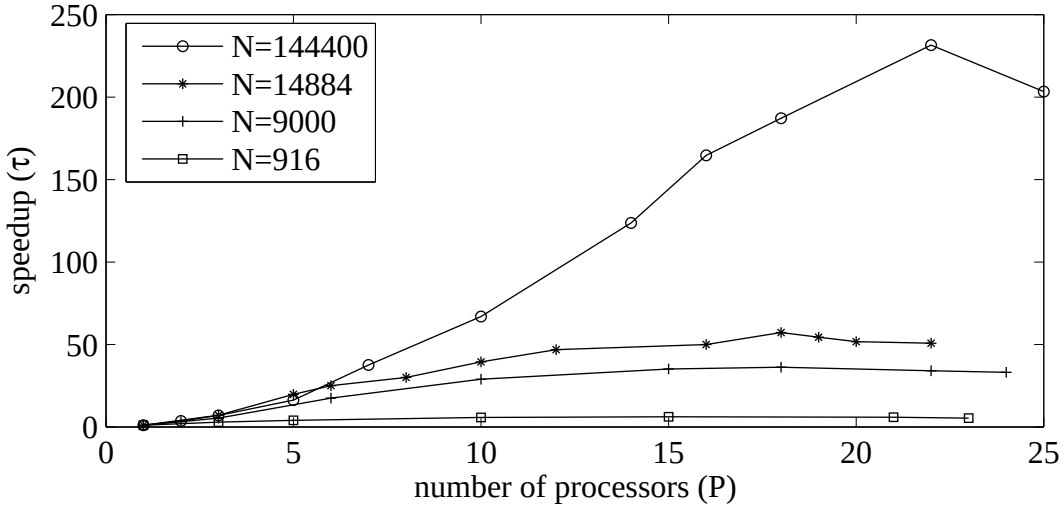
• **overhead**: Normally it is defined as $\nu - 1$. The overhead in parallel simulations usually consists of the load imbalance, synchronization and communication times.

Table 3.4: Computation wall-clock time and corresponding speedup of 144400 particles simulated on processors located on the single and different computing machines for a simulation time of 0.16 s and simulation time step $\Delta t = 10^{-5}$ s.

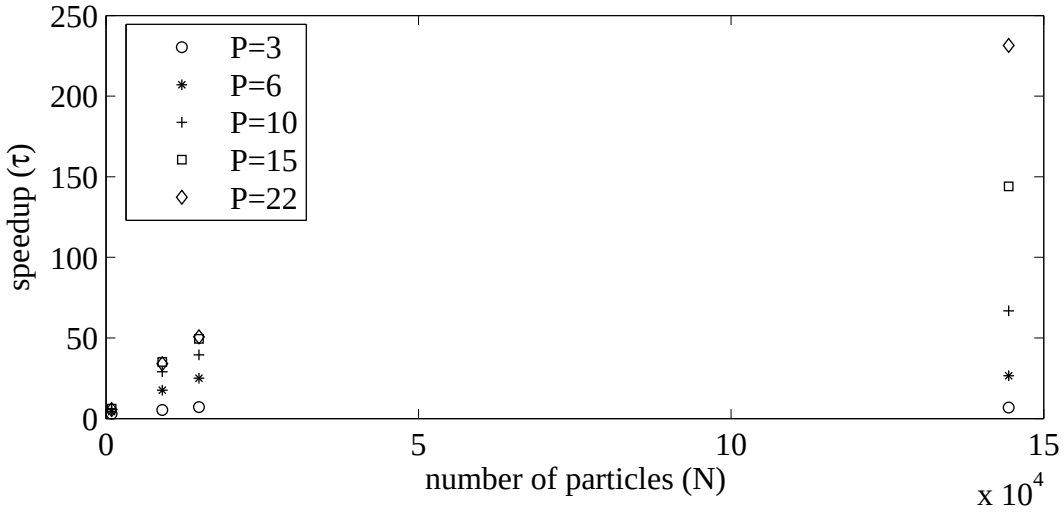
number of processors	single computing machine		multiple computing machines	
	time (h)	speedup τ	time (h)	speedup τ
1	2600	1	2600	1
2	–	–	732.6	3.6
3	943.6	2.8	377.2	6.9
5	–	–	160.6	16.2
7	444.4	5.9	69.2	37.6
10	–	–	38.9	66.9
14	211.6	12.3	21.0	123.6
16	184.1	14.1	15.8	164.6
18	–	–	13.9	187.2
22	159.5	16.3	11.2	231.5
25	–	–	12.8	203.3

Going further on in increasing the problem size, $N = 144400$ particles are then simulated on single and multiple computing machines, see Table 3.4. It is observed that running this simulation on multi-processors belong to a single machine will keep the speedup in the usual range below the ideal case, see again Fig. 3.13b, while changing to multiple machines will enable the simulation to take advantage from the speed of the individual cache memories located locally on each machine. Therefore, the scalability of the computations will increase to exceed the ideal case to the superlinear behavior due to the same pre-mentioned reasons, see Fig. 3.15a. Figure 3.15b allows comparison for different problem sizes. Enlarging the problem size will improve the granularity and minimize the overhead, the case which will decrease the computation time and increase the speedup of the problem computations.

To present our results, we insert them to XPVM for analyzing and monitoring, see Fig. 3.16. XPVM is a *graphical console* and monitor for PVM and provides a graphical interface to the PVM console commands and information, along with several animated views to monitor the execution of PVM programs. These views provide information



(a) influence of the problem size (N)

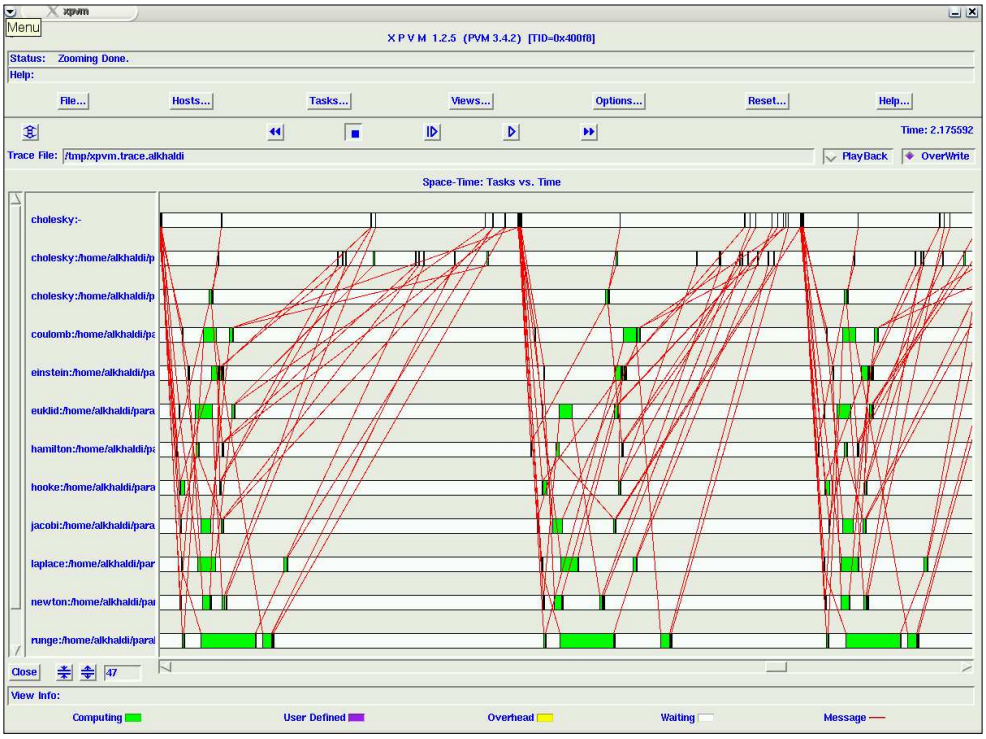


(b) influence of the number of processors (P)

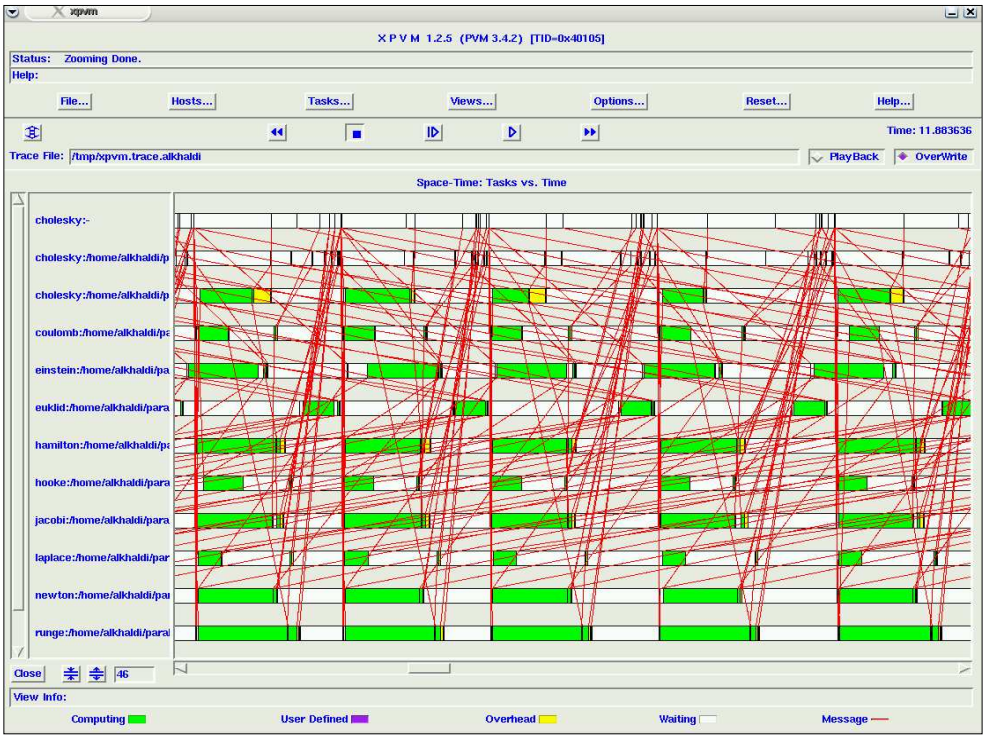
Figure 3.15: The speedup of parallel computations for different number of particles performed on different processors of multiple computing machines

about the interactions among tasks in a parallel PVM program, to assist in debugging and performance tuning.

For comparison purposes, one can see that the performance as well as the speedup of running 9000 particles on 10 machines is much better than that of 916 particles on the same number of machines. The reason is that the communication and data flow are more efficient between the different tasks as the number of particles increases and hence, the relative communication cost will decrease. The decrease in performance of the small size problems compared to the large parallel implementations is due to the disparity of the overhead costs and the communications which can comprise a significant portion of the total execution time of the parallel task.



(a) N=916 particles



(b) N=9000 particles

Figure 3.16: Data-message exchange among 10 processors in XPVM

The strong dependency of the speedup of parallel simulations on the problem size N is recognized in Fig. 3.15b. In Fig. 3.16, if the *computational state* is represented by scanned areas of the green strips, the *overheading* by the yellow strips and the *waiting state* for messages among processors by white ones, then the ratio of these areas with respect to each other represents the size of *granularity* of the system and how much it is fine or coarse. Each of these strips is a direct representative of the performance of any of the working processors in the system. Therefore, the green color appears much more in Fig. 3.16b of 9000 particles than it does in Fig. 3.16a of 916 particles. This means that the performance, and then the speedup, improves with increasing the system size.

Chapter 4

Screening and Particle Segregation

As a practical application of the granular materials, a classification of dry particles according to their size is selected to be investigated and analyzed. *Sieving* or *screening* has been widely used both in industries, as a unit operation for large scale separation of particles according to size, and in laboratories, as a tool for the analysis of particle distribution, usually at a small scale [17, 56].

One of the key operations in the processing of quality particulate solids is the separation of particles according to their *size*. Hundreds of millions of tons of particulate materials are subjected to industrial screening each year and an understanding of the kinetics involved clearly has a great economic significance, see [104]. To date most size separation methods are still dominated by the conventional sieving or screening techniques, which usually employs a screen mesh as multiple *go/no-go* gauges to separate particles.

In this chapter a mathematical and numerical investigation of the particulate motion on an inclined tumbling screen using the discrete element method will be presented. Special attention will be paid to the implementation of the sieve holes, their boundaries and the suggested approaches for allowing particles to pass through apertures or to rebound when approaching the screen surface. A parametric study of different operational variables of the machine and the screen and its influence on the screening efficiency has been conducted. The computational study shows the advantage of using the discrete element method to understand the complex granular separation process.

The operation of particle separation is divided into two main categories, continuous and batch operations. In continuous operation, the particles are continuously fed into the separation unit during the whole separation process. This type of particle separation is usually called '*screening*'. On the other hand, batch operation is used if the particulate material is charged only once. This kind of batch separation is commonly described by the term '*sieving*'.

Screens can be classified into three groups according to their mode of particle movement,

see [115]. These groups are: (a) vibrating screens in which the particulate material moves relative to the screen in a vertical plane, (b) flat screens in which the particulate material moves relative to the screen in the plane of the screen and (c) rotating screens in which the screen surface is cylindrical and the particulate material cascades over the inner surface as the screen is rotated. Furthermore they differ in their shapes, number of decks, screen configurations and design according to what kind and specifications of materials prepared to be screened. In Fig. 4.1 pictures are presented for different models of the TSM machines, see the Allgaier-Werke GmbH website www.allgaier.de.

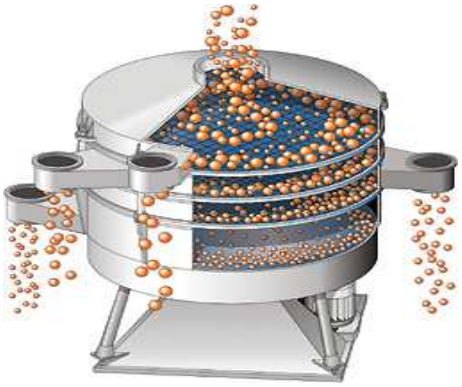


illustration of TSM

six-deck machine

fixing sieves

continuous screening

machines in field

pharmaceutical TSM

Figure 4.1: Picture gallery of different models of the TSM's, see www.allgaier.de

Screeners, which are in the focus of our research, are sifting units which are rotated as material is fed into their interior. The finer particles should fall through the sieve opening and oversized particles are ejected out through certain outlets. Screeners are available in three main types: drum sifter, rectangular deck, and round deck. A rotary round deck separating machine for screening round particles of different sizes is the main problem of interest in this study. To reach our goal of better understanding the processes, a tumbler screening machine has been modelled as a multibody system [97] and investigated in our discrete element simulation program.

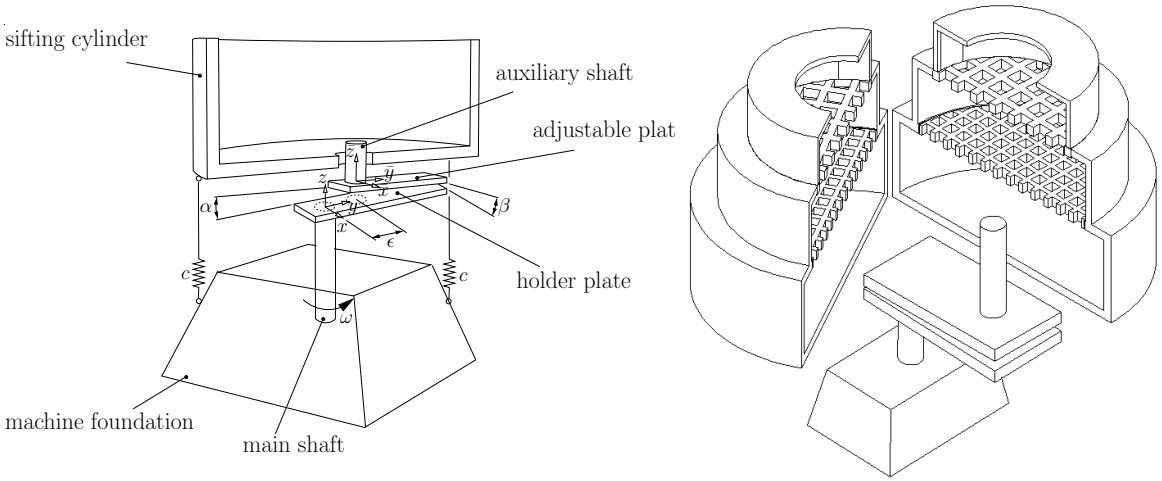
4.1 Description of the tumbler screening machine

The investigated tumbler screening machine (TSM) consists of several main parts, see Fig. 4.2a.

1. **Driving motor:** It runs in different speeds which should be selected depending on the type of material to be separated.

2. **Machine foundation:** It is heavy enough to hold the machine structure and to isolate the environment against vibrations.

3. **Holder and adjustable plates:** The holder plate is horizontal and held by the main rotating shaft of the machine. To create the tumbling motion, an adjustable plate is fixed over the holder plate with certain tilt angles α and β. The overall sifting unit is connected to the adjustable plate by an auxiliary shaft. The inclination angles are chosen to obtain the best efficiency of the separation process.



(a) main parts of the machine (b) internal cross section

Figure 4.2: Tumbler screening machine

4. **Rotating shafts:** The main shaft is driven by a V-belt connected to the driving electrical motor. This shaft and the auxiliary shaft have a certain margin of eccentricity in between and they hold the entire body of the machine and transmit the motion to the main sifting unit.

5. **Sifting meshes:** These separating units consist of a group of potentially different-sized sieves along the successive decks of the machine, see Fig. 4.2b. The undersized particles may pass through the holes. The oversized particles with some of the undersized ones will be rebound from the mesh plane when they strike the obstructed portion of the screen. Of course, the mesh widths are decreasing from the top to the bottom.

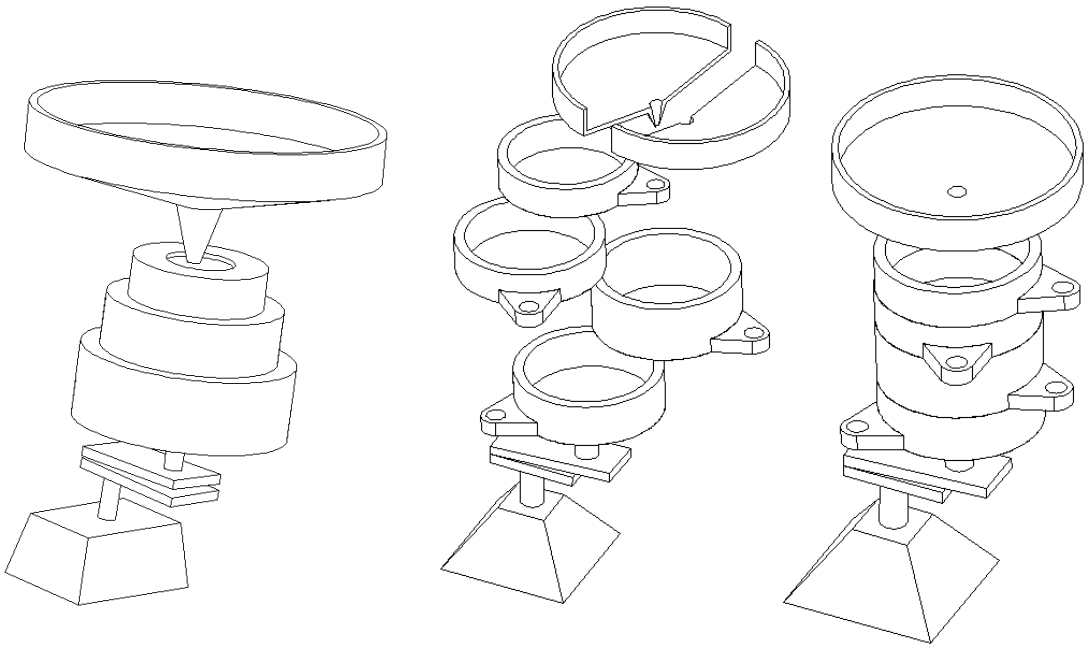
6. **Sifting vessel:** The main cylinder is charged with the bulk material. This vertical rotating vessel can be designed as a uniform or stepped multi level oblique barrel, see Fig. 4.3a. The number of these levels depends directly on the particles to be separated and on the process. Different exits are distributed at each sorting level and located at the outer periphery around the body of the machine, see Fig. 4.3b. A wide variety of machine sizes is available, e.g. produced from Allgaier-Werke GmbH, Uhingen, Germany. Diameters are ranging in their machines from 60cm to 290cm.

