Newcastle University
School of Engineering
Geotechnics and Structures

Image-informed numerical modelling of particulate systems with irregular grains

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy

Vasileios Angelidakis

Newcastle upon Tyne, UK

September 2021

Acknowledgements

First, I want to thank my supervisors, Professor Stefano Utili and Dr Sadegh Nadimi, for their kind and generous support over the past four years. The work presented in this thesis has been a team effort, refined by their guidance and critique, for which I am grateful.

Stefano trusted me with this project and introduced me to the world of discrete element simulations and open-source code development, an experience which has been an amazing intellectual journey. He never stopped challenging me, and giving me networking and research opportunities, which have ultimately helped me to further develop my skill set.

Sadegh has been incredibly generous in sharing his time and insight over the past years. I cherish our brainstorming sessions on particle shape characterisation, and how digital imaging can provide us with a deeper understanding of granular materials. Sadegh inspires me to be a better researcher and makes sure to lead by example, for which reason I feel lucky to have worked with him. He has my most sincere thanks for contributing to my academic growth.

I want to extend my gratitude to Dr Vasilis Sarhosis, from the School of Civil Engineering of the University of Leeds, UK, for his kind support and advice during his time at Newcastle University.

Newcastle University is gratefully acknowledged for providing my doctoral scholarship.

I would like to thank Dr Masahide Otsubo from the University of Tokyo, Japan, for providing numerical simulations of the triaxial behaviour of sand and rice grains in Chapter 4, using the open-source molecular-dynamics code LAMMPS.

I want to thank Professor Fang Liu from the Department of Geotechnical Engineering at the College of Civil Engineering, Tongji University, China, for kindly hosting me at Tongji University, during my secondment in Shanghai. I owe gratitude to Professor Junhua Xiao and his research group from the College of Transportation, Tongji University, for open-handedly providing me with the scans of railway ballast grains used in Chapter 3 and Chapter 6. The time and efforts of Dr De Zhang in particular are gratefully acknowledged, in providing said scans.

I owe my most sincere thanks to Dr Chia Weng Boon, who has been extremely generous with his time for the past years, guiding me through the source codes of the `Potential Blocks` and `Potential Particles`, developed inside the open-source code YADE, during his doctoral studies at the University of Oxford. His specialised insight has saved me significant time, while our many brainstorming and troubleshooting sessions on Skype have always been interesting and productive.

I would like to thank Professor Katalin Bagi and Ákos Orosz, from the Budapest University of Technology and Economics, Hungary, for our interesting research interactions on developing novel methods for the characterisation of particle form.

I want to thank the developers' and users' communities of YADE for their tireless eagerness to help, to answer questions (or sometimes to interestingly counter-ask them) and to propose solutions to numerical and theoretical issues arising from DEM simulations. Their drive and expertise helped me overcome several algorithmic obstacles during the past four years and motivated me to appreciate and support open-source endeavours even more. In particular, Bruno, Janek, Jan, Jérôme, Robert, thank you all.

Based on my experience with academia so far, I have found research work equally fascinating and draining on an intellectual and emotional level. Having a support system has been key in keeping my motivation and productivity high. I want to thank the people closest to me, my family and dear friends, for their continuous love and support which doesn't stop to amaze me every single day.

Vasileios Angelidakis
Newcastle upon Tyne, UK
September 2021

Abstract

Granular materials are everywhere around us. Their omnipresence makes our interaction with them on a daily basis a certainty, and yet our understanding of their mechanical behaviour is far from complete. Regarding geotechnical applications, most natural granular materials, such as silts, sands, gravels and ballast, feature irregular particle shapes, a fact that makes their mechanical behaviour all the more complex across scales, from micro to meso and macro. A multitude of experimental and numerical studies have demonstrated the importance of particle morphology in the shear strength of particulate materials, although rarely demonstrating a direct link or mechanisms of causality between them. This is mainly due to the high complexity of the problem but also partially due to the lack of intelligible and accessible tools to quantify the morphology of three-dimensional irregular particles.

This thesis aims to contribute to the current state-of-art studying the characterisation of granular materials by providing analytical and numerical tools for shape characterisation. Regarding analytical tools, this thesis attempts a critical review of existing indices to characterise and classify particle form, while introducing a new set of indices. Regarding numerical tools, this thesis provides novel software solutions for automatic particle shape characterisation and for the generation of image-informed numerical models. These open-source tools are meant to shed light on the inherent subjectivity of performing shape characterisation on a practical level. Regarding the generation of numerical models based on imaging data, algorithmic implementations are offered to create simplified polyhedra and multi-sphere particles at user-defined fidelity levels of resolution, the morphology of which can also be characterised and compared to that of the original fidelity level.