7. **Mounting springs:** These springs are attached directly to the body of the sifting vessel. Due to these springs, this vessel is almost prevented from rotation and kept in a very low frequent oscillation mode about its rotation axis. Otherwise, and if there are no springs, the barrel would completely rotate in a full circle around its own axis while the machine is running. Depending on the stiffness of the springs along with the machine speed, the magnitude of oscillation can be determined. Due to the attached springs the barrel hardly rotates.

8. **Feeding container:** In the simulation of continuous screening operation, the TSM is charged by the particle flow through a feeding container. The size of the exit nozzle of the container determines the particle flow rate, see Fig. 4.3. The machines are charged from the top and in the middle of the highest level of the sifting unit.

4.2 Operation and machine movement

The machine is modelled as a multibody system [97]. It consists basically of the machine itself with different layers of sieves and the particulate material to be screened and classified. It will be useful to point out here that the term *particulate material* refers to a granular material used in industry such as rocks, minerals, gravel, grain, seed, medical tablets or other similar matter in this wide family of alternatives.



(a) stepped barrel shape (b) uniform barrel shape

Figure 4.3: Stepped and uniform TSM models

4.2.1 Machine movement

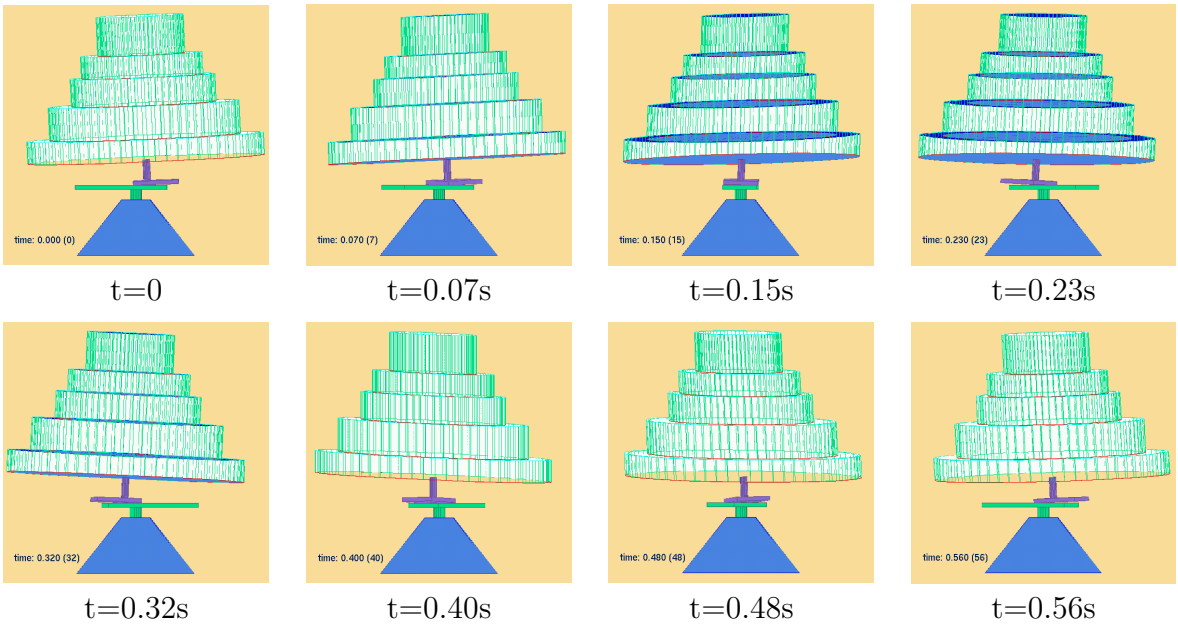
The basic motion of the tumbler screening machine is gyratory, see Fig. 4.4. The angular velocity ω is assumed to be constant during the separation process. Since the barrel of the machine is constrained by a system of four springs, it will prevent rotation around its axis. Therefore, and depending on the spring stiffnesses, the barrel will slightly oscillate around its axis in an oscillatory motion of frequency ω_b.

Using a special axially spring-mounted graphic recorder, the spatial movement of the machine can be represented on paper as a curve of ellipse-like profile for repeatable adjustments. The recording of the screening action permits the optimum operating data to be reproduced, which will assist quality assurance.

4.2.2 Particle movement

The TSM can be used for materials with particle sizes between micrometers and millimeters, such as those used in the chemical, pharmaceutical, foodstuffs and plastics industries. The complicated three-dimensional screening motion of the machine is mainly independent of the material load. This screening motion is adjustable to improve the separation process where experiments are necessary to find optimal process parameters.

As the screening material is continuously fed into the center of the top screen, it is distributed over the entire screening surface from the center towards the periphery. The particles will often travel in a spiral motion, see Fig. 4.5, due to the combination of the

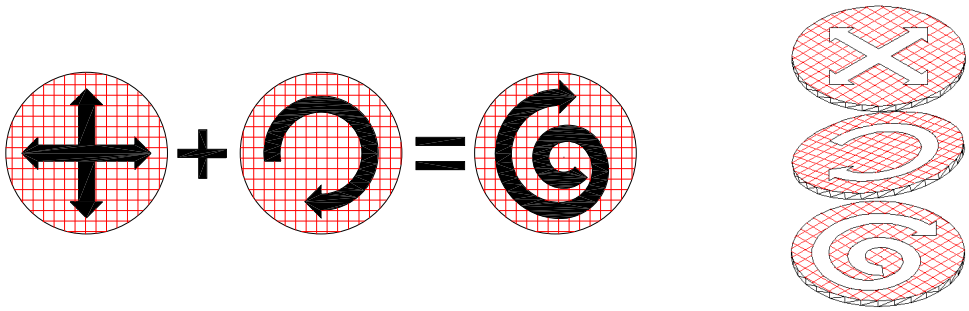


t=0 t=0.07s t=0.15s t=0.23s

t=0.32s t=0.40s t=0.48s t=0.56s

Figure 4.4: Motion of tumbler screening machine

tangential and radial inclination of the screen and to the friction between the particles and the surface area of the mesh, see e.g. [109].



2D view 3D view

Figure 4.5: Spiral motion of the bulk material over the screen surface

In the center, many fine particles will pass through the mesh holes to produce fine-sized fractions. Towards the outside and as the material is fed on to the center of the top screen, the horizontal and vertical accelerations are increasing, causing the oversized particles to move to the periphery of the screen to be discharged through a tangential outlet. These particles are carried to the outlets and guided by adjustable deflectors which influence their flow. This process is repeated on every screen deck.

For a dry particulate classification, the particle segregation flowchart, see Fig. 4.6, clarifies the particle transportation between the different layers of the rotating sifting unit, particle movement, reflections, falling and sorting of the oversized and undersized types of particles inside the tumbling machine during the separation process.

Increasing the rate of the feeding material and adjusting well the screening pattern by

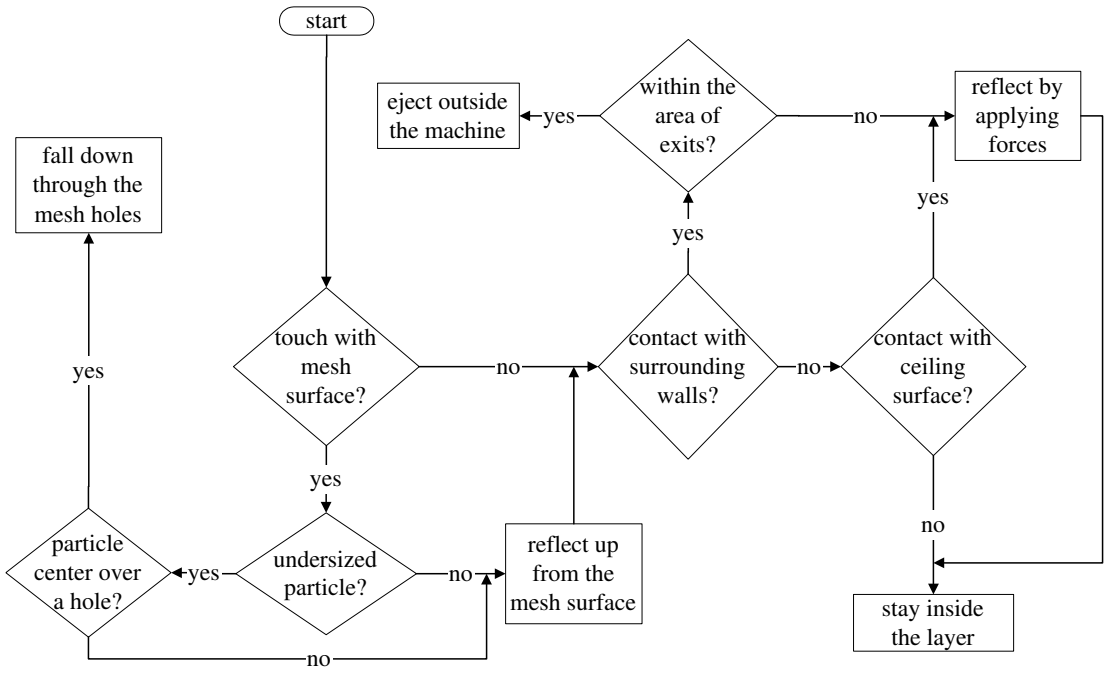


Figure 4.6: Flowchart of the segregation process

controlling and altering the inclination angles and/or the shaft eccentricities, will increase the horizontal and the vertical component of the motion over the screen. Increasing the horizontal component of the motion will usually cause the oversized particles to discharge at a faster rate. On the other hand, the increment in the vertical component along with using some rubber balls under the screen will often reduce the effect of the particle-screen blocking especially for the near-sized particles which will help in dislodging particles that may be held in the interstices of the screen.

4.3 Particle modelling and contact calculations

4.3.1 Particle-to-particle contact

The contact between particles along with the neighborhood search and the contact force computations using the penalty approach are all discussed in detail in Sections 2.2 and 2.3.

4.3.2 Particle-to-mesh contact

In order to derive the mathematical formulation of the system, a global inertial system $K_g\{o, x^g, y^g, z^g\}$ and a local rotating coordinate system $K_l\{c, x^l, y^l, z^l\}$ are defined. The body fixed coordinate system K_l is rotating with the tumbling cylinder with its origin

located at the base of the auxiliary shaft on the inclined adjustable plate, see Fig. 4.7. To check the contacts and calculate forces, the particle positions and velocities should be expressed in this coordinate system.

• Translation vector and rotation matrix

Since we will write the particle position $\mathbf{r}_i$ and the particle velocity $\mathbf{v}_i$ in terms of the rotating coordinate system K_l, it is required to derive the translation vector and the rotation matrix based on the geometrical parameters of the inclined plate α and β and on the angular velocities of the machine ω around a vertical axis of rotation and of the barrel ω_b around its inclined axis.

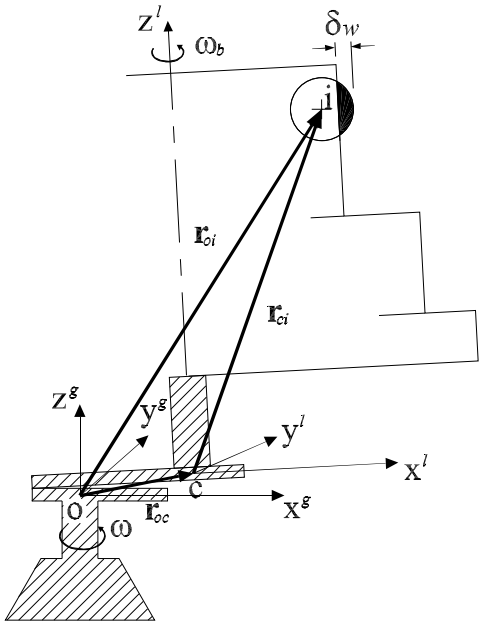


Figure 4.7: Vector description of particles inside the machine

The general form of the translation vector $\mathbf{r}_{oc}^g$ from point o to point c given in the coordinate system K_g is derived using the NEWEUL software [50] as

$$\mathbf{r}_{oc}^g = \begin{bmatrix} u\cos(\omega t) - u\cos(\omega t)\cos\beta + e\sin(\omega t) - \epsilon\sin(\omega t) + \\ \qquad u\sin(\omega t)\sin\alpha\sin\beta - e\sin(\omega t)\cos\alpha \\ u\sin(\omega t) - u\sin(\omega t)\cos\beta - e\cos(\omega t) + \epsilon\cos(\omega t) - \\ \qquad u\cos(\omega t)\sin\alpha\sin\beta + e\cos(\omega t)\cos\alpha \\ u\sin\beta\cos\alpha + e\sin\alpha \end{bmatrix}, \tag{4.1}$$

where u and e are two geometrical dimensions related to the adjustable inclined plate, ϵ is the eccentricity between the fixed main shaft and the rotating auxiliary one.

Similarly, the general form of the rotation matrix $\mathbf{A}_{lg}$ between the coordinate systems can

be written as

$$\mathbf{A}_{lg} = \begin{bmatrix} A^{11} & A^{12} & A^{13} \\ A^{21} & A^{22} & A^{23} \\ A^{31} & A^{32} & A^{33} \end{bmatrix}, \tag{4.2}$$

where the matrix components $A^{ij} : i, j = 1, 2, 3$, are

$$\begin{aligned}
A^{11} =& \cos(\omega_b - \omega)t\cos(\omega t)\cos\beta - \cos(\omega_b - \omega)t\sin(\omega t)\sin\alpha\sin\beta - \\
& \sin(\omega_b - \omega)t\sin(\omega t)\cos\alpha, \\
A^{12} =& \cos(\omega_b - \omega)t\sin(\omega t)\cos\beta + \cos(\omega_b - \omega)t\cos(\omega t)\sin\alpha\sin\beta + \\
& \sin(\omega_b - \omega)t\cos(\omega t)\cos\alpha, \\
A^{13} =& -\cos(\omega_b - \omega)t\cos\alpha\sin\beta + \sin(\omega_b - \omega)t\sin\alpha, \\
A^{21} =& -\sin(\omega_b - \omega)t\cos(\omega t)\cos\beta + \sin(\omega_b - \omega)t\sin(\omega t)\sin\alpha\sin\beta - \\
& \cos(\omega_b - \omega)t\sin(\omega t)\cos\alpha, \\
A^{22} =& -\sin(\omega_b - \omega)t\sin(\omega t)\cos\beta - \sin(\omega_b - \omega)t\cos(\omega t)\sin\alpha\sin\beta + \\
& \cos(\omega_b - \omega)t\cos(\omega t)\cos\alpha, \\
A^{23} =& \sin(\omega_b - \omega)t\cos\alpha\sin\beta + \cos(\omega_b - \omega)t\sin\alpha, \\
A^{31} =& \cos(\omega t)\sin\beta + \sin(\omega t)\sin\alpha\cos\beta, \\
A^{32} =& \sin(\omega t)\sin\beta - \cos(\omega t)\sin\alpha\cos\beta \quad \text{and} \\
A^{33} =& \cos\alpha\cos\beta\ .
\end{aligned} \tag{4.3}$$

From this general case where the machine rotates with ω around the axis of the main shaft while the barrel additionally rotates with ω_b around its own axis which is assumed to be constant, three special cases can be concluded

► **case (1):** The barrel freely rotates around its own axis in absence of the effect of the attached springs with a speed similar to the machine speed, i.e. $\omega_b = \omega$. The translation vector $\mathbf{r}^g_{oc}$ from point o to point c is independent on ω_b and stays the same as in Equation (4.1), while the rotation matrix $\mathbf{A}_{lg}$ between the coordinate systems can be recalculated using Equation (4.2) as

$$\mathbf{A}_{lg} = \begin{bmatrix} \cos(\omega t)\cos\beta - \sin(\omega t)\sin\alpha\sin\beta & \sin(\omega t)\cos\beta + \cos(\omega t)\sin\alpha\sin\beta & -\cos\alpha\sin\beta \\ -\sin(\omega t)\cos\alpha & \cos(\omega t)\cos\alpha & \sin\alpha \\ \cos(\omega t)\sin\beta + \sin(\omega t)\sin\alpha\cos\beta & \sin(\omega t)\sin\beta - \cos(\omega t)\sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}, \tag{4.4}$$

where α and β are the two inclination angles of the adjustable plate of the machine.

► **case (2):** The barrel does not rotate around its own axis, i.e. $\omega_b = 0$, but it does around the axis of the main shaft. This constraint appears due to effect of the attached

springs around the machine. The translation vector $\mathbf{r}_{oc}^{g}$ again stays the same while the rotation matrix $\mathbf{A}_{lg}$ can be expressed by its components using Equations (4.3) as

$$\begin{aligned}
A^{11} =& \cos^2(\omega t)\cos\beta - \sin(\omega t)\cos(\omega t)\sin\alpha\sin\beta + \sin^2(\omega t)\cos\alpha, \\
A^{12} =& \sin(\omega t)\cos(\omega t)\cos\beta + \cos^2(\omega t)\sin\alpha\sin\beta - \sin(\omega t)\cos(\omega t)\cos\alpha, \\
A^{13} =& -\cos(\omega t)\cos\alpha\sin\beta - \sin(\omega t)\sin\alpha, \\
A^{21} =& \sin(\omega)t\cos(\omega t)\cos\beta - \sin^2(\omega t)\sin\alpha\sin\beta - \sin(\omega t)\cos(\omega)t\cos\alpha, \\
A^{22} =& \sin^2(\omega t)\cos\beta + \sin(\omega t)\cos(\omega t)\sin\alpha\sin\beta + \cos^2(\omega t)\cos\alpha, \\
A^{23} =& -\sin(\omega t)\cos\alpha\sin\beta + \cos(\omega t)\sin\alpha, \\
A^{31} =& \cos(\omega t)\sin\beta + \sin(\omega t)\sin\alpha\cos\beta, \\
A^{32} =& \sin(\omega t)\sin\beta - \cos(\omega t)\sin\alpha\cos\beta \quad \text{and} \\
A^{33} =& \cos\alpha\cos\beta \ .
\end{aligned} \tag{4.5}$$

► **case (3):** The barrel rotates *back and forth* in an oscillatory motion around its own axis while in the same time keeps rotating and tumbling around the main machine axis. This oscillatory motion can be run on different frequencies. Normalizing the oscillation frequency ω_b by defining the oscillation factor $K = \omega_b/\omega$, one can study the influence of this motion on the performance of the screening operation.

In this study a linear triangular oscillation mode of the machine cylinder is assumed. This model is investigated as an approximation of the real motion and not to fit exactly the real machine. In this kind of motion the barrel will rotate counterclockwise in the forward direction until $\psi = \psi_{max}$ and return back in clockwise direction until $\psi = 0$ and so on, where ψ is the angular displacement of the barrel, therefore

$$\psi = \omega_b t, \quad \omega_b = K\omega, \quad \psi_{max} = \frac{1}{2}\omega_b t_p \ , \tag{4.6}$$

where t_p is period of oscillation, see Fig. 4.8. The barrel displacement function ψ_i of the ith period of oscillation can be given as

$$\psi_i = \begin{cases} \omega_b(t - t_{s_i}) & t_{s_i} \le t \le t_{s_i} + n\Delta t \ , \\ -\omega_b(t - t_{s_i}) - 2n\Delta t & t_{s_i} + n\Delta t < t \le t_{s_i} + 2n\Delta t \ , \end{cases} \tag{4.7}$$

where $t_{s_i} = 2i(n\Delta t)$ is the starting time of the ith cycle of oscillation. Therefore, the oscillation period $t_p = 2(n\Delta t)$, i.e. the barrel changes its rotational direction every (n) time steps of simulation.