Combining the produced analytical and numerical tools, this thesis demonstrates a seamless workflow between particle imaging data and numerical modelling, using the discrete element method and non-spherical particles. This workflow is utilised to develop a methodology for the generation of Representative Element Volumes (REVs) of non-spherical particles, which represent the polydispersity of both particle size and shape, aiming to link quantitative morphology characterisation at the particle scale and mechanical characterisation at the level of a representative assembly of particles. The methodology is then applied to systematically generate REVs of railway ballast using image-informed multi-sphere particles of various levels of simulation fidelity, allowing for a parametric study of the effect of several modelling parameters on the shear strength of the material.

Publications

- **Angelidakis, V.**, Nadimi, S. and Utili, S., 2022. Elongation, flatness and compactness indices to characterise particle form. *Powder Technology*, **396**, pp.689-695. `https://doi.org/10.1016/j.powtec.2021.11.027`
- **Angelidakis, V.**, Nadimi, S., Otsubo, M. and Utili, S., 2021. CLUMP: A Code Library to generate Universal Multi-sphere Particles. *SoftwareX*, **15**, p.100735. `https://doi.org/10.1016/j.softx.2021.100735`
- **Angelidakis, V.**, Nadimi, S. and Utili, S., 2021. SHape Analyser for Particle Engineering (SHAPE): Seamless characterisation and simplification of particle morphology from imaging data. *Computer Physics Communications*, **265**, p.107983. `https://doi.org/10.1016/j.cpc.2021.107983`
- Orosz, A, **Angelidakis, V.** and Bagi, K., 2021. Surface orientation tensor to predict preferred contact orientation and characterise the form of individual particles. *Powder Technology*, **394**, pp.312-325. `https://doi.org/10.1016/j.powtec.2021.08.054`
- Dosta, M., André, D., **Angelidakis, V.**, Caulk, R., Celigueta, M.,Á., Chareyre, B., Dietiker, J.,F., Girardot, J., Govender, N., Hubert, C., Kobyłka, R., Moura, A.,F., Skorych, V., Weatherley, D.,K. and Weinhart, T., 202X. Comparing open-source DEM frameworks for simulations of common bulk processes (under review).
- **Angelidakis, V.**, Nadimi, S., Utili, S., 202X. A methodology to generate representative element volumes in the DEM using realistic particle shapes. *Géotechnique* (in preparation).

Published software

- **SHAPE**: SHape Analyser for Particle Engineering. Role: Main developer. github.com/vsangelidakis/SHAPE. GPLv3.0.
- **CLUMP**: Code Library to generate Universal Multi-sphere Particles. Role: Main developer. github.com/vsangelidakis/CLUMP. GPLv3.0.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Background

The discrete element method (DEM) has been extensively used during the last four decades for the study of granular materials. Being able to simulate grain-to-grain interactions at the particle scale, it has provided valuable insights on the micro to macro effects of geomaterials. The majority of discrete element studies still employ spherical particles, while recently, increasing attention has been brought to the merits of non-spherical particle modelling.

Although the effect of the grain shape has been identified to be paramount on the mechanical response of geomaterials (Barrett, 1980, Clayton et al., 2009), its significance is often highlighted qualitatively. In particular, the particle shape of realistic grains is employed only in a small -but increasing- part of the relevant literature, with the majority of past studies using either regularized shapes, like ellipsoids, superquadratics (Podlozhnyuk et al., 2017) and platonic solids (André et al., 2012) or randomly generated polyhedral particle shapes (Eliáš, 2014).

The advancement of imaging technologies, complemented by the increased state-of-the-art computing capabilities during the past decade has allowed for more numerical studies considering realistic particle shapes. The challenge ahead is to generate assemblies of irregular particles using the Discrete Element Method, which can lead to representative mechanical behaviour.

Typically, the particles used in numerical simulations have simplified morphological features compared to the originally scanned one. This reduction of fidelity is a necessary step, due to the high computational cost associated to the modelling of very detailed particle morphologies. Albeit necessary, this poses a challenge, as oversimplification of the particle morphology can lead to a mismatch with reality, for models that are otherwise image-informed. The achieved fidelity is closely related to the simplification approach employed during particle generation in the DEM, as well as to the metrics used to quantify fidelity. For instance, fidelity can be quantified in terms of number of particle faces making the surface of a polyhedron or in terms of number of spheres making a multi-sphere particle.