• Particle location

After defining the rotating coordinate system K_l and finding the related translation vector and rotation matrix, the position of any particle i inside the rotating vessel, see again Fig. 4.7, can be expressed as

$$\mathbf{r}_{ci}^{g} = \mathbf{r}_{oi}^{g} - \mathbf{r}_{oc}^{g} \quad \text{or} \quad \mathbf{r}_{ci}^{l} = \mathbf{A}_{lg}(\mathbf{r}_{oi}^{g} - \mathbf{r}_{oc}^{g}) \ . \tag{4.8}$$

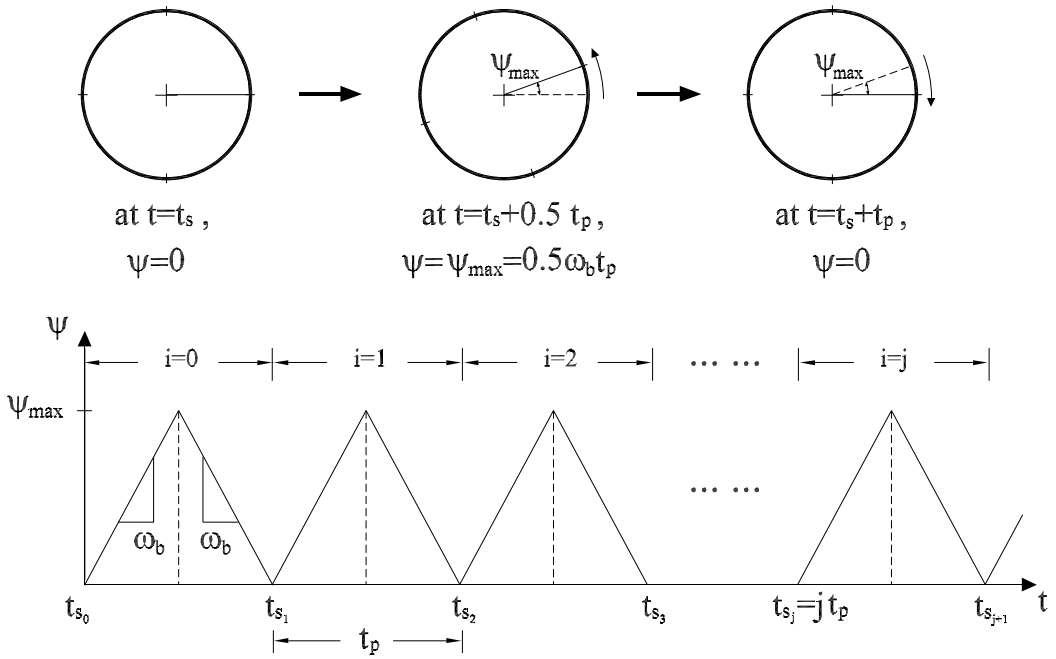


Figure 4.8: Linear triangular oscillation mode of the machine cylinder

Knowing the particle positions in the new coordinate system, the neighboring list is then constructed and, therefore, the overlapping and the contact forces can be determined using Equations (2.37) and (2.36).

• Particle velocity

Similarly, the particle velocity can also be expressed with respect to the rotating coordinate system by differentiating Equation (4.8) once relative to time

$$\dot{\mathbf{r}}^l_{ci} = \mathbf{A}_{lg}(\dot{\mathbf{r}}^g_{oi} - \dot{\mathbf{r}}^g_{oc}) + \dot{\mathbf{A}}_{lg}(\mathbf{r}^g_{oi} - \mathbf{r}^g_{oc}) \ . \tag{4.9}$$

Substituting $\mathbf{v}^l_i = \dot{\mathbf{r}}^l_{ci}$ and $\mathbf{v}^g_i = \dot{\mathbf{r}}^g_{oi}$ in Equation (4.9) for particle velocity we get

$$\mathbf{v}^l_i = \mathbf{A}_{lg}(\mathbf{v}^g_i - \dot{\mathbf{r}}^g_{oc}) + \dot{\mathbf{A}}_{lg}(\mathbf{r}^g_{oi} - \mathbf{r}^g_{oc}) \ . \tag{4.10}$$

The terms $\dot{\mathbf{r}}^g_{oc}$ and $\dot{\mathbf{A}}_{lg}$ in Equation (4.10) represent the rate of change of the translation vector and the rotation matrix with respect to time. By direct differentiation of Equations (4.1) and (4.2), respectively, the rate of the translation vector is given as

$$\dot{\mathbf{r}}^g_{oc} = \omega \begin{bmatrix} -u\sin(\omega t) + u\sin(\omega t)\cos\beta + e\cos(\omega t) - \epsilon\cos(\omega t) + \\ \qquad u\cos(\omega t)\sin\alpha\sin\beta - e\cos(\omega t)\cos\alpha \\ u\cos(\omega t) - u\cos(\omega t)\cos\beta + e\sin(\omega t) - \epsilon\sin(\omega t) + \\ \qquad u\sin(\omega t)\sin\alpha\sin\beta - e\sin(\omega t)\cos\alpha \\ 0 \end{bmatrix} \tag{4.11}$$

and for the rotation matrix as

$$\dot{\mathbf{A}}_{lg} = \frac{d}{dt}\mathbf{A}_{lg} = \begin{bmatrix} \dot{A}^{11} & \dot{A}^{12} & \dot{A}^{13} \\ \dot{A}^{21} & \dot{A}^{22} & \dot{A}^{23} \\ \dot{A}^{31} & \dot{A}^{32} & \dot{A}^{33} \end{bmatrix}, \tag{4.12}$$

where the matrix components $\dot{A}^{ij} : i, j = 1, 2, 3$, are

$$\begin{aligned}
\dot{A}^{11} =& [-\omega\cos(\omega_b-\omega)t\sin(\omega t) - (\omega_b-\omega)\cos(\omega_b-\omega)t\cos(\omega t)]\cos\beta - \\
& [\omega\cos(\omega_b-\omega)t\cos(\omega t) - (\omega_b-\omega)\sin(\omega_b-\omega)t\sin(\omega t)]\sin\alpha\sin\beta - \\
& [\omega\sin(\omega_b-\omega)t\cos(\omega t) + (\omega_b-\omega)\cos(\omega_b-\omega)t\sin(\omega t)]\cos\alpha, \\
\dot{A}^{12} =& [\omega\cos(\omega_b-\omega)t\cos(\omega t) - (\omega_b-\omega)\sin(\omega_b-\omega)t\sin(\omega t)]\cos\beta + \\
& [-\omega\cos(\omega_b-\omega)t\sin(\omega t) - (\omega_b-\omega)\cos(\omega_b-\omega)t\cos(\omega t)]\sin\alpha\sin\beta + \\
& [-\omega\sin(\omega_b-\omega)t\sin(\omega t) + (\omega_b-\omega)\cos(\omega_b-\omega)t\cos(\omega t)]\cos\alpha, \\
\dot{A}^{13} =& (\omega_b-\omega)\sin(\omega_b-\omega)t\cos\alpha\sin\beta + (\omega_b-\omega)\cos(\omega_b-\omega)t\sin\alpha, \\
\dot{A}^{21} =& -[-\omega\sin(\omega_b-\omega)t\sin(\omega t) + (\omega_b-\omega)\cos(\omega_b-\omega)t\cos(\omega t)]\cos\beta + \\
& [\omega\sin(\omega_b-\omega)t\cos(\omega t) + (\omega_b-\omega)\cos(\omega_b-\omega)t\sin(\omega t)]\sin\alpha\sin\beta - \\
& [\omega\cos(\omega_b-\omega)t\cos(\omega t) - (\omega_b-\omega)\sin(\omega_b-\omega)t\sin(\omega t)]\cos\alpha, \\
\dot{A}^{22} =& -[\omega\sin(\omega_b-\omega)t\cos(\omega t) + (\omega_b-\omega)\cos(\omega_b-\omega)t\sin(\omega t)]\cos\beta - \\
& [-\omega\sin(\omega_b-\omega)t\sin(\omega t) + (\omega_b-\omega)\cos(\omega_b-\omega)t\cos(\omega t)]\sin\alpha\sin\beta + \\
& [-\omega\cos(\omega_b-\omega)t\sin(\omega t) - (\omega_b-\omega)\cos(\omega_b-\omega)t\cos(\omega t)]\cos\alpha, \\
\dot{A}^{23} =& (\omega_b-\omega)\cos(\omega_b-\omega)t\cos\alpha\sin\beta - (\omega_b-\omega)\sin(\omega_b-\omega)t\sin\alpha, \\
\dot{A}^{31} =& -\omega\sin(\omega t)\sin\beta + \omega\cos(\omega t)\sin\alpha\cos\beta, \\
\dot{A}^{32} =& \omega\cos(\omega t)\sin\beta + \omega\sin(\omega t)\sin\alpha\cos\beta \quad \text{and} \\
\dot{A}^{33} =& 0\ .
\end{aligned} \tag{4.13}$$

In the case of $\omega_b = \omega$, direct differentiation of Equation (4.4) will give

$$\dot{\mathbf{A}}_{lg} = \omega \begin{bmatrix} -\sin(\omega t)\cos\beta - \cos(\omega t)\sin\alpha\sin\beta & \cos(\omega t)\cos\beta - \sin(\omega t)\sin\alpha\sin\beta & 0 \\ -\cos(\omega t)\cos\alpha & -\sin(\omega t)\cos\alpha & 0 \\ -\sin(\omega t)\sin\beta + \cos(\omega t)\sin\alpha\cos\beta & \cos(\omega t)\sin\beta + \sin(\omega t)\sin\alpha\cos\beta & 0 \end{bmatrix}. \tag{4.14}$$

Similarly, the case of the no local rotation of the barrel, i.e. $\omega_b = 0$, direct differentiation

of Equation (4.5) will give

$$\begin{aligned}
\dot{A}^{11} =& -\omega\cos(2\omega t)\sin\alpha\sin\beta + \omega\sin(2\omega t)[\cos\alpha - \cos\beta], \\
\dot{A}^{12} =& -\omega\sin(2\omega t)\sin\alpha\sin\beta - \omega\cos(2\omega t)[\cos\alpha - \cos\beta], \\
\dot{A}^{13} =& \omega\sin(\omega t)\cos\alpha\sin\beta - \omega\cos(\omega t)\sin\alpha, \\
\dot{A}^{21} =& -\omega\sin(2\omega t)\sin\alpha\sin\beta - \omega\cos(2\omega t)[\cos\alpha - \cos\beta], \\
\dot{A}^{22} =& \sin^2(\omega t)\cos\beta + \sin(\omega t)\cos(\omega t)\sin\alpha\sin\beta + \cos^2(\omega t)\cos\alpha, \\
\dot{A}^{22} =& \omega\cos(2\omega t)\sin\alpha\sin\beta - \omega\sin(2\omega t)[\cos\alpha - \cos\beta], \\
\dot{A}^{23} =& -\omega\cos(\omega t)\cos\alpha\sin\beta - \omega\sin(\omega t)\sin\alpha, \\
\dot{A}^{31} =& -\omega\sin(\omega t)\sin\beta + \omega\cos(\omega t)\sin\alpha\cos\beta, \\
\dot{A}^{32} =& \omega\cos(\omega t)\sin\beta + \omega\sin(\omega t)\sin\alpha\cos\beta \quad \text{and} \\
\dot{A}^{33} =& 0\ .
\end{aligned} \tag{4.15}$$

Since the tumbling screening machine consists from four parts collected together to form the assembled shape of the machine, i.e. the base, the horizontal holder plate, the inclined adjustable plate and the barrel of the machine, the rotation matrices and translation vectors of the first three parts are also derived using the NEWEUL software [50] as

► **upper adjustable plate:** The translation vector is

$$\mathbf{r}^g_{oc} = \begin{bmatrix} u\cos(\omega t) - u\cos(\omega t)\cos\beta + e\sin(\omega t) - \epsilon\sin(\omega t) + \\ \qquad u\sin(\omega t)\sin\alpha\sin\beta - e\sin(\omega t)\cos\alpha \\ u\sin(\omega t) - u\sin(\omega t)\cos\beta - e\cos(\omega t) + \epsilon\cos(\omega t) - \\ \qquad u\cos(\omega t)\sin\alpha\sin\beta + e\cos(\omega t)\cos\alpha \\ u\sin\beta\cos\alpha + e\sin\alpha \end{bmatrix}, \tag{4.16}$$

where u and e are two geometrical dimensions related to the adjustable inclined plate, ϵ is the eccentricity between the fixed main shaft and the rotating auxiliary one. The rotation matrix can be also written as

$$\mathbf{A}_{lg} = \begin{bmatrix} \cos(\omega t)\cos\beta - \sin(\omega t)\sin\alpha\sin\beta & \sin(\omega t)\cos\beta + \cos(\omega t)\sin\alpha\sin\beta & -\cos\alpha\sin\beta \\ -\sin(\omega t)\cos\alpha & \cos(\omega t)\cos\alpha & \sin\alpha \\ \cos(\omega t)\sin\beta + \sin(\omega t)\sin\alpha\cos\beta & \sin(\omega t)\sin\beta - \cos(\omega t)\sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}. \tag{4.17}$$

► **lower holder plate:** Similarly, the translation vector $\mathbf{r}^g$ is

$$\mathbf{r}^g = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.18}$$

and the corresponding rotation matrix is

$$\mathbf{A}_{lg} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) & 0 \\ -\sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.19}$$

► **machine foundation:** The translation vector is simply $\mathbf{r}^g = \mathbf{0}$ and the corresponding rotation matrix is $\mathbf{A}_{lg} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

4.3.3 Contact forces with the mesh

After all positions and velocities of the particles are determined in the rotating coordinate system K_l, the contact forces have to be calculated. The particle interactions inside the machine could be as *particle-particle*, *particle-walls* and *particle-mesh* contacts, see Fig. 4.9. The first two types were discussed in detail in Section 2.2. In this section we will focus on the contact with the machine sieves.

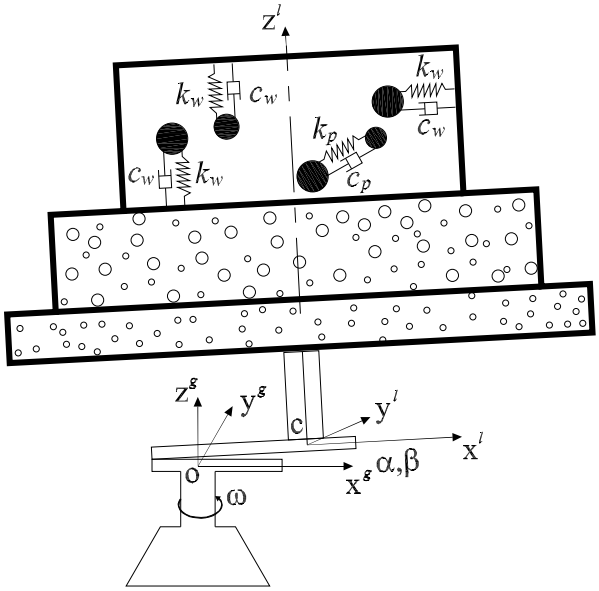


Figure 4.9: Particle contact models with walls, ceilings and sieves of the machine

The particle transportation through the mesh depends directly on the position of the specified particle with respect to the hole boundaries, see [3]. In reality, the mesh is built of a combination of intersecting wires over which the particles will have contact. It happens that the particle contacts with the mesh wires and is reflected in different directions around the hole. This will depend on the exact point on the wire where the particle hits. Taking into consideration all those precise contacts with the mesh wires will lead to many calculations and infeasible computational efforts. In order to reduce these computations while still reaching physical results, another less computationally expensive approach is suggested. This approach depends on defining a reduced mesh hole which is only used in simulation, to consider all contact possibilities and adjust the number of passed and reflected particles in a heuristic way. This method assumes that the undersized particles pass through the mesh if they are small enough and their centers are over the reduced hole, otherwise they are reflected. The mesh thickness is set to zero.

The reduction in the mesh hole diameter can be adjusted to agree with actual sieving observations of the real machine.

Two different models for defining the reduced mesh hole are suggested, these models are

▶ **model (1):** In this model the radius of the undersized particle r is taken into consideration. In this model the undersized particle will not fall down if its center is on the perimeter of the reduced hole *ghij*, but it should be at least at a distance of r inside the periphery in order to fall, i.e. the center of the particle have to be inside the square area *klmn*, see Fig. 4.10a. The *adjustable hole clearance* b is a parameter defined to be used as a simulation parameter to adjust the size of the mesh openings and can be written as

$$b = \zeta(w - r) \ , \qquad 0 \leq \zeta \leq 1 \ , \tag{4.20}$$

where ζ is a non-dimensional scaling factor which keeps the value of the parameter b within certain limits, i.e. $0 \leq b \leq (w - r)$.

▶ **model (2):** This model is a special case of the previous model where $r = 0$. It takes into consideration the position of the undersized particles with respect to the boundaries of the reduced hole, see Fig. 4.10b. The undersized particle will fall down only if its center is on or inside the perimeter of the reduced square hole *ghij*, otherwise it reflects. The value of b can be written as

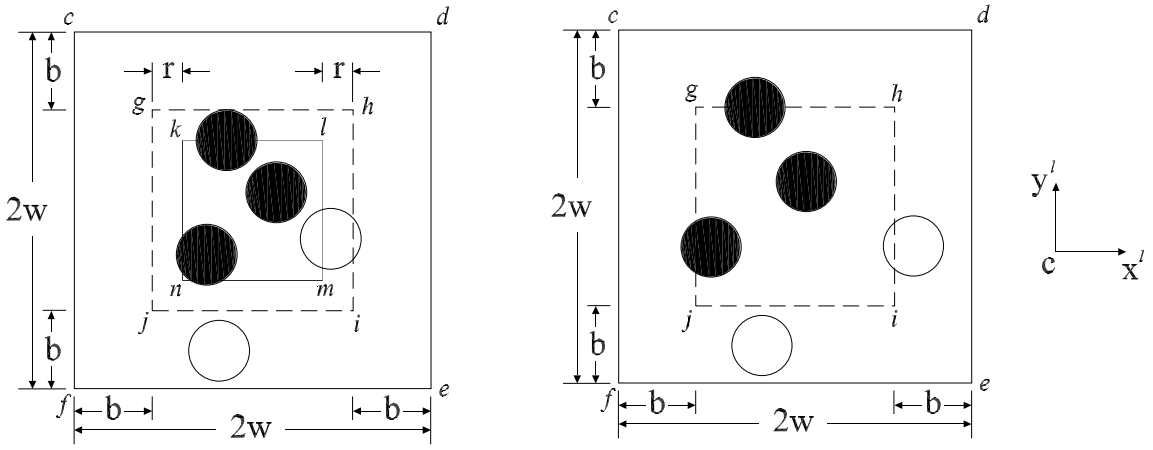
$$b = \zeta w \ , \qquad 0 \leq \zeta \leq 1 \ . \tag{4.21}$$

The value of b is within the limits $0 \leq b \leq w$. For $\zeta = 0$ the hole is maximally opened as in the hole cdef and, therefore, all particles over it will fall. In the case of $\zeta = 1$ the hole will be totally closed and hence no sieving will occur.

Although that using of the first model is more precise than second one, we can not consider the second model be far from the correctness. Some of our simulations are performed to investigate these two models. It was found that value of $b = 0.6w$, for example, in the second model could mostly be equivalent or almost close to a value of $b = 0.8(w - r)$ in the first model. Therefore, some of our results are based on the first model while others are considering the second model for simulating the screening operation.

The direction of the velocity vector of the particles has without doubt a major effect on the reflected particles and then on the number of the particles which will pass down through the mesh holes. The direction of the velocity vector will determine the exact point where the particle will touch the plane of the mesh.

The contact forces with the machine walls and the ceiling are computed analoguely to the particle-particle forces. More interesting is the contact with the mesh surface. For practical purposes, sifters can be designed in many shapes according to the desired holes having e.g. circular, square or polygonal shapes. Here we are interested in square-patterned meshes. The number of holes of barrel k in a square mesh depends directly on the mesh



(a) first model ($r \neq 0$) (b) second model ($r = 0$)

Figure 4.10: Different models of the reduced hole construction. Black balls will fall down while the white ones reflect from the mesh surface.

radius R_k, the width of the holes $2w$ and width of the solid part g between the different rows of the holes, see Fig. 4.11,

$$m_k = \frac{R_k}{2w_k + g_k} + 1 \ , \tag{4.22}$$

where m_k is the number of holes along the radius of the mesh plate. Since the radial number of holes is always an integer, m_k should be rounded to the nearest integer n_k.

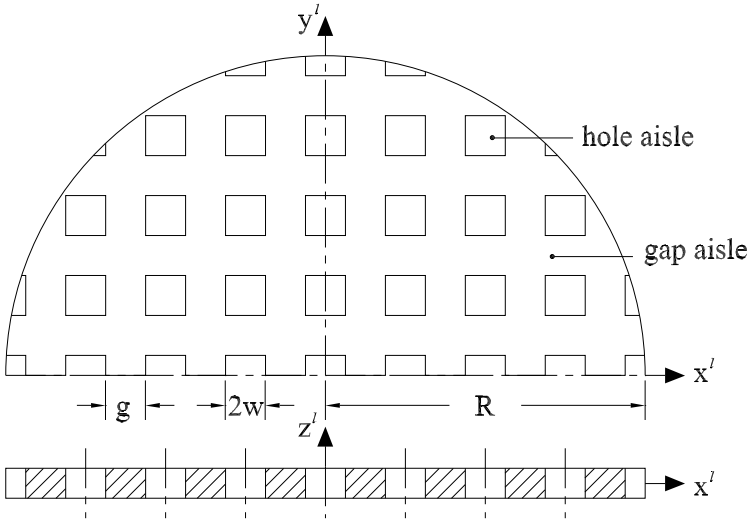


Figure 4.11: Geometrical description of the square mesh

The mesh plate consists of $(2n_k - 1)$ aisles of square holes in the $\bar{x}$ direction and the same number of aisles holes in the $\bar{y}$ direction. It also consists of $(2n_k - 2)$ aisles of gaps and the identical number of aisles in both x^l and y^l directions, respectively.

The particle-mesh contact should be detected in both directions x^l and y^l. The particle will be considered to be reflected if it is in contact with the mesh surface and its center

is over the solid part and not over a reduced hole, otherwise it will fall through the mesh if it is small enough. At any level k, consider the points a and b on the borders of two successive holes on the mesh surface, see Fig. 4.12,

$$x_a^l = w + j(2w + g) \ , \quad x_b^l = x_a^l + g \ , \quad j = -(n_k - 1), ..., \ j = (n_k - 2) \ , \quad (4.23)$$

where j is an integer counter over the number of the gap aisles in the mesh surface along the x^l direction. To check the particle position with respect to the mesh holes in x^l direction, a flag integer c_x is defined as

$$c_x = \begin{cases} 0 & \text{if } (x_a^l - b) \leq r_x^l \leq (x_b^l + b), \\ 1 & \text{else,} \end{cases} \quad (4.24)$$

where r_x^l is the x-coordinate of the particle position in the x^l direction. Reduced holes with adjustable clearance b help in controlling the falling particles through the mesh holes during the simulation process. The adjustable parameter b is selected to adjust the simulation results in order to be in a good agreement with experimental observations which allow predictions for further computations. To run the simulations and check our codes, always a value of $b = 0.8w$ is used in our study. It will be most interesting for the future comparisons to see how this value must be chosen correctly.

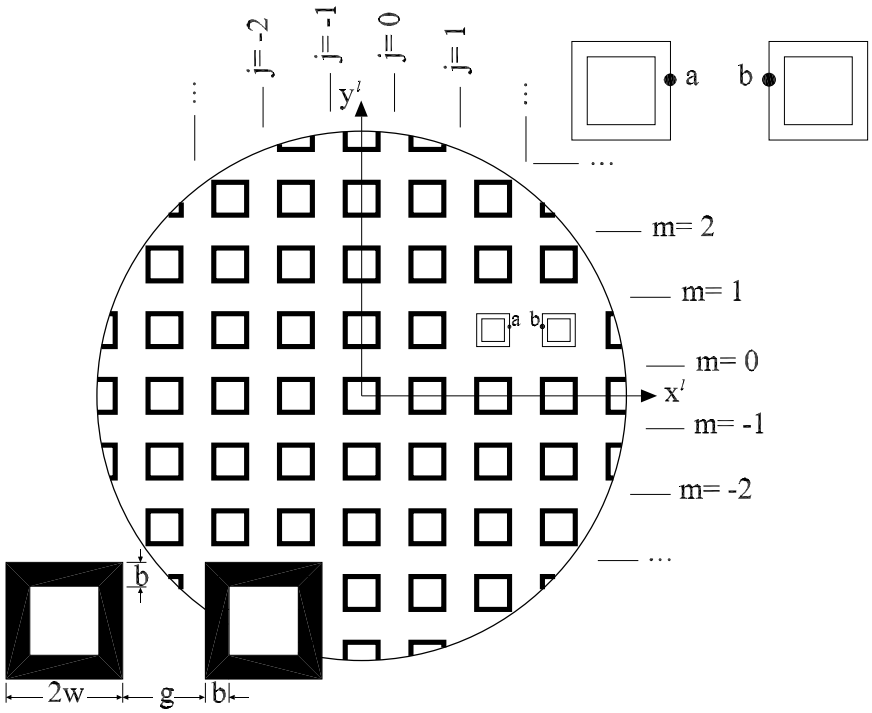


Figure 4.12: Contact detection with mesh surface

In the same way, the y^l direction is investigated yielding c_y. In order to determine the contact forces with the mesh surface, it is checked if the particle is in direct contact with

the solid part of the mesh or if it is over a hole

$$c_x c_y = \begin{cases} 1 & \text{particle potentially falls down,} \\ 0 & \text{particle is reflected.} \end{cases} \tag{4.25}$$

All oversized particles will definitely reflect up from the mesh surface. Some of the undersized particles will reflect also upward when they hit the solid part of the mesh and not a hole, otherwise they will fall down.

In case of friction, the approaching angle of contact between the particle and the mesh is different from the reflection angle over the surface of the mesh. The direction of the particle velocity with respect to the machine at the instant of contact determines the direction of the frictional force in the tangential plane of the mesh surface. Therefore, the reflection angle is determined according to the normal and frictional force components between the particle and the mesh plane.

The friction force affects the particles and tries to reduce their velocities. Here, the friction force between the particle and the mesh surface is usually in the direction of their relative velocity. The tangential component of the relative velocity of particle j in the mesh plane can be written as

$$v_{t_j} = \sqrt{v_{x_j^l}^2 + v_{y_j^l}^2} \ . \tag{4.26}$$

The unit vector of tangential velocity can be expressed for $v_{t_j}^l \neq 0$ as

$$\mathbf{t}_j = \begin{bmatrix} t_{x^l} \\ t_{y^l} \\ 0 \end{bmatrix}_j = \frac{1}{v_{t_j}^l} \begin{bmatrix} v_{x^l}^l \\ v_{y^l}^l \\ 0 \end{bmatrix}_j \ . \tag{4.27}$$

The force between particle j and the mesh surface in the normal direction to the mesh plane can be written as

$$\mathbf{N}_j^l = N_j \mathbf{n}_j = |f_{z_j^l}| \mathbf{n}_j \ , \tag{4.28}$$

where $\mathbf{N}_j^l$ and $\mathbf{n}_j$ are the normal contact force and normal unit vector to the mesh plane in the K_l coordinate system, respectively. Using Coulomb's law, the tangential friction force $\mathbf{T}$ is

$$\mathbf{T}_j^l = -\mu_d N_j \mathbf{t}_j = \frac{-\mu_d |f_{z_j^l}|}{v_{t_j}^l} \begin{bmatrix} v_{x^l}^l \\ v_{y^l}^l \\ 0 \end{bmatrix}_j = \frac{-\mu_d}{v_{t_j}^l} |k_w \delta_{m_j} - c_w v_{z_j^l}^l| \begin{bmatrix} v_{x^l}^l \\ v_{y^l}^l \\ 0 \end{bmatrix}_j \ , \tag{4.29}$$

where μ_d is the dynamic coefficient of friction between particles and the mesh surface, δ_m is the particle-mesh overlap, k_w and c_w are the elastic spring stiffness and the viscous damping coefficient with the mesh plane, respectively. Using Equations (4.28) and (4.29), the overall contact force $\mathbf{F}_j^l$ between particle j and the mesh surface measured in the

rotating coordinate system K_l is then

$$\mathbf{F}_j^l = \mathbf{N}_j^l + \mathbf{T}_j^l = |k_w\delta_{m_j} - c_w v_{z_j^l}^l| \begin{bmatrix} \frac{-\mu_d}{v_t^l} v_{x^l}^l \\ \frac{-\mu_d}{v_t^l} v_{y^l}^l \\ \mathrm{SIGN}(k_w\delta_m - c_w v_{z^l}^l) \end{bmatrix}_j , \tag{4.30}$$

where the function SIGN(q) is 1 for $q > 0$ and otherwise -1. This force can be also expressed in the global coordinate system K_g as

$$\mathbf{F}_j^g = \mathbf{A}_{gl}\mathbf{F}_j^l = \mathbf{A}_{lg}^T\mathbf{F}_j^l , \tag{4.31}$$

where $\mathbf{A}_{gl} = \mathbf{A}_{lg}^{-1} = \mathbf{A}_{lg}^T$ is due to the orthogonality of the rotation matrix.

4.3.4 Numerical time integration

To solve the differential equations numerical time integration is necessary. To calculate the trajectories of particles and determine their new positions and orientations, different integrators can be used. Algorithms developed by Verlet in 1967 are among the most popular in molecular dynamics, i.e. Verlet integration, velocity-Verlet and leapfrog-Verlet, see Section 2.1.1. These integrators offers better stability than the simpler Euler integration methods with often sufficient accuracy and stability. The new position and orientation of particle i at the time step $(m+1)$ are

$$\mathbf{Y}_i^{m+1} = 2\mathbf{Y}_i^m - \mathbf{Y}_i^{m-1} + h^2\ddot{\mathbf{Y}}_i^m , \tag{4.32}$$

where $\mathbf{Y}_i$ denotes the generalized coordinates of body i and h is the integration time step.

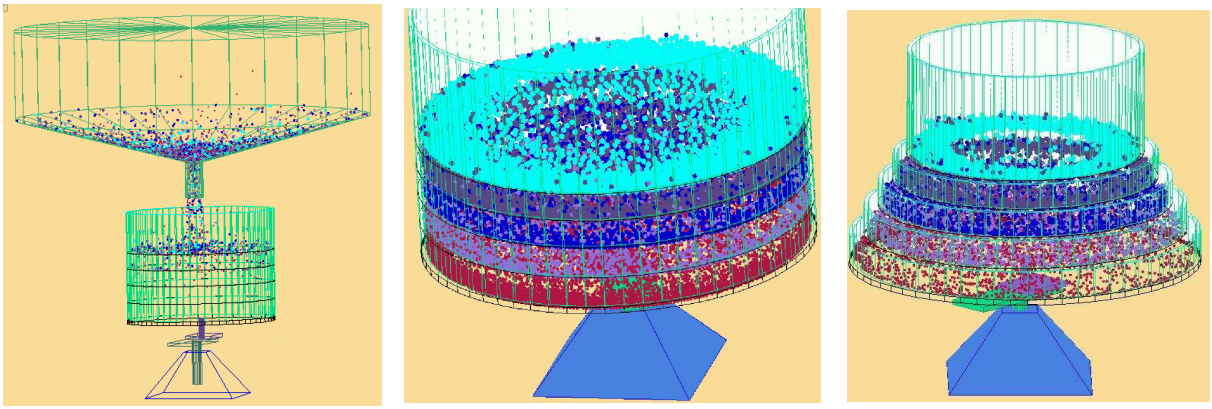
4.4 Parametric study and simulation results

Different simulations have been performed to investigate screening phenomena in the rotating tumbling machine. Particle distribution and sifting rates of the separated particles have been studied. Screening phenomena are very sensitive to the machine parameters, e.g., plate inclinations, shaft eccentricities and size of the mesh openings and rotational speed of the machine.

Due to the complicated motion of the particles over the screen surface and the various factors that influence such a motion, the understanding of the actual mechanisms involved in screening is still in its infancy. Attempts in the past to describe the performance of screening processes mathematically have adopted either a probabilistic approach or a kinetic approach [47]. Most of the attempts have been limited to defining the frequency

and amplitude of vibration, and possibly the direction of motion of the screen surface [107].

In our work, the particles have been tested for both batch and continuous feeding in a uniform and a stepped multi level oblique vertical cylinder. For continuous screening, a package of 3483 particles has been tested and the efficiency of machine has been recorded for different machine conditions, see Fig. 4.13a. Our simulation codes have also been tested for a higher number of particles which requires greater computational effort and more simulation time. These simulations are observed carefully through animations which show an acceptable separation process over the different layers of the machine, see Fig. 4.13b,c. The finer particles frequently fall through the sieve openings while the oversized particles should be ejected through certain outlets located around the machine body. Some of the undersized particles are unintentionally forced to travel outside the machine together with the sorted ones. Those unavoidable and also undesired particles, which we call here *gangue particles* reduce the efficiency of the machine. For an efficient separation process, the selection of proper operating conditions reduces the number of these gangue particles.



(a) continuous, 3483 bodies (b) batch, 27130 bodies (c) batch, 7688 bodies

Figure 4.13: Continuous and batch screening using a TSM

Mathematically, to measure machine efficiency, either the *number-of-particles* or the *mass-of-particles* can be adopted. Depending on the number of particles, the characteristic value c, which is a kind of measure of the machine performance, can be expressed as

$$c = \frac{100}{M} \sum_{i=1}^{M} c_i \ , \qquad c_i = \frac{s_i}{s_i + g_i} \ , \tag{4.33}$$

where c_i is the individual efficiency of layer i of the machine, s_i and g_i are the number of the oversized sorted and the undersized gangue particles at level i of the machine, respectively, M is the total number of barrels in the machine where $i = 1$ is the lowest level and $i = M$ the highest one, see Fig. 4.14. Since there are no gangue particles in the

lowest level, we have $g_1 = 0$, therefore Equation (4.33) can be written as

$$c = \frac{100}{M}\left(1 + \sum_{i=2}^{M} c_i\right) \ . \tag{4.34}$$

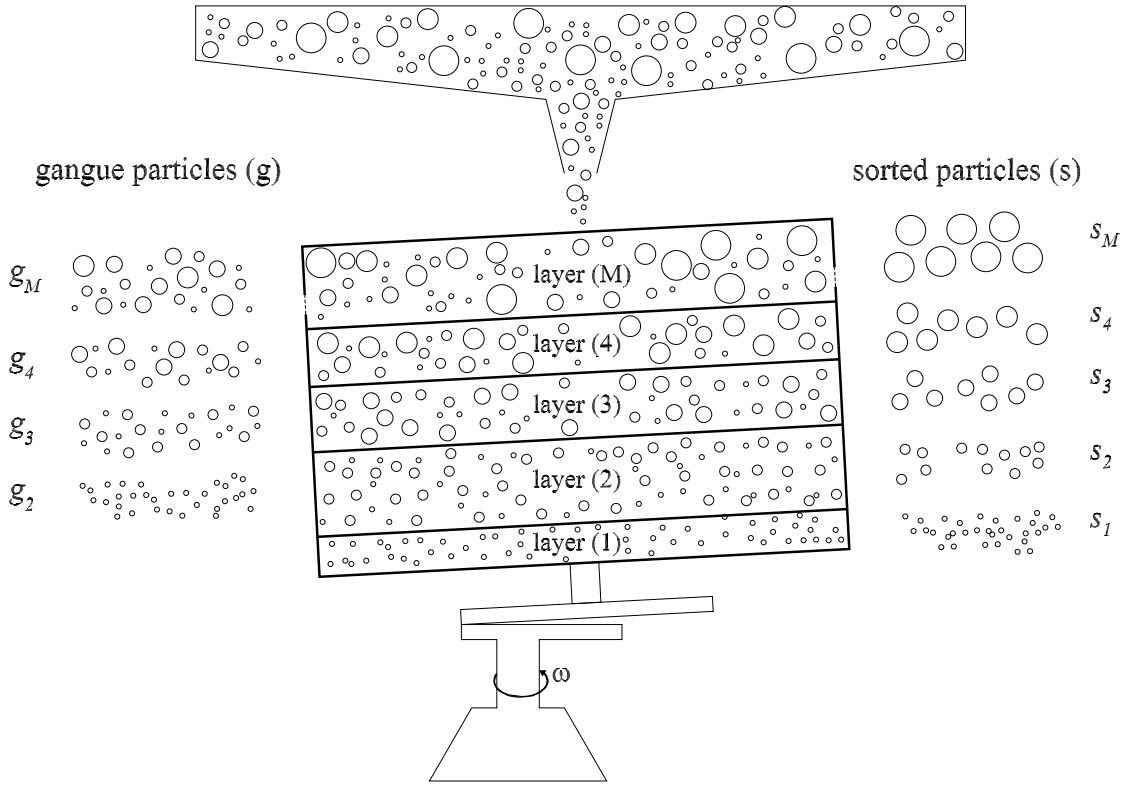


Figure 4.14: Screening process using tumbling machine of M-decks. The sorted particles are to the right while the undesired gangue particles are to the left of the machine

If the mass-of-particles is adopted as the base of efficiency calculation, then Equation (4.33) should be built again upon the mass measurements as

$$c = \frac{100}{M}\sum_{i=1}^{M} c_i \ , \qquad c_i = \frac{m_{s_i}}{m_{s_i} + m_{g_i}} \ , \tag{4.35}$$

where m_{s_i} and m_{g_i} are the total mass of the oversized sorted and the undersized gangue particles at level i of the machine, respectively. These masses can be calculated as

$$m_{s_i} = \sum_{k=1}^{s_i} \rho V_{s_k} = \frac{4}{3}\pi\rho \sum_{k=1}^{s_i} \rho r_{s_k}^3 \ , \tag{4.36}$$

where r_{s_k} is the radius of the sorted particle k assuming round-shaped particles. The radius r_{s_k} is always constant for any layer i since the sorted particles are identical in size, therefore Equation (4.36) can be written as

$$m_{s_i} = \frac{4}{3}\pi\rho s_i r_{s_k}^3 \ , \qquad 1 \le i \le M \ , \tag{4.37}$$

where ρ is the mass density of the particles in the machine assuming they are of the same material. On the other hand, the total mass of the gangue particles of the layer i can be similarly written as

$$m_{g_i} = \sum_{k=1}^{g_i} \rho V_{g_k} = \frac{4}{3}\pi\rho \sum_{k=1}^{g_i} \rho r_{g_k}^3 \ , \qquad 2 \leq i \leq M \ , \tag{4.38}$$

where r_{g_k} is the radius of the gangue particle k assuming spherical particles. It can be easily recognized that the mass of the gangue particles in the lowest level is not counted since always $g_1 = 0$.

The machine efficiency c would be 100% if the number of gangue particles is $g_i = 0$ in the different layers of the machine. Although some decks of the machine are more efficient than others during the sorting process, it is difficult to find a clear prediction explaining this disparity. The weight and size ratio of the different particles in the mixture, the material properties and the operation conditions influence this phenomenon.

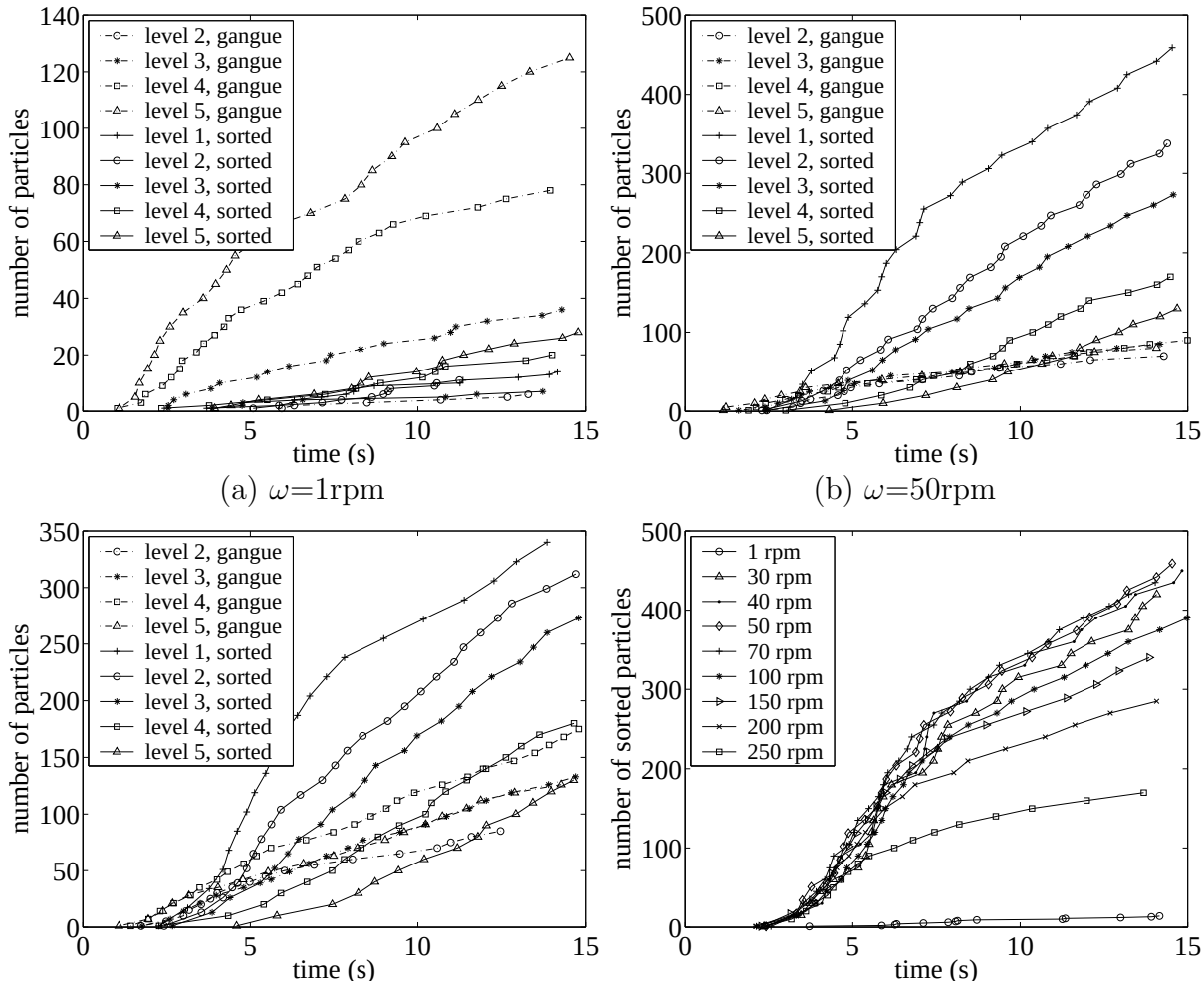
Continuous screening operation and batch sieving are simulated. Larger particle numbers can be investigated but the focus is first put to the basic mechanisms and relations. These particles consists of a mixture of five different sizes with their numbers and radii as: (1210, 10mm), (832, 12mm), (611, 14mm), (467, 16mm) and (363, 18mm). Four squared-pattern sieves of hole dimensions $w = \{11, 13, 15, 17\}$mm are used in a uniform-radius TSM with $R = 80$cm. The ratio $w/g = 1$ for the sieve wires for all decks of the machine, i.e. $g = \{11, 13, 15, 17\}$mm.

4.4.1 Influence of the machine speed

The tumbling screening machine has the ability to run in a range of angular velocities. This angular speed of the machine ω, which is assumed as being constant during the sorting operation process, has a great influence on the sifting rate of the mixed particles at each of the different sorting levels of the machine. These simulations are run for $\omega_b = \omega$. Using computer simulation enables us to change the machine speed ω over a wide range in order to study its effect on the machine performance.

The angular velocity ω should not be too high in order to avoid too fast motion of particles. Too high velocities of the particles will decrease the separation rate of the mixture and reduce the chance for undersized particles to fall down through the mesh openings. On the other side the rotational speed should not be too slow since low separation rates and a big number of undesired gangue particles can then be expected. This is due to the fact that the particles experience not enough mixing due to their small velocities. This insufficient mixing reduces the chance of the undersized particles to fall through the mesh, see Fig. 4.15.

In order to reach the best rate of particle separation, we have to find the rotational speed



(a) ω=1rpm (b) ω=50rpm

(c) ω=150rpm (d) lowest sorting level

Figure 4.15: Influence of rotational speed on the particle sifting rates at different sorting levels of the machine in continuous screening operation, $\alpha = 1°$, $\beta = 0.65°$, $w/g = 1$, $b = 0.8w$ and feeding rate 147 particles per second.

of the machine which maximizes the number of sorted particles and minimizes the gangue ones. In order to do this, we simulate our problem for different angular velocities and measure the efficiency of each set of geometrical, contact and material parameters. It is observed that the sorting under very low rotational velocity leads to very bad separation rates. The main reason for this is that the mesh is blocked soon. This would be a disastrous situation for an industrial process. The number of undesired gangue particles are even much higher than those of desired sorted ones in most of the machine levels, see Fig. 4.15a. Low efficiency of $c = 40\%$ is recorded for this case.

On the other hand, a relatively high velocity of 150rpm leads to better results and higher efficiency of about $c = 72\%$, see Fig. 4.15c. Going further, the very high velocity of 250rpm will decrease the efficiency again to $c = 58\%$. It is observed that the intermediate speed of 50rpm gives the best sifting rates with maximum efficiency about $c = 78\%$ during the continuous screening process, see Fig. 4.16. From this figure, it is also clear that for

all machine speeds the efficiency starts low and approaches its steady operating mode in about $10 \div 20s$.

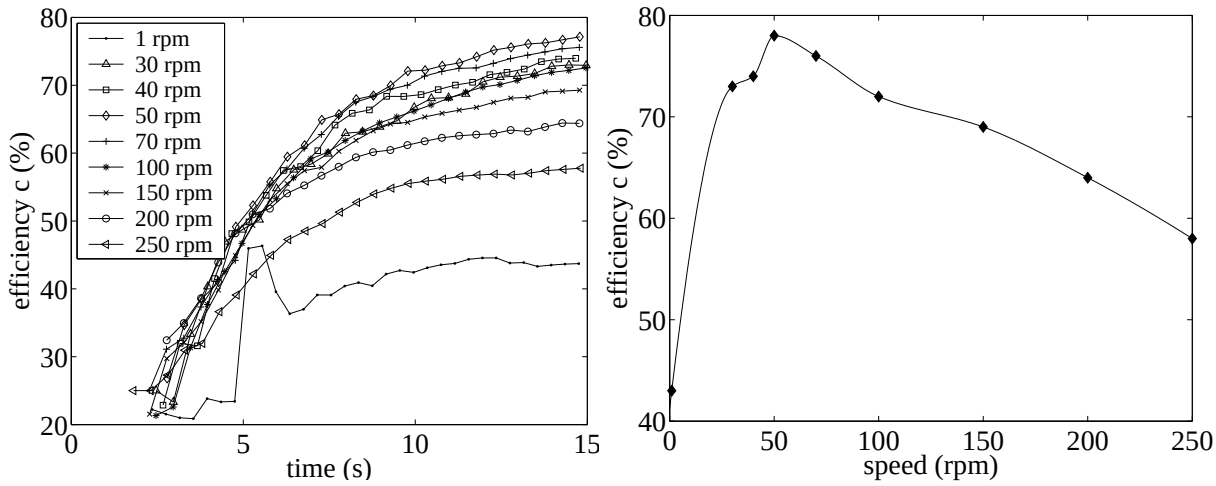


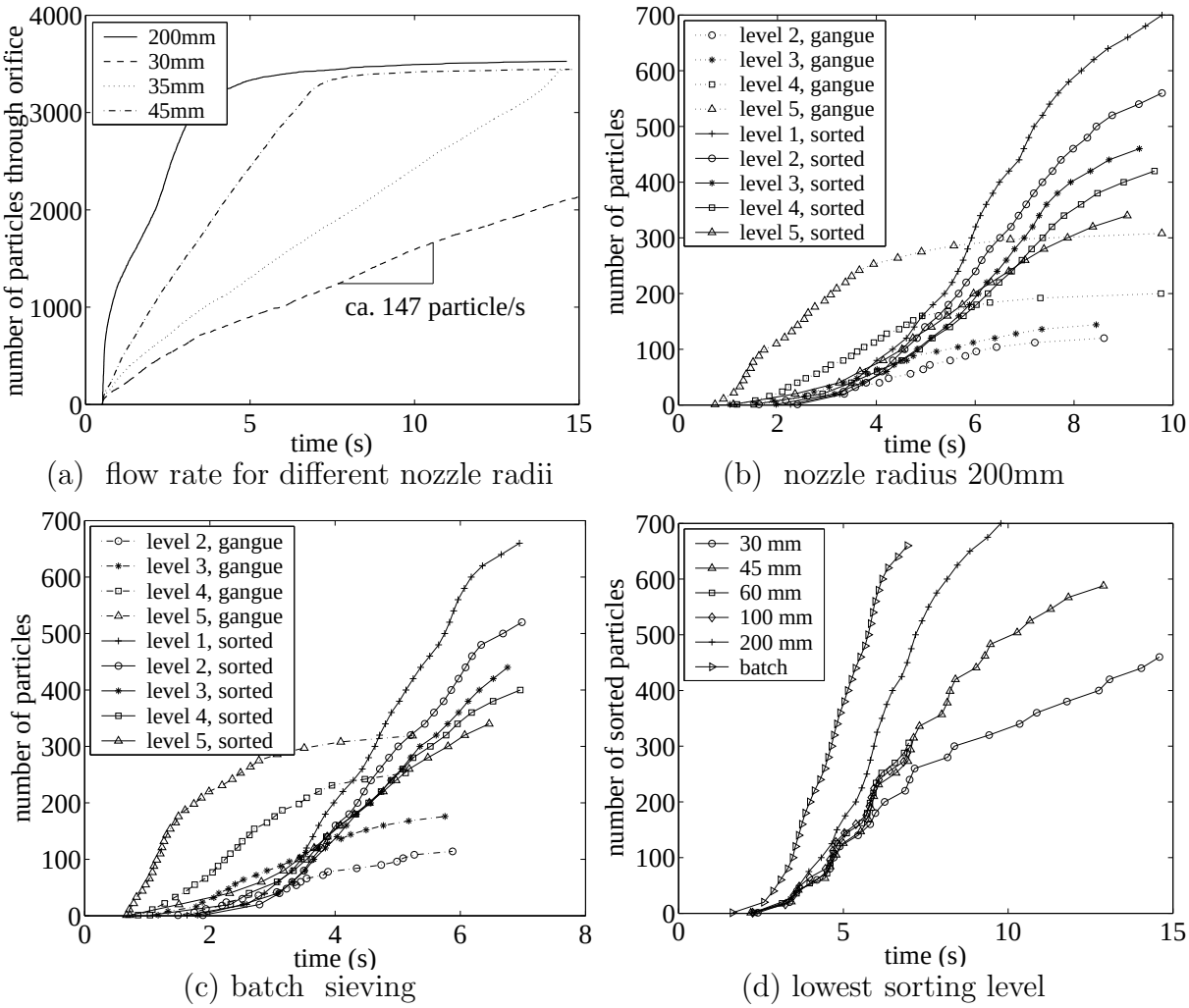
Figure 4.16: Efficiency of TSM machine for different speeds in the continuous screening operation mode, $\alpha = 1°$, $\beta = 0.65°$, $w/g = 1$, $b = 0.8w$ and and feeding rate 147 particles per second.

4.4.2 Influence of the feeding rate

We next study the effect of the feeding rate of the mixed particulate material on the machine efficiency. Increasing the size of the output nozzle of the feeding container will increase the feeding rate of the particle flow. Since the number of particles in the feeding container is limited, the feeding rate of the particles through a relatively large orifice will not be kept constant through the whole simulation process. In this case, most of the particles in the feeding container will fall down continuously and in a constant rate just only in the beginning of the simulation, see Fig. 4.17a. For small orifice size, e.g. 30mm, the flow rate will stay constant at 147 particle/s over the whole simulation since there is enough material in the feeding cylinder which keeps the particles flow through the feeding nozzle constant after a few seconds. This constant flow rate is used in our simulation study for continuous screening operations.

Continuous screening and batch sieving are studied. In continuous screening, it is observed that low feeding rates with nozzle radius of 30mm will lead to a low screening performance. There, the sorted particles travel slowly to the machine exits together with some of the undesired gangue ones, see Fig. 4.15b. The rate of the sorted particles is observed to improve with increasing feeding rate. Opening the nozzle output further will increase the feeding rate of the particle flow and improve the screening efficiency of the machine during the constant rate feeding period, see Fig. 4.17b.

In case of batch sieving, 3483 particles have been tested. The machine was started running



(a) flow rate for different nozzle radii (b) nozzle radius 200mm

(c) batch sieving (d) lowest sorting level

Figure 4.17: Influence of the feeding rate on the sorted number of particles at different sorting levels of TSM machine in batch sieving and continuous screening operation modes, $\alpha = 1°$, $\beta = 0.65°$, $w/g = 1$, $b = 0.8w$ and $\omega = 50$rpm.

with a big number of particles in the very beginning of the operating process. Those particles are fed immediately inside the machine barrel. Due to these heavy packed particles and the partially blocked mesh, it is observed that the gangue particles will rapidly accumulate in the beginning of the simulation. Many collisions between different particles will happen and force many of these particles to travel rapidly through the machine exits without even touching the mesh. It is not surprising that one of the most dangerous situations is blocking of the mesh and one has to ensure that the feeding rate is not so high that this can happen. As the time proceeds, the rate of the gangue particles will decrease to be steady after $5s$ simulation time, see Fig. 4.17c. Fewer gangue particles will then travel outside the machine exits since most of the particles are already separated at the different levels inside the machine. In the meantime, only the right-sized-particles will be sorted and directly run away through the machine exits, which improves the performance of the separation process.

As an indication of the influence of the feeding rates on the machine efficiency, the number of sorted particles in the lowest level of the sifting unit is recorded, see Fig. 4.17d. The lowest-level-particles are usually opposed to many collisions and face different obstacles during their way from the top until they reach their final destination in the lowest deck of the machine. It is observed that very low feeding rates are not recommended in the case of continuous screening, see Fig. 4.18. Increasing the rate of the feeding material will increase the machine efficiency. Although batch sieving might reveal better sorting efficiency compared to the continuous screening as long as no blocking occurs, it is not commonly used in industrial processes and screening technology since most processes have to run without interruption for a long time.

There exist detailed investigations about practical feeding rates for certain nozzle openings, see e.g. [13]. Since this depends highly on the used material and many more aspects, it would require detailed investigations to carefully compare this by our simplified simulation model.

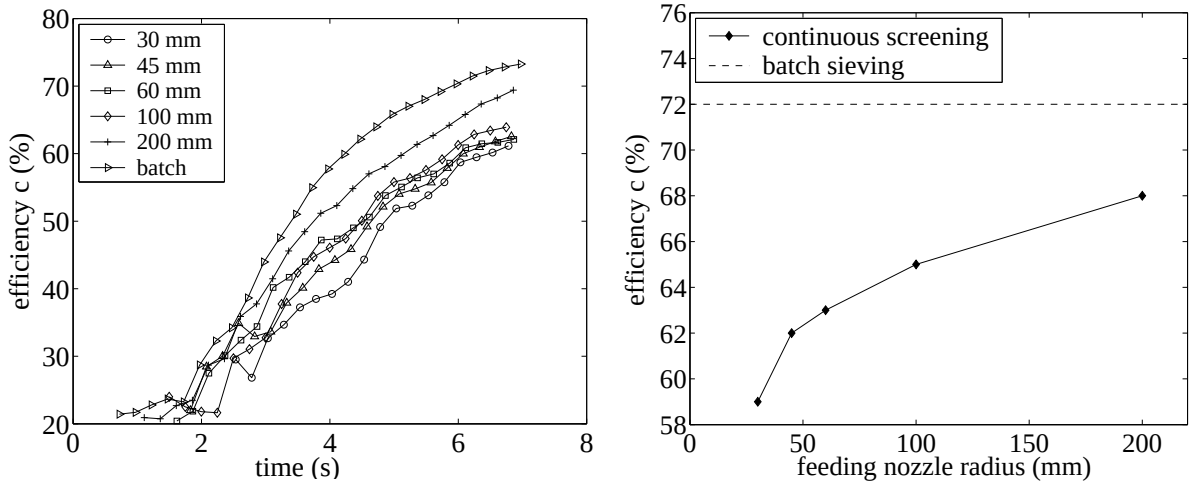


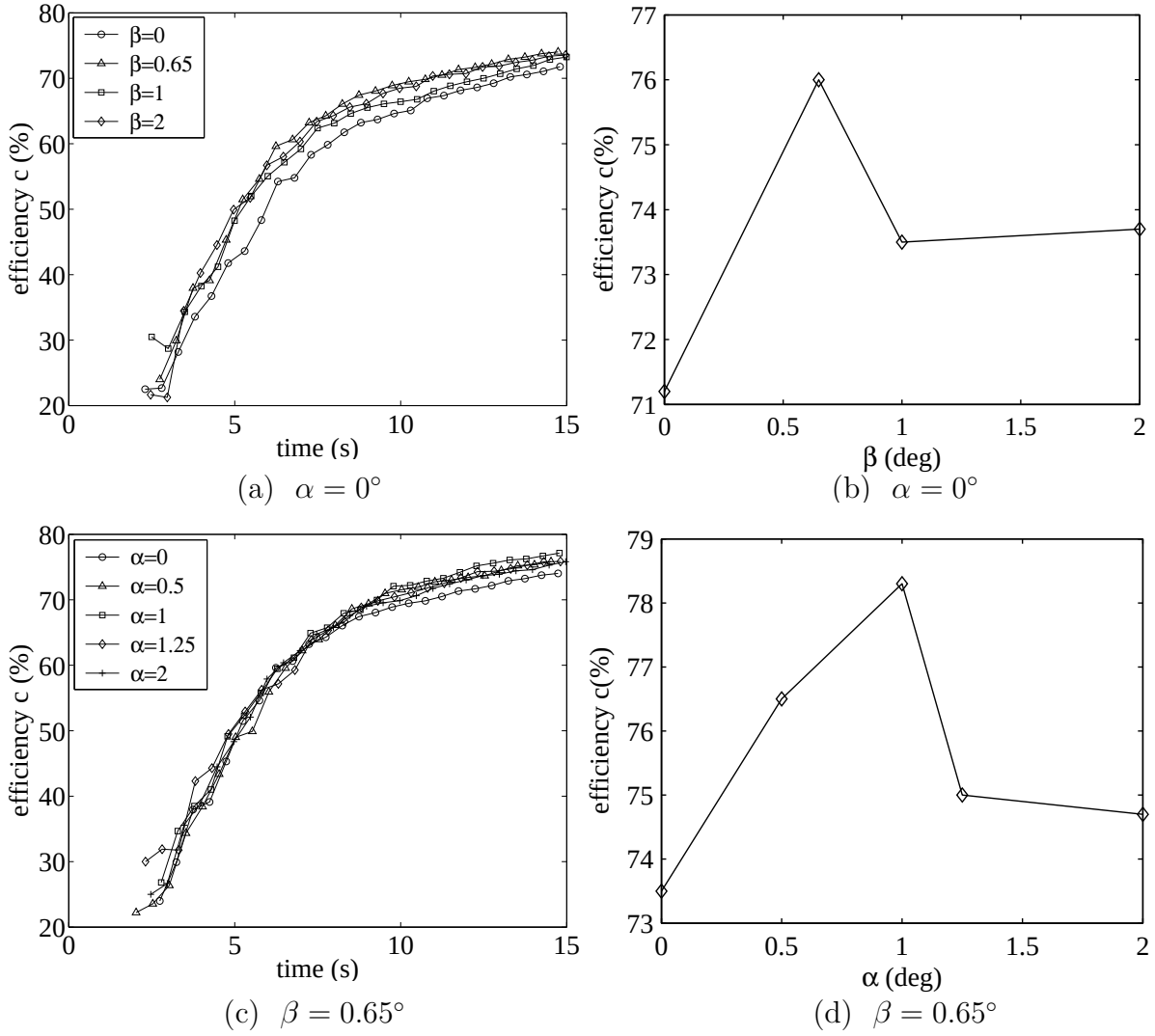
Figure 4.18: Efficiency of TSM machine for different openings of the feeding nozzle in batch sieving and continuous screening operation modes, $\alpha = 1°$, $\beta = 0.65°$, $w/g = 1$, $b = 0.8w$ and $\omega = 50$rpm.

4.4.3 Influence of the inclination angles

The inclination angles α and β of the sifting unit have a great influence on the machine performance. To study their effect, we fix one of them and change the other. It is observed that the steady state of machine efficiency will be reached after 13 s of simulation time, see Fig. 4.19a, c. For an inclination condition of $\alpha = 0°$, it is noticed that increasing the angle β will improve the machine performance. The maximum efficiency of $c = 76\%$ will be obtained when $\beta = 0.65°$, see Fig. 4.19b.

The inclination angles should be selected to achieve the best separation rates and the

maximum performance of the machine. For continuous screening of 147 *particles/s*, speed of 50rpm and our setup it was followed that $\alpha = 1°$ and $\beta = 0.65°$ are the optimal values for a maximum efficiency of $c = 78.2\%$ of the TSM machine, see Fig. 4.19d.



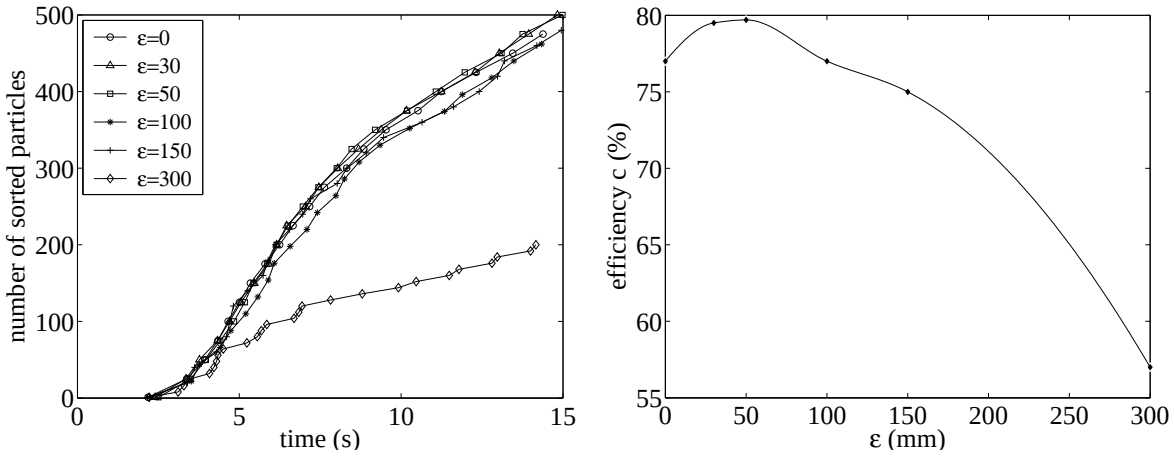
(a) $\alpha = 0°$ (b) $\alpha = 0°$

(c) $\beta = 0.65°$ (d) $\beta = 0.65°$

Figure 4.19: Efficiency of TSM machine for different inclination angles α and β of the sifting unit, $w/g = 1$, $b = 0.8w$, $\omega = 50$rpm and feeding rate 147 particles per second.

4.4.4 Influence of the shaft eccentricity

The eccentricity between the main vertical and the inclined holder shafts of the machine has a clear influence on the efficiency of the machine. In order to get a quantitative idea about this influence, the number of sorted particles in the lowest sorting level of the machine is recorded. It is found that small eccentricities will often be better than to work with large shaft eccentricities. High range eccentricities of $\epsilon = 300$mm (which are technologically infeasible) show a small number of 200 sorted particles in comparable with the



(a) lowest sorting level (b) overall efficiency

Figure 4.20: Influence of the shaft eccentricity on the sorted number of particles and on the machine efficiency, $\alpha = 1°$, $\beta = 1°$, $w/g = 1$, $b = 0.8w$, $\omega = 50$rpm and feeding rate 147 particles per second.

number of about 500 particles obtained using small range of eccentricities, see Fig. 4.20a. It is clear that a large eccentricities reveal bad performance. Small eccentricities less than $\epsilon = 100$mm appear good performance behaviour where $\epsilon = 40$mm leads to a maximum screening performance of about 80%, see Fig. 4.20b.

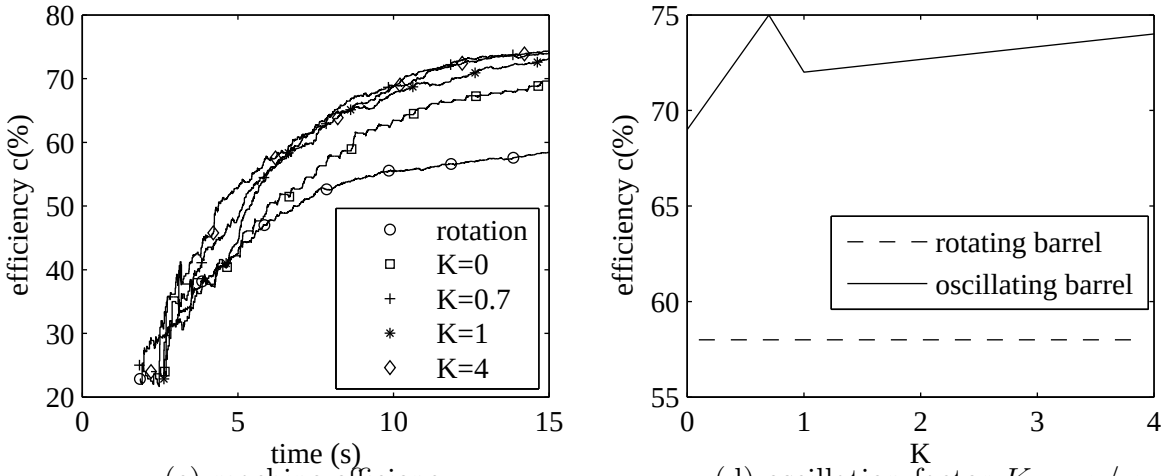
4.4.5 Influence of the barrel oscillation

The vibratory motion of the machine cylinder is also investigated. As the machine tumbles and rotates with ω around its main axis, the sorting process is measured while the barrel can move in three possible cases, i.e. rotating around its axis with ω_b, oscillating back and forth around its axis with different magnitudes of oscillation factor $K = \omega_b/\omega$ and blocking case without any rotation $\omega_b = 0$.

In this five-deck example the barrel is allowed to rotate with the same speed of the machine, i.e. $\omega_b = \omega = 150$rpm. For this case the sorting efficiency is measured to be relatively low, see Fig. 4.21. The continuous motion of the barrel will give the particles more energy and let them not damp easily. A low efficiency about $c = 58\%$ is recorded for this case during a continuous screening process.

On the other hand, a significant improvement is observed when the barrel is prevented to rotate around its individual axis of rotation. In this case of $\omega_b = 0$ or $K = 0$, the efficiency increases to about $c = 69\%$ due to the tumbling action of the machine, see Fig. 4.21b. Going further by allowing the sifting unit to oscillate around its axis, the results are better. Increasing the oscillation factor to about $K = 0.7$ for the applied triangular-mode oscillation described in Section 4.3.2 will obtain a maximum optimal efficiency of $c = 75\%$, see Fig. 4.21b. The medium oscillatory motion proves to be useful

for improving the separation process of particles. This improvement is due to the chance of little shaking that the particles can be opposed during the process. This shaking motion helps the particles to fit in a proper way over the holes of the sieves and to run to the exists of the machine.



(c) machine efficiency (d) oscillation factor $K = \omega_b/\omega$

Figure 4.21: Influence of the barrel oscillation on the machine efficiency, $\alpha = 1°$, $\beta = 0.65°$, $\zeta = 0.6$, $w/g = 1$, $\omega = 150$rpm, oscillation period 0.02 second and the feeding rate 147 particles per second.

For further investigations, another example of 775 glass particles has been simulated. These round particles consists of two different ball-sizes with almost equal numbers but different radii as: (389, 1.5mm) and (386, 2mm). A uniform-radius TSM with $R = 30$cm with a single sieve of hole dimension $w = 1.575$mm and $g = 0.8$mm. In this example the efficiency of the two-deck machine is measured depending on the mass of particles and not on their numbers as it was in the first example. Referring to Equation (4.36), the machine efficiency can be calculated as

$$c = \frac{100}{2}\sum_{i=1}^{2} c_i = 50\sum_{i=1}^{2}\frac{m_{s_i}}{m_{s_i}+m_{g_i}} = 50\left(1+\frac{m_{s_2}}{m_{s_2}+m_{g_2}}\right)\ , \tag{4.39}$$

where m_{g_1}=0 in the lowest deck of the machine. The rate of the overall outlet particles from both layers is also measured as

$$\rho_u = \frac{m_{s_2}+m_{g_2}}{t} \quad \text{and} \quad \rho_l = \frac{m_{s_1}}{t}\ , \tag{4.40}$$

where ρ_u and ρ_l are the rate of the overall outlet particles of the upper and lower layers, respectively. The percentage G of how much undesired gangue particles are being among the sorted ones, can be also determined by

$$G = 100\left(\frac{m_{g_2}}{m_{s_2}+m_{g_2}}\right)\ . \tag{4.41}$$

Depending the mass-based evaluation in measuring the machine performance will be adopted in the following subsections.

4.4.6 Influence of the surface friction coefficient

The roughness of the mesh surface a noticeable influence on the sorting operation. The friction between the particles and the sieve affects the motion of the particle through the driving tangential force acts on it. The influence of the degree of roughness is studied in this problem by changing the friction coefficient μ_d of Equation (4.30) and recording the corresponding machine efficiencies and sorting rates, see Fig. 4.22.

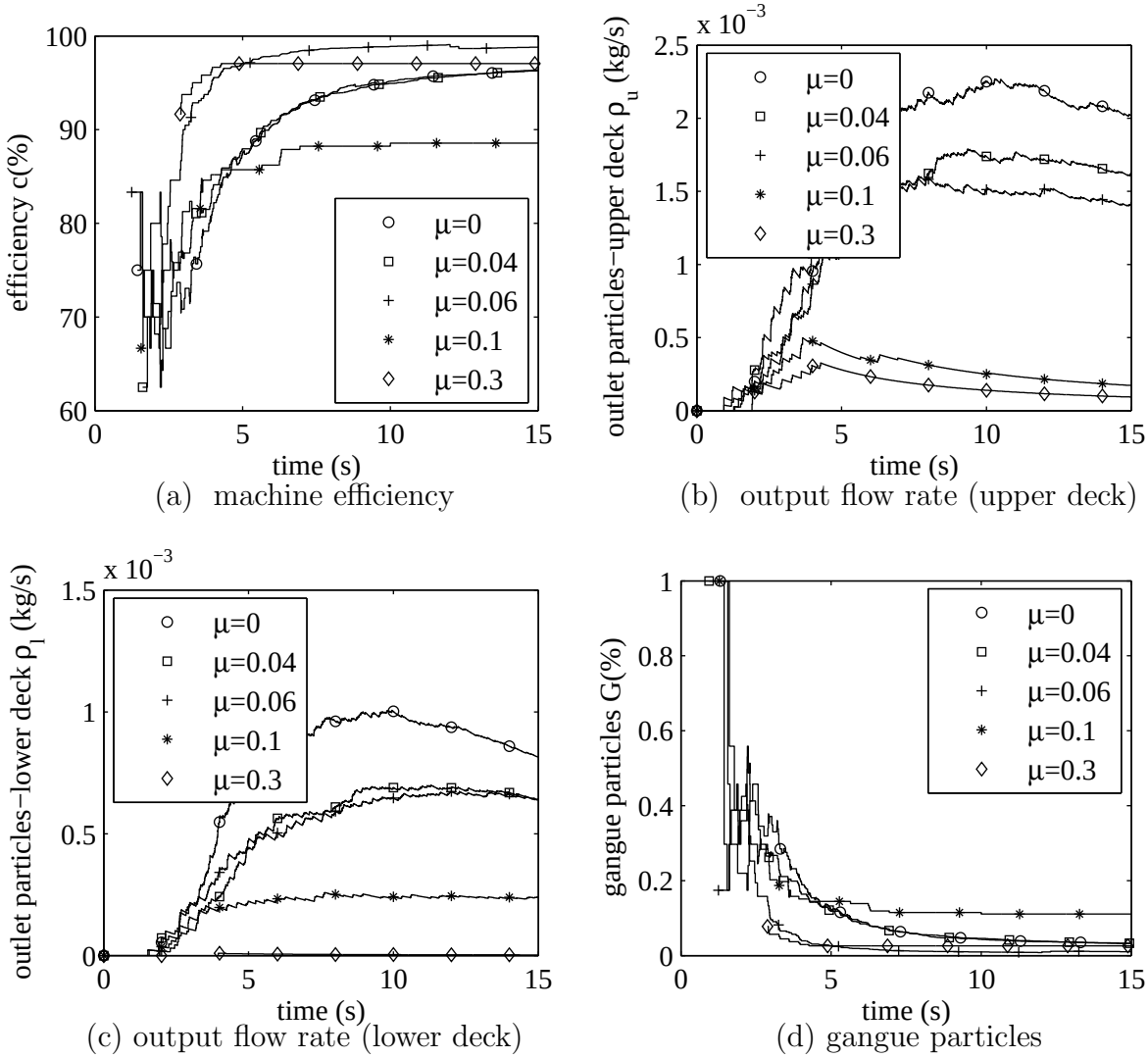
The frictional force plays a significant role in determining the particles movement. With no doubt, a high friction coefficients will hold the particles over the sieve inside the machine while low ones will increase the smoothness of the sieve surface and give more freedom to the particles to move with the help of the tumbling action away from the center towards the machine periphery. This fact is shown by the results in Fig. 4.22b,c. For a high friction coefficient of $\mu = 0.3$ the rate of the outlet particles is zero or almost zero from the upper and lower decks of the machine. The smaller the friction coefficient the higher the output rates.

On the other hand, decreasing the friction coefficient to a minimum is not always the solution for increasing the machine performance. The reason is that both the good and the bad particles will have the same chance to be driven towards the outlets. This appears in Fig. 4.22d. For e.g. $\mu = 0$, the percentage of the undesired gangue particles is higher than for $\mu = 0.04$ or $\mu = 0.06$. An optimal value of μ should exist to balance between a high sorting efficiency c, maximum output rates ρ and minimum percentage of gangue particles G. According to Fig. 4.22, this value is close to $\mu = 0.06$ for this example. The value which is used in simulations for the friction coefficient has to match the reality. Therefore, some special material and smoothness tests should be performed in order to get this kind of information about the friction coefficients.

4.4.7 Influence of the system size

To study the influence of the number of particles N on the performance of the screening operation and on the requested computation time, three system sizes are considered. The system sizes of 775, 4950 and 9900 glass balls of two equal particle groups of diameters 3mm and 4mm are simulated. A uniform machine radius of $R = 30$cm with a single sieve of hole dimension $w = 1.575$mm and $g = 0.8$mm. A continuous feeding process of a constant feeding rate of 357 particles per second is applied, see Fig. 4.23a. The machine barrel tumbles and rotates around the main shaft with $\omega = 180$rpm but is constrained to rotate around its own axis.

Increasing the number of particles of the system will make the machine need more time to reach to its steady state, see Fig. 4.23b. This state is much more interesting and essential to be studied and investigated than the transient state. Since the number of particles



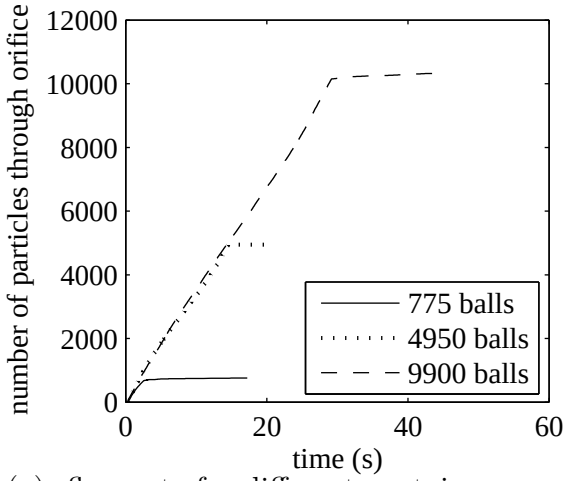
(a) machine efficiency (b) output flow rate (upper deck)

(c) output flow rate (lower deck) (d) gangue particles

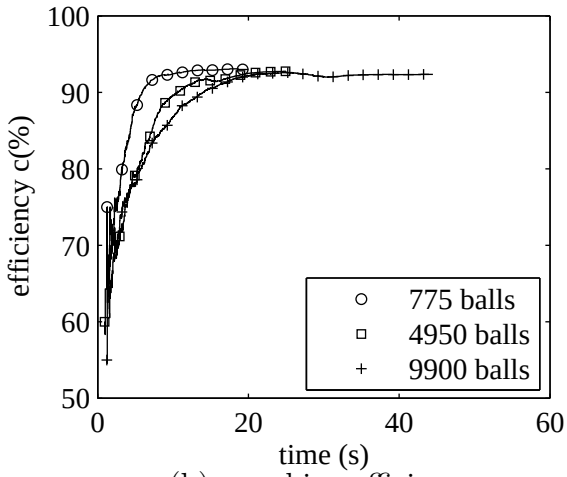
Figure 4.22: Influence of the mesh roughness of a non-rotating barrel on the sorted number of particles, $\alpha = 0.6°$, $\beta = 0.5°$, $\zeta = 0$, $w/g = 1.97$ and $\omega = 180$rpm.

used in simulation is not as much as that in the real experiments, the transient state can be measured more easily in simulation. On the contrast, this state is difficult to be measured in the field during the very beginning of the running operation. The vice versa is correct, the steady state can be observed well and measured in reality compared with the short-range simulation processes.

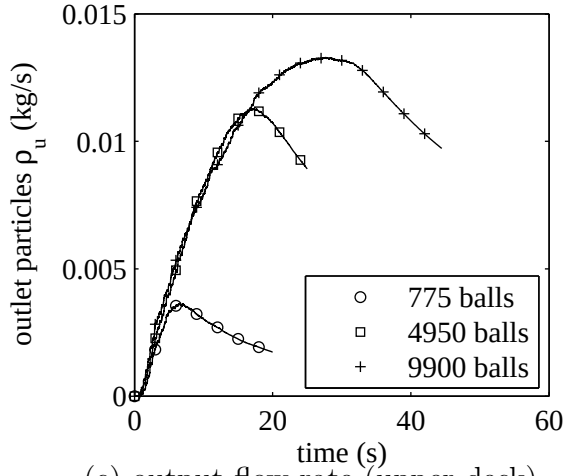
In order to reach to the steady state behavior in simulating high number of particles in a continuous screening operation, the simulation time have to be extended as long as possible to allow the particles in the feeding vessel to fall inside the machine, and therefore the computations will be then quite expensive. As shown in Fig. 4.23f, a simulation of 775 particles for 20s needs about 2.5h to be simulated. Increasing the system size to 4950 particles in the feeding vessel for 25s will increase the computation time to 36h. Going further by enlarging the system size to 9900 particles and accordingly the simulation time to 45s, the computations needs 153h, i.e. almost one week, to be done. Increasing the



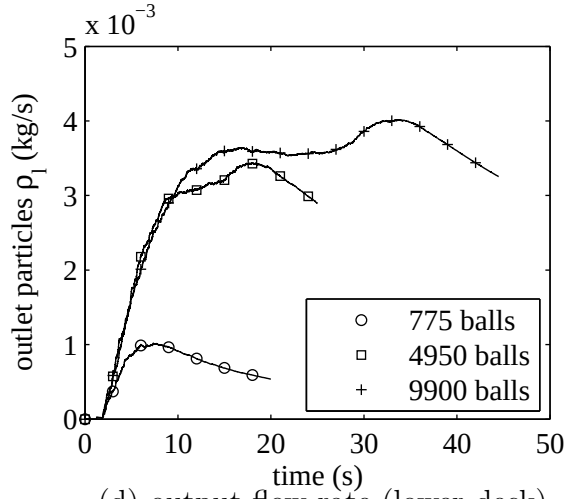
(a) flow rate for different container capacity



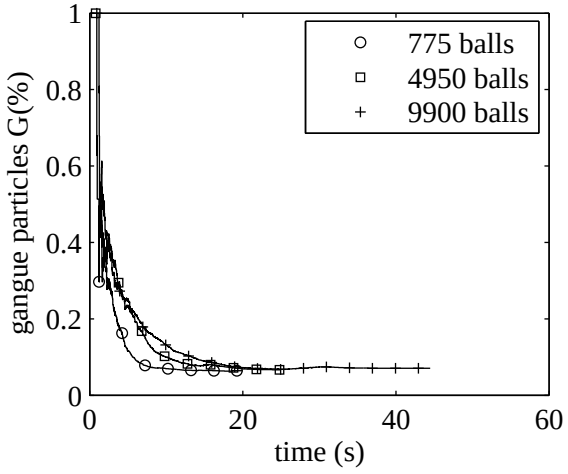
(b) machine efficiency



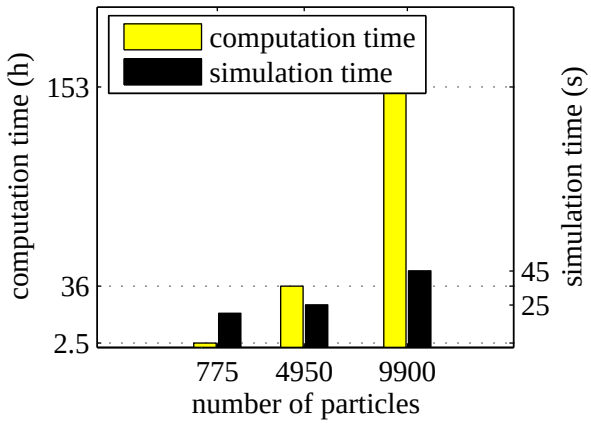
(c) output flow rate (upper deck)



(d) output flow rate (lower deck)



(e) gangue particles



(f) computation/simulation time

Figure 4.23: Curve extrapolation of the number of particles, $\alpha = 0.6°$, $\beta = 0.5°$, $w/g = 1.97$, $\zeta = 0.6$ and $\omega = 180$rpm.

number of particles by enlarging the system size N will often scale up the output-rates curves in a nonlinear manner, see Fig. 4.23c,d. Increasing the number of particles more to reach that numbers of hundreds of millions as it is in the real machine, will probably extrapolate these curves in order to match those obtained from the real experiments. But, unfortunately, this numbers could be very difficult or even impossible to be simulated sequentially on a single computer especially if these simulations have to be rerun frequently for many times when looking to do a parametric study of the machine variables. Therefore, and since screening operation is a dynamic process, some effort on spatial decomposition with movable-boundary domains can be paid to parallelize the problem and therefore reduce the corresponding computation time of the simulations.

Chapter 5

Conclusions and Closing Remarks

The behavior of granular media is investigated in this thesis. Extending existing algorithms, parallelizing the molecular serial codes and supporting this with studying and analyzing a physical industrial application of particle screening technology using a tumbling screening machine under various operational parameters and screen configurations are illustrated in this study. The goal is achieved by having a good overview of the operational and dynamical parameters which affect the TSM efficiency and the separation process. Reaching to different scalability factors by speeding up the simulations using spatial decomposition approach is also achieved.

For this purpose, Chapter 1 briefly introduced some contact problems in granular media with some computational procedures used in sequential and parallel computations. A general overview of some studies and the influence parameters which affect the particle screening technology were presented in this introductory chapter.

Going further, Chapter 2 concentrated on clarifying the basics of the granular media and some frequently-used algorithms and models, e.g. the discrete element method and the penalty approach for contact detection and force computations. This chapter introduced dealing with particle-to-particle and particle-to-wall contacts under different conditions of damping, friction and adhesion. The searching algorithms and neighborhood computations were disscussed and compared. It was found that here the Linked Linear List approach is more efficient than the Verlet approach. Different integration approaches was also discussed. It was found that Verlet integrators are efficient, accurate and appropriate to solve the equations of motion of the granular systems.

No doubt that one of the main obstacles for granular media computations is the long computation time of simulations, especially for large-sized system of particles. Chapter 3 introduced some parallel strategies and concentrated on the spatial decomposition approach using message-passing libraries for solving the parallel task. This method allows scalability and good results especially when load balancing is done.

In this chapter, the efficiency of computations is observed to be kept within normal ranges and below the ideal behavior when different processes are run in parallel on one machine. On the other hand, a superlinear behavior is recorded when using different computers with many processors due to the individual cache memory effect of each of the machines used in the network. This chapter also underlined the relation between the speedup and the size of the system. It was found that the performance improves with increasing the number of particles. The reason is due to the communication and data flow which become more efficient between the different tasks as the number of particles increases and therefore, the communication cost will directly decrease and accordingly, the computational speedup will increase.

In Chapter 4, as a practical industrial application, the particle screening phenomenon over a rotary tumbling screen is studied. The discrete element method is applied and used as a simulation tool for the separation process. This method proves its ability to be a powerful numerical modeling tool for solving problems in granular media. In an attempt to better understand the mechanism of the particle transport between the different layers of the sifting system, different computational studies for achieving optimal operation have been performed.

As a result of collisions, the particles dissipate kinetic energy due to the normal and frictional contact losses. The particle distribution, sifting rate of the separated particles and the efficiency of the segregation process have been studied. For specific geometrical and contact parameters particle transportation, sifting rates and machine efficiency are recorded. Particles are simulated in uniform and stepped models of tumbling cylinders. For both continuous screening and batch sieving, it was found that the segregation process is very sensitive to the rotational speed of the machine. This speed should be selected to be within certain limits to maximize the number of sorted particles and to improve the sifting rates for the different machine levels. Too high and too low speeds will lead to a bad screening performance.

Furthermore, the particle feeding rates, inclination angles and shaft eccentricity have a great influence on the machine efficiency. Small angles between 0.5° to 1° and eccentricities between 25 to 50mm are recommended. The sieve roughness has also an influence on the number of particles that stay or leave the machine. An optimal value of relatively medium friction coefficient is recommended. Moreover, the barrel oscillation has a significant influence on the sorting process. Oscillatory motion of the barrel showed better performance relative to the non-rotating or even continuous-rotating motion. For the same operating conditions, batch sieving shows better results compared to continuous screening.

Improving the accuracy of the simulation requires to be more realistic in implementing the contact forces and the associated contact parameters of the dynamical system of the granular system. Physical contacts inside the TSM require some more detailed investigations.

These parameters can be obtained from special experiments. For better understanding of the particle sorting and transportation between the different layers of the machine, experimental studies have to be performed.

For future investigations of this work the following points may be mentioned:

▶ Regarding to the presented approach in Chapter 2, different particle shapes, e.g. polygonal, elliptical, etc. can be modelled and analyzed under various loading conditions of damping, friction and adhesion.

▶ The spatial decomposition method which was used in Chapter 3 can be extended from the strip-pattern model to the spatial-cubic model involving more number of particles. Comparing this method with other parallel techniques is quite recommendable.

▶ For future modifications of the presented approach in modelling the tumbling screening machine in Chapter 4, following points are noticeable:

• Using different shapes of particulate materials rather than spherical balls together with different mesh configurations.

• Extending the operational running time of the machine by screening much more particles in continuous and batch screening operations. This will need to run the simulations in parallel environment for time consuming purposes.

• Using much more realistic and precise approaches to check the contacts between balls and sieve wires should be used rather than depending only on the probabilistic approach. Designing the deflector in a proper way will help the particles to be guided much close to that in reality.

• Performing real experiments to observe and measure the transient and steady state modes of the screening operation and use them for comparison purposes.

Appendix

A.1 Computational details- input and output files

In this section a general overview of the input and output files of the program MOLDYN is given and described, see also [71].

A.1.1 Input files

The input files should be constructed in a proper way that they contain all necessary information required to describe the whole system of particles. The three-used input files, i.e. the control-input file, the particles-input file and the walls-input file are briefly described.

▶ **control-input file:** This file describes the control variables which are necessary for formulas and mathematical calculations in the program. This file has the name extension (*.cin) and looks like

```
times.t0            0.00
times.tend          10.0
times.dt            1.0e-5
control.grav       -9.81
control.rho         20.0
control.mue_s       0.45
control.mue_d       0.45
sd.stiff_lo         7.25e3
sd.stiff_k_un       7.25e3
sd.stiff_adh       -3.00e2
sd.damp_particle    0.00
sd.damp_wall        0.38
sd.e                0.27
```

In this file the start and end simulation times along with the simulation time step are defined. The other variables represents the gravity acceleration, the mass density, the

static and dynamic friction coefficients, the loading/unloading and adhesive stiffness coefficients, the particle-particle and particle-wall damping coefficients and finally the normal coefficient of restitution.

▶ **particle-input file:** This file contains the initial locations, velocities, orientations and angular velocities of the particles. It has the name extension (*.pin) and looks like

```
3257
-1.7  0.0  2.2  0.0  1.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.13  1
 1.5  0.3  1.4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.15  0
-2.6  0.4  2.1  0.0  1.0  0.1  0.0  0.0  0.0  0.0  0.0  0.0  0.15  1
 3.5  0.2  2 0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.17  0
 3.5  0.2  4.1  2.4  1.0  1.2  0.0  0.0  0.0  0.0  0.0  0.0  0.13  1
 4.2  0.3  2.8  0.6  0.0  0.8  0.0  0.0  0.0  0.0  0.0  0.0  0.15  1
...
```

In this file the total number of particles appears in the very beginning. Columns from 1 to 3 represent the location of the particle in the three coordinates x,y and z. Columns from 4 to 6 denote the linear velocity of the particle. Columns from 7 to 12 represent the angular position and angular velocity of the particles. The radii of the particles are located one column before the last one. The state of motion of the particles during simulation appears as a flag in the last column, i.e. 1 for the movable particle while 0 for the stationary one.

▶ **wall-input file:** The location of the walls and the direction of their positive normals are identified in this file which has the extension (*.win) and appears as

```
 6  0.0030
 1.0000  0.0000  0.0000 -0.0050   0.0855   0.0855
-1.0000  0.0000  0.0000  0.1715   0.0855   0.0855
 0.0000  1.0000  0.0000  0.0855  -0.0005   0.0855
 0.0000 -1.0000  0.0000  0.0855   0.1715   0.0855
 0.0000  0.0000  1.0000  0.0855   0.0855  -0.0005
 0.0000  0.0000 -1.0000  0.0855   0.0855   0.1715
```

This file represents a box of 6 walls with a wall thickness of 3 mm. The first three columns are the normal components in the direction of the three coordinate axes, while the last three are the coordinates of the center points of the wall, see Fig. 2.7b. The walls can also be defined by their end points from which the normals can be computed.

A.1.2 Output files

The results of simulations are stored in output files. These files contains data about coordinates and velocities of the trajectories at selected predefined output times. These

files should be written in a certain format to be read from the particle animation tools, e.g. ANIM for 3D [12] and XBALLS for 2D [60]. In case of ANIM, the file extension is usually used as (*.str) while as (*.xb) in case of XBALLS. The (*.xb) output file looks like

```
3257 0.030 0.0 0.0 0.1710 0.1710
0.050000 0.120000  0.000000 -0.000098 0.004000 0.000000
0.070000 0.070000  0.000000 -0.000098 0.004000 0.000000
0.099996 0.004000  0.000000 -0.000098 0.004000 0.000000
...
3257 0.430 0.0 0.0 0.1710 0.1710
0.29056 0.119508  0.000000 -0.098296 0.004000 0.000000
0.40440 0.069402  0.000000 -0.088594 0.004000 0.000000
...
3257 1.650 0.0 0.0 0.1710 0.1710
0.330045 0.616286  0.000000 -0.495609 0.004000 0.000000
0.763331 0.679008  0.000000  0.208233 0.004000 0.000000
...
```

The file shows a system of 3257 particles with their locations and velocities for three output time steps, i.e. at $t = 0.03$s, $t = 0.43$s and $t = 1.65$. The components of the particle location are printed in the first three columns, while the particle velocity components appear in the next three columns. Information about the wall dimensions is also appeared in the first line of the output file. Furthermore, the translation vector and the rotation matrix of all particles and walls are added to the ANIM output (*.str) file while other data about particle locations are stored similar to the XBALLS output file. The (*.str) output file appears as

```
0.01 999 12
-0.078490 0 0.996915 0 1 0 -0.996915 0 -0.078490  1.987678 0.000000 0.156497
-0.233537 0 0.972348 0 1 0 -0.972348 0 -0.233537  1.938696 0.000000 0.465635
...
1 0 0  0 1 0  0 0 1 -0.981598 0.000000 0.830000
1 0 0  0 1 0  0 0 1 -0.961598 0.000000 0.830000
...
```

In this file the first nine columns represent the nine elements of the rotation matrix while the last three columns are the elements of the particle location represented in the translation vector.

Bibliography

[1] Alexander, A.; Shinbrot, T.; Muzzio, F.: Granular Segregation in the Double-Cone Blender: Transitions and Mechanisms, *Physics of Fluids*, Vol. 13, No. 3, pp. 578-587, 2001.

[2] Alkhaldi, H.; Eberhard, P.: Computation of Screening Phenomena in a Vertical Tumbling Cylinder, *Proceedings in Applied Mathematics and Mechanics (PAMM)*, Berlin, Germany, 2006, submitted for publication.

[3] Alkhaldi, H.; Eberhard, P.: Particle Screening Phenomena in an Oblique Multi-Level Tumbling Reservoir - A Numerical Study Using Discrete Element Simulation, *Granular Matter*, 2007, accepted for publication.

[4] Alkhaldi, H.; Eberhard, P.: Segregation of Particulate Material Using the Discrete Element Method, *Proceedings in Symposium on Computational Contact Mechanics (IUTAM)*, Hannover, Germany, 2006, submitted for publication.

[5] Allen, M.; Tildesley, D.: *Computer Simulation of Liquids*, Oxford: Clarendon, 1987.

[6] Andrew R. Leach: *Molecular Modelling: Principles and Applications*, Reading: Addison-Wesley Longman, 1996.

[7] Aversa, R.; Martino, B.; Mazzocca, N.; Venticinque, S.: A Hierarchical Distributed-Shared Memory Parallel Branch & Bound Application with PVM and OpenMP for Multiprocessor Clusters, *Parallel Computing*, Vol. 31, pp. 1034-1047, 2005.

[8] Barnes, J.: A Modified Tree Code: Don't Laugh, It Runs, *Journal of Computational Physics*, Vol. 87, No. 1, pp. 161-170, 1990.

[9] Barnes, J.; Hut, P.: A Hierarchical $O(NlogN)$ Force Calculation Algorithm, *Nature*, Vol. 324, No. 4, pp. 446-449, 1986.

[10] Barney, B.: Introduction to Parallel Computing,
see: *http://www.llnl.gov/computing/tutorials/parallel_comp*, (July 2006).

[11] Beazley, D.; Lomdahl, P.: Message-Passing Multi-Cell Molecular Dynamics on the Connection Machine 5, *Parallel Computing*, Vol. 20, pp. 173-195, 1994.

[12] Bestle, D.; Eberhard, P.: *NEWOPT/ANIM 2.2 - Ein Programmpaket zur Analyse und Optimierung von mechanischen Systemen (in German)*, Institut B für Mechanik, Universität Stuttgart, AN-35, 1994.

[13] Beverloo, W.; Leniger, H.; Velde, J.: The Flow of Granular Solids through Orifices, *Chemical Engineering Science*, Vol. 15, pp. 260-269, 1961.

[14] Blelloch, G.; Narlikar, G.: A Practical Comparison of N-Body Algorithms, *Parallel Algorithms, Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 30, pp. 1-16, 1997.

[15] Brooks, B.; Bruccoleri, R.; Olafson, B.; States, D.; Swaminathan, S.; Karplus, M.: CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, *Journal of Computational Chemistry*, Vol. 4, pp. 187-217, 1983.

[16] Buyya, R.: *High Performance Cluster Computing: Programming and Applications*, Englewood Cliffs: Prentice-Hall, 1999.

[17] Coulson, J.; Richardson, J.: *Chemical Engineering Volume 2, Particle Technology and Separation Process*, 4th edition, Oxford: Pergamon, 1991.

[18] Cundall, P.: A Computer Model For Simulating Progressive Large Scale Movements in Block Rock Systems, *Symposium of the International Society of Rock Mechanics*, Nancy, France, pp. 129-136, 1971.

[19] Cundall, P.; Hart, R.: Numerical Modelling of Discontinua, *Engineering Computation*, Vol. 9, pp. 101-113, 1992.

[20] Cundall, P.; Strack, O.: A Discrete Numerical Model for Granular Assemblies, *Geotechnique*, Vol. 29, pp. 47-65, 1979.

[21] de Gennes, P.: Granular Matter: A Tentative View, *Reviews of Modern Physics*, Vol. 71, pp. 374-382, 1999.

[22] Dudek, M.; Ponder, J.: Accurate Modeling of the Intramolecular Electrostatic Energy of Proteins, *Journal of Computational Chemistry*, Vol. 16, pp. 791-816, 1995.

[23] Dury, C.; Ristow, J.: Competition of Mixing and Segregation in Rotating Cylinders, *Physics of Fluids*, Vol. 11, No. 6, pp. 1387-1394, 1999.

[24] Eberhard, P.: *Kontaktuntersuchungen durch hybride Mehrkörpersystem/ Finite Elemente Simulationen* (in German), Aachen: Shaker Verlag, 2000.

[25] Eberhard, P.; Alkhaldi, H.: Efficient Computation of Colliding Particles in a Vertical Tumbling Sorting Machine, *Proceedings Second International Congress on Computational Mechanics and Simulation (ICCMS)*, Vol. 1, pp. 81-87, I.K. Publishing House, New Delhi, India, 2006.

[26] Eichinger, M.; Grubmüller, H.; Heller, H.; Tavan, P.: FAMUSAMM: An Algorithm for Rapid Evaluation of Electrostatic Interactions in Molecular Dynamics Simulations, *Journal of Computational Chemistry*, Vol. 18, pp. 1729-1749, 1997.

[27] Faber, V.; Lubeck, O.; White, A: Superlinear Speedup of an Efficient Sequential Algorithm is Not Possible, *Parallel Computing*, Vol. 3, pp. 259-260, 1986.

[28] Fincham, D.: Parallel Computers and Molecular Simulation, *Molecular Simulation*, Vol. 1, pp. 1-45, 1987.

[29] Finkel, R.; Fishburn, J.: Parallelism in Alpha-Beta Search, *Artificial Intelligence*, Vol. 19, pp. 89-106, 1982.

[30] Fleißner, F.; Eberhard, P.: Dynamical Particle Simulation with Parallel Cache-Aware Domain Decomposition Strategies, *PAMM Proceedings in Applied Mathematics and Mechanics*, Vol. 5, No. 1, Luxemburg, pp. 657-658, 2005.

[31] Flynn, M.: Very High-Speed Computing Systems, *Proceedings of the IEEE Publication*, New York, pp. 1901-1909, 1966.

[32] Foster, I.: *Designing and Building Parallel Programs*, Boston: Addison-Wesley Longman Publishing Corporation, 1995.

[33] Fox, G.; Johnson, M.; Lyzenga, G.; Otto, S.; Salmon, J.; Walker, D.: *Solving Problems on Concurrent Processors: General Techniques and Regular Problems*, Volume I, Englewood Cliffs: Prentice-Hall, 1988.

[34] Gao, G.: An Efficient Hybrid Dataflow Architecture Model, *Journal of Parallel and Distributed Computing*, Vol. 19, No. 4, pp. 293-307, 1993.

[35] Garcia-Rojo, R.; McNamara, S.; Herrmann, H.: Discrete Element Methods for the Micro-Mechanical Investigation of Granular Ratcheting, *European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS*, Jyväskylä, pp. 1-8, 2004.

[36] Geist, A.; Beguelin, A.; Dongarra, J.; Jiang, W.; Manchek, R.; Sunderam, V.: *PVM 3 User's Guide and Reference Manual*, Oak Ridge, Tennessee: Oak Ridge National Laboratory, 1994.

[37] Greengard, L.; Rokhlin, V.: A Fast Algorithm for Particle Simulation, *Journal of Computational Physics*, Vol. 73, pp. 325-348, 1987.

[38] Gupta, S.: Computing Aspects of Molecular Dynamics Simulation, *Computer Physics Communication*, Vol. 70, pp. 243-270, 1992.

[39] Gustafson, J.: Reevaluating Amdahl's Law, *Communications of the ACM*, Vol. 31, No. 5, pp. 532-533, 1988.

[40] Gustafson, J.; Montry, G.; Benner, R.: Development of Parallel Methods for a 1024-Processor Hypercube, *SIAM Journal on Scientific and Statistical Computing*, Vol. 9, No. 4, pp. 609-638, 1988.

[41] Heller, H.; Grubmüller, H.; Schulten, K.: Molecular Dynamics Simulation on a Parallel Computer, *Molecular Simulation*, Vol. 5, pp. 133-165, 1990.

[42] Helmbold, D.; McDowell, C.: Modeling Speedup(n) greater than n, *IEEE Transactions Parallel and Distributed Systems*, Vol. 1, No. 2, pp. 250-256, 1990.

[43] Herrmann, H.; Müller, M.: Simulations of granular media, *Conference Proceedings from the Workshop 'Molecular Dynamics on Parallel Computers'*, World Scientific, pp. 1-10, 1999,
see also: *http://www.hlrs.de/people/mueller/papers/parallelMD99/parallelMD.html*.

[44] Hertz, H.: Über die Berührung fester elastischer Körper (in German), *Journal für die reine und angewandte Mathematik*, Vol. 92, pp. 156-171, 1882.

[45] Jansen, M.; Glastonbury, J.: The Size Separation of Particles by Screening, *Powder Technology*, Vol. 1, pp. 334-343, 1967.

[46] Kaye, B.; Robb, N.: An Algorithm for Deducing an Effective Sieve Residue from the Rate of Powder Passage through a Sieve, *Powder Technology*, Vol. 24, pp. 125-128, 1979.

[47] Kelly, E.; Spottiswood, D.: *Introduction to Mineral Processing*, New York: Wiley-Interscience, 1999.

[48] Khakhar, D.; McCarthy, J.; Ottino, M.: Radial Segregation of Granular Mixtures in Rotating Cylinders, *Physics of Fluids*, Vol. 9, No. 12, pp. 3600-3614, 1997.

[49] Khulief, Y.; Shabana, A.: A Continuous Force Model for the Impact Analysis of Flexible Multibody Systems, *Mechanism and Machine Theory*, Vol. 22, pp. 213-224, 1987.

[50] Kreuzer, E.; Leister, G.: *Programmsystem NEWEUL (in German)*, Institut B für Mechanik, Universität Stuttgart, AN-32, 1991.

[51] Kuhl, E.; D'Addetta, G.; Herrmann, H.; Ramm, E.: A Comparison of Discrete Granular Material Models with Continuous Microplane Formulations, *Granular Matter*, Vol. 2, pp. 113-122, 2000.

[52] Lätzel, M.; Luding, S.; Herrmann, H.: From Discontinuous Models Towards a Continuum Description, *International Symposium on Continuous and Discontinuous Modelling of Cohesive Frictional Materials*, Springer Verlag, Berlin, pp. 215-230, 2001.

[53] Lankarani, H.; Nikravesh, P.: A Contact Force Model with Hysteresis Damping for Impact Analysis of Multibody Systems, *Journal of Mechanical Design*, Vol. 112, pp. 369-376, 1990.

[54] Lee, J.; Ladd, A.: Axial Segregation of a Settling Suspension in a Rotating Cylinder, *Physical Review Letters*, Vol. 95, No. 1-4, 2005.

[55] Li, J.; Webb, C.; Pandiella, S.; Campbell, G.: A Numerical Simulation of Separation of Crop Seeds by Screening-Effect of Particle Bed Depth, *Institution of Chemical Engineers IChemE Part C*, Vol. 80, pp. 109-117, 2002.

[56] Li, J.; Webb, C.; Pandiella, S.; Campbell, G.: Discrete Particle Motion on Sieves - A Numerical Study Using the DEM Simulation, *Powder Technology*, Vol. 133, pp. 190-202, 2003.

[57] Lindahl, E.; Hess, B.; van der Spoel, D.: GROMACS 3.0: A Package for Molecular Simulation and Trajectory, *Journal of Molecular Modeling*, Vol. 7, pp. 306-317, 2001.

[58] Luding, S.: Collisions and Contacts between two Particles, *Physics of Dry Granular Media*, E350-NATO ASI series, Dordrecht, Kluwer Academic Publishers, pp. 285-314, 1998.

[59] Luding, S.: The Micro-Macro Mechanics of Granular Materials, *GACM report 2*, pp. 22-28, 2003.

[60] Luding, S.: XBALLS: Animation Software,
see: *http://www.icp.uni-stuttgart.de/∼lui/*, (December 1994).

[61] Luding, S.; Clement, E.; Blumen, A.; Rajchenbach, J.; Duran, J.: Anomalous Energy Dissipation in Molecular Dynamics Simulations of Grains, *Physical Review E*, Vol. 50, pp. 4113-4122, 1994.

[62] Lyubartsev, A.; Laaksonen, A.: MDynaMix - A Scalable Portable Parallel MD Simulation Package for Arbitrary Molecular Mixtures, *Computer Physics Communications*, Vol. 128, pp. 565-589, 2000.

[63] Mao, K.; Xu, Z.; Wang, M.; Chen, T.: Efficient Computation of Particle Motions in Discrete Element Modeling of Particle Damping, *Eighth International Symposium on Plasticity and Impact Mechanics*, New Delhi, India, pp. 994-1005, 2003.

[64] Matthey, T.: *Framework Design, Parallelization and Force Computation in Molecular Dynamics*, Ph.D. thesis, Department of Informatics, University of Bergen, Norway, 2002.

[65] McLoughlin, G.; Fergusson, I.: High Performance Computers and Export Control Policy: Issues for Congress, *Report for Congress*, The Library of Congress, Washington, pp. 1-28, 2003.

[66] Menabrea, L.: Notions sur la Machine Analytique de M. Charles Babbage (in French), *Bibliothèque Universelle de Genève*, Vol. 41, pp. 352-376, 1842.

[67] Mishra, B.; Rajamani, R.: The Discrete Element Method for the Simulation of Ball Mills, *Applied Mathematical Modelling*, Vol. 16, pp. 598-604, 1992.

[68] Mishra, B.; Thornton, C.: An Improved Contact Model for Ball Mill Simulation by the Discrete Element Method, *Advanced Powder Technology*, Vol. 13, No. 1, pp. 25-41, 2002.

[69] Miya, E.: Suggestion on Superlinear Speed Up Terminology, *Network News Posting*, December 1988.

[70] Mou, G.; Hudak, P.: An Algebraic Model for Divide-and-Conquer Algorithms and Its Parallelism, *Journal of Supercomputing*, Vol. 2, pp. 257-278, 1988.

[71] Muth, B.: *Simulation von Kontaktvorgängen einfacher Körper mit Methoden der Molekulardynamik* (in German), DIPL-87, Institute B of Mechanics, University of Stuttgart, Germany, 2001.

[72] Muth, B.; Eberhard, P.; Luding, S.: Contact Simulation for Many Particles Considering Adhesion, *Mechanics Based Design of Structures and Machines*, Vol. 31, No. 3, pp. 433-457, 2003.

[73] Muth, B.; Müller, M.; Eberhard, P.; Luding, S.: Contacts Between Many Bodies, *Machine Dynamics Problems*, Vol. 28, No. 1, pp. 101-114, 2004.

[74] Nakagawa, M.: Axial Segregation of Granular Flows in a Horizontal Rotating Cylinder, *Chemical Engineering Science*, Vol. 49, No. 15, pp. 2540-2544, 1994.

[75] Nelson, M.; Humphrey, W.; Gursoy, A.; Dalke, A.; Kale, L.; Skeel, R.; Schulten, K.: NAMD : A Parallel, Object-Oriented Molecular Dynamics Program, *International Journal of Supercomputer Applications and High Performance Computing*, Vol. 10, No. 4, pp. 251-268, 1996.

[76] Nyland, L.; Prins, J.; Reif, J.: A Data-Parallel Implementation of the Adaptive Fast Multipole Algorithm, *DAGS/PC Symposium*, Dartmouth College, Hannover, pp. 1-12, 1993.

[77] Parkinson, D.: Parallel Efficiency can be Greater than Unity, *Parallel Computing*, Vol. 3, pp. 261-262, 1986.

[78] Pfister, J.; Eberhard, P.: Frictional Contact of Flexible and Rigid Bodies, *Granular Matter*, Vol. 4, No. 1, pp. 25-36, 2002.

[79] Plimpton, S.: Fast Parallel Algorithms for Short-Range Molecular Dynamics, *Journal of Chemical Physics*, Vol. 117, pp. 1-19, 1995.

[80] Plimpton, S.; Hendrickson, B.: A New Parallel Method for Molecular Dynamics Simulation of Macromolecular Systems, *Journal of Computational Chemistry*, Vol. 17, pp. 326-337, 1996.

[81] Pimpton, S.; Hendrickson, B.: Parallel Molecular Dynamics with the Embedded Atom Method, *Material Research Society Symposium Proceedings MRS-291*, Pittsburgh, pp. 37-42, 1993.

[82] Potapov, A.; Campbell, C.: A Fast Model for the Simulation of Non-Round Particles, *Granular Matter*, Vol. 1, pp. 9-14, 1998.

[83] Potter, D.: *Computational Physics*, New York: Wiley, 1972.

[84] Pöschel, T.: Molecular Dynamics of Arbitrarily Shaped Granular Particles, *Journal of Physics I France 5*, Vol. 5, pp. 1431-1455, 1995.

[85] Pöschel, T.; Schwager, T.: *Computational Granular Dynamics, Models and Algorithms*, Heidelberg: Springer-Verlag, 2005.

[86] Pütz, M.; Kolb, A.: Optimization Techniques for Parallel Molecular Dynamics Using Domain Decomposition, *Computer Physics Communication*, Vol. 113, pp. 145-167, 1998.

[87] Rapaport, D.: Large-Scale Molecular Dynamics Simulation Using Vector and Parallel Computers, *Computer Physics Reports*, Vol. 9, pp. 1-53, 1988.

[88] Rapaport, D.: *The Art of Molecular Dynamics Simulation*, Cambridge: Cambridge University Press, 1995.

[89] Rea, A.: An Introduction to PVM,
see: *http://www.pcc.qub.ac.uk/tec/courses/pvm/ohp22/PVM-slides22.doc_1.html*, (Sep. 1995).

[90] Refson, K.: Moldy: A Portable Molecular Dynamics Simulation Program for Serial and Parallel Computers, *Computer Physics Communications*, Vol. 126, No. 3, pp. 309-328, 2000.

[91] Rhodes, M.: Introduction to Particle Technology, Chichester: Wiley-Interscience, 2005.

[92] Roth, J.; Gähler, F.; Trebin, H.R.: A Molecular Dynamics Run with 5.180.116.000 Particles, *International Journal of Modern Physics*, Vol. 11, No. 2, pp. 317-322, 2000.

[93] Sadd, M.; Tai, Q.; Shukla, A.: Contact Law Effects on Wave Propagation in Particulate Materials Using Distinct Element Modeling, *The International Journal of Non-Linear Mechanics*, Vol. 28, No. 2, pp. 251-265, 1993.

[94] Sanz-Serna, J.; Calvo, M.: *Numerical Hamiltonian Problems*, London: Chapman and Hall, 1994.

[95] Sawley, L.; Cleary, W.: *Parallel Discrete-Element Method for Industrial Granular Flow Simulation*, Csiro Mathematical & Information Sciences, Clayton, 1999.

[96] Schäfer, J.; Dippel, Wolf, D.; Force Schemes in Simulations of Granular Materials, *Journal de Physique I*, Vol. 6, pp. 5-20, 1996.

[97] Schiehlen, W.; Eberhard, P.: *Technische Dynamik* (in German), Stuttgart: Teubner, 2004.

[98] Schinner, A.: Fast Algorithms for the Simulation of Polygonal Particles, *Granular Matter*, Vol. 2, pp. 35-43, 1999.

[99] Severens, I.: *DEM Simulations of Toner Behavior in the Development Nip of the Oce Direct Imaging Print Process*, PhD thesis, Faculty of Mathematics and Computing Science, Eindhoven University of Technology, The Netherlands, 2005.

[100] Shabana, A.: *Dynamics of Multibody Systems*, New York: Wiley-Interscience, 1989.

[101] Sitharam, T.: Numerical Simulation of Particulate Materials Using Discrete Element Modelling, *Current Science*, Vol. 78, No. 7, pp. 876-886, 2000.

[102] Smith, W.: Molecular Dynamics on Hypercube Parallel Computers, *Computer Physics Communications*, Vol. 62, pp. 229-248, 1991.

[103] Smith, W.; Forester, T.: DL_POLY 2.0: A General-Purpose Parallel Molecular Dynamics Simulation Package, *Journal of Molecular Graphics and Modelling*, Vol. 14, No. 3, pp. 136-141, 1996.

[104] Standish, N.: The Kinetics of Batch Sieving, *Powder Technology*, Vol. 41, pp. 57-67, 1985.

[105] Standish, N.; Bharadwaj, A.; Hariri-Akbari, G.: A Study of the Effect of Operating Variables on the Efficiency of a Vibrating Screen, *Powder Technology*, Vol. 48, pp. 161-172, 1986.

[106] Standish, N.; Meta, I.: Some Kinetic Aspects of Continuous Screening, *Powder Technology*, Vol. 41, pp. 165-171, 1985.

[107] Subasinghe, G.; Schaap, W.; Kelly, E.: Modelling the Screening Process: A Probabilistic Approach, *Powder Technology*, Vol. 59, pp. 37-44, 1989.

[108] Swope, H.; Andersen, W.; Berens, P.; Wilson, K.: A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters, *Journal of Chemical Physics*, Vol. 76, pp. 637-649, 1982.

[109] Takahashi, Y.; Kataoka, M.; Uekusa, M.; Terumichi, Y.: Behavior of Three Kinds of Particles in Rotary Barrel with Planetary Rotation, *Multibody System Dynamics*, Vol. 13, No. 2, pp. 195-209, 2005.

[110] Tamayo, P.; Mesirov, J.; Boghosian, B.: Parallel Approach to Short Range Molecular Dynamics Simulations, *Proceedings of Supercomputing 91*, IEEE Computer Society Press, pp. 462-470, 1991.

[111] Tsuji, Y.; Tanaka, T.; Ishida, T.: Lagrangian Numerical Simulation of Plug Flow of Cohesionless Particles in a Horizontal Pipe, *Powder Technology*, Vol. 71, pp. 239-250, 1991.

[112] Van Gunsteren, W.; Berendsen, H.: Computer Simulation of Molecular Dynamics: Methodology, Applications and Perspectives in Chemistry, *Chemie-International Edition*, Vol. 29, pp. 922-1023, 1990.

[113] Verlet, L.: Computer Experiments on Classical Fluids, *Physical Review*, Vol. 159, pp. 98-103, 1967.

[114] Weiner, P.; Kollman, P.: AMBER: Assisted Model Building with Energy Refinement- A General Program for Modeling Molecules and their Interactions, *Journal of Computational Chemistry*, Vol. 2, pp. 287-303, 1981.

[115] Wessel, J.: Siebmaschinen (in German), *Aufbereitungstechnik*, Vol. 2, pp. 449-456, 1963.

[116] West, M.; Kane, C.; Marsden, J.E.; Ortiz, M.: Variational Integrators, The Newmark Scheme, and Dissipative Systems, *International Conference on Differential Equations*, Berlin, pp. 1009-1011, 1999.

[117] Wilson, G.: *Parallel Programming for Scientists and Engineers*, Cambridge: MIT Press, 1995.

[118] Wilson, R.; Ilnytskyi, J.: Parallel Computer Simulation Techniques for the Study of Macromolecules, *Computer Simulations of Liquid Crystals and Polymers*, Kluwer Academic Publishers, pp. 335-356, 2004.

[119] Yamane, K.; Nakagawa, M.; Tanaka, T.; Tsuji, Y.: Steady Particulate Flows in a Horizontal Rotating Cylinder, *Physics of Fluids*, Vol. 10, No. 6, pp. 1419-1427, 1998.

Symbols

In this part, the basic symbols and parameters used in this thesis are explained. Here, only the main ones are mentioned.

$\mathbf{a}_x$,$\mathbf{a}_y$,$\mathbf{a}_z$	acceleration vector given in the inertial frame
b	adjustable hole clearance of sieve holes
b_k	width of the halo region of processor k
c	overall TSM efficiency
c_i	individual efficiency of layer i of TSM
c_n	normal damping coefficient of contact
c_p	viscous damping coefficient among particles
c_t	tangential damping coefficient of contact
c_w	viscous damping coefficient with walls
c_x,c_y	flag integers in the x and y directions
d_w	wall thickness
e	geometrical dimension on the adjustable inclined plate of TSM
e_n	coefficient of normal restitution
g	global coordinate system, width between sieve aisles, number of gangue particles
$\mathbf{g}$	vector of gravity acceleration
$\mathbf{g}_h$	vector of global indices of the home particles
$\mathbf{g}_i$	vector of global indices of the immigrant particles
h	time integration step
h_k	halo region of processor k
i	counter
j	counter
k	counter
k_{ad}	adhesive spring coefficient
k_l	loading spring coefficient
k_n	normal spring constant
k_p	elastic spring constant among particles
k_t	tangential spring constant
k_{un}	unloading spring coefficient
k_w	elastic spring constant with walls
l	local (rotating) coordinate system

m	particle mass, unrounded number of holes, unit of length: meter
m_g	mass of gangue particles
m_{ij}	equivalent mass of particles i and j
m_s	mass of sorted particles
n	rounded number of holes in the radial direction of the sieve
$\mathbf{n}$	normal unit vector
p_k	processor k
r	particle radius
r_g	radius of gangue particle
r_{max}	radius of the largest particle
r_s	radius of sorted particle
r_v	radius of the Verlet circle
$\mathbf{r}$	vector of particles radii
$\mathbf{r}_{old}$	old position vector given in the inertial frame
$\mathbf{r}_x$,$\mathbf{r}_y$,$\mathbf{r}_z$	position vectors given in the inertial frame
$\dot{\mathbf{r}}_x$,$\dot{\mathbf{r}}_y$,$\dot{\mathbf{r}}_z$	velocity vector given in the inertial frame
$\ddot{\mathbf{r}}_x$,$\ddot{\mathbf{r}}_y$,$\ddot{\mathbf{r}}_z$	acceleration vector given in the inertial frame
s	number of sorted particles, unit of time: second
$\mathbf{s}$	vector of contact forces due to home-visiting interactions
t	time
t_0	starting time of simulation
t_c	time of contact
t_{end}	ending time of simulation
t_{max}	maximum time of penetration
t_p	parallel execution time, period of oscillation
t_s	serial execution time, starting time
$\mathbf{t}$	tangential unit vector
u	geometrical dimension on the adjustable inclined plate of TSM
v_0	initial velocity
$v_n^{(aft)}$	normal velocity after collision
$v_n^{(bef)}$	normal velocity before collision
$\mathbf{v}_x$,$\mathbf{v}_y$,$\mathbf{v}_z$	velocity vector given in the inertial frame
w	half width of the square hole of the TSM sieve
xyz	axes of the reference coordinate system
A_{ij}	element of matrix $\mathbf{A}$ of i row and j column
$\mathbf{A}_{gl}$	rotation matrix from local to global coordinate system
$\mathbf{A}_{lg}$	rotation matrix from global to local coordinate system
$\mathbf{E}$	merged matrix of the home and visiting particles
$\mathbf{f}$	vector of contact forces due to home-home and home-visiting interactions
$\mathbf{F}$	contact force

$\mathbf{F}^n$	normal contact forces
$\mathbf{F}^t$	tangential contact forces
G	percentage of gangue particles
$\mathbf{G}_v$	received contact force matrix of visiting particles due to home-visiting contacts
$\mathbf{H}$	matrix of the individual home particles
$\mathbf{I}$	matrix of the immigrant particles, inertia tensor
K	oscillation factor
K_g	global (inertial)coordinate system
K_l	local (rotating)coordinate system
L	width of the overall particle domain
L_k	width of the particle donmain of processor k
M	number of decks of the TSM
M_k	number of particles belongs to the halo-region h_k
$\mathbf{M}$	vector of applied torques
N	overall number of particles, unit of force: Newton
N_k	number of particles of processor k
$\mathbf{N}$	normal contact force
P	number of processors
$\mathbf{P}_h$	force matrix of home particles due to home-home & home-visiting interactions
$\mathbf{P}_v$	contact force matrix of visiting particles due to home-visiting interactions
R	barrel radius of the TSM
T	time period of oscillation
$\mathbf{T}$	tangential friction force
V	particle volume
V_g	volume of gangue particle
V_s	volume of sorted particle
$\mathbf{V}$	matrix of the visiting particles
$\mathbf{Y}$	generalized system coordinates
α	inclination angle in the frontal direction
$\boldsymbol{\alpha}$	vector of angular acceleration
β	inclination angle in the transverse direction
δ	spring elongation, penetration during overlap
$\dot{\delta}$	relative velocity
$\ddot{\delta}$	relative acceleration
δ_0	maximum overlap after unloading
δ_m	particle-mesh overlap
δ_{max}	maximum overlap
δ_r	overlap at start reloading
δ_t	stretching of the tangential spring
ϵ	shaft eccentricity

ϵ_m	momentum restitution coefficient
η	efficiency of parallel simulation, damping parameter
η_0	damping ratio
μ_d	dynamic friction coefficient
μ_h	hysteresis damping factor
μ_s	static friction coefficient
ν_d	normalized cost of parallel simulation
π	pi: 22/7
ρ	particle mass density, rate of outlet particles
$\boldsymbol{\rho}$	vector of particles density
τ	speedup of parallel simulation, dummy variable
ω	angular velocity of the TSM
ω_0	undamped frequency
ω_b	barrel angular velocity around its inclined axis
ψ	barrel angular displacement
ζ	non-dimensional scaling factor
Δt	integration time step
ΔT	dissipative energy in impact
Ω	natural frequency, particle angular velocity