As a result, it becomes evident that the particle type, the measure of fidelity and and the simplification method during particle generation in a numerical simulation do not allow for a universal answer to exist in the question of which fidelity level corresponds to a good balance between computational efficiency and accuracy of results. Instead, quantitative shape characterisation can provide a solution to this, via comparison of the morphology of the simplified particle to the original particle morphology, for each new material of interest and employed modelling technique. If a shape analysis software can provide automated particle shape characterisation, simplification and model generation in a numerical simulation at user-defined fidelity levels, the required level of fidelity can be estimated in terms of morphological fidelity (compared to the original fidelity level) and accuracy of results (comparing models of different morphological fidelity with some form of "ground truth", usually an experimental benchmark).

Utilising shape characterisation information can be used in the generation of numerical models. In particular, shape characterisation can provide a means of selecting which particles to use in a simulation of irregular particles, at what percentages and fidelity levels. This can be facilitated with a methodology to generate representative element volumes that represent both particle size and shape based on imaging data, in an automated and seamless manner.

1.2 Aim and objectives

This thesis aims to investigate particle shape effects on the macro-mechanical behaviour of granular media, through the prism of quantitative shape characterisation. A key point of interest is to build a methodology towards linking particle morphology and mechanical behaviour, which will for the quantification of how much the morphology of real particles can be simplified and still be called realistic. To this end, multi-sphere particles were employed using the DEM, to simulate irregular particles based on imaging data of real particles. Particle shapes are reconstructed from available digital imaging data of geomaterials, using an open-source shape characterisation code, `SHAPE`, which was developed in-house, as part of this doctoral thesis, and was distributed under the open-source GPLv3.0 licence in Angelidakis et al. (2021b). `SHAPE` also generates simplified particle shapes for different user-defined levels of morphological fidelity. In the efforts to create numerical models with particle shapes informed by imaging data, a second code was developed as part of this thesis, `CLUMP` (Angelidakis et al., 2021a), providing implementations of different methods to generate multi-sphere particles, licensed under GPLv3.0 as well. This thesis focuses on materials with irregular grains, like the products of crushed rock, with the main engineering application concerning railway ballast. Though, the numerical and analytical tools developed as part of this thesis have application to any granular material with irregular particles.

The main objectives of this doctoral thesis are as below:

- $\mathbf{O_1}$**:** To critically review existing formulae that have been proposed to characterise and classify particle form, identify their limitations and develop a new characterisation and classification system to overcome them.

- $\mathbf{O_2}$**:** To develop open-source numerical tools for automatic particle shape characterisation of assemblies with three-dimensional particles and for the generation of simplified polyhedral and multi-sphere particles, based on digital imaging data.

- $\mathbf{O_3}$**:** To conduct particle-shape characterisation analyses for datasets of real particles, generate simplified ones, which can be used in numerical simulations, and compare the morphology of the original and simplified particles.

- $\mathbf{O_4}$**:** To generate and simulate Representative Element Volumes (REVs) of irregular particles under triaxial loading conditions using the Discrete Element method. These REVs will be generated to faithfully represent the statistical distribution of both particle size and shape, utilising the produced codes, to create particles with varying, user-defined levels of morphological fidelity.

Figure 1.1 illustrates a flowchart with the main steps of a methodology employed in this thesis to generate discrete element simulations across scales, utilising available digital imaging data of ballast grains. Starting from imaging data representing particle geometry, such as point clouds and surface meshes (derived with laser scanning) or as labelled voxelated images (derived using computed tomography) the surface of the particles is reconstructed and their interior is tessellated. Their morphology is characterised at the particle level, in an automated manner, and simplified particle shapes are generated at fidelity levels affordable by the conventional available numerical tools. The simplified particle can be represented either as convex polyhedra or multi-sphere particles.

Mechanical characterisation ensues at the scale of a representative assembly of particles, using the discrete element method. The particle shapes and material properties resulting from the morphological and mechanical characterisation, respectively, can then be used in boundary value models of real engineering applications. In this methodology, the right balance can be struck between computational efficiency and numerical accuracy, as multiple levels of simulation fidelity can be employed, based on the available computational resources and the needed level of accuracy for the project at hand.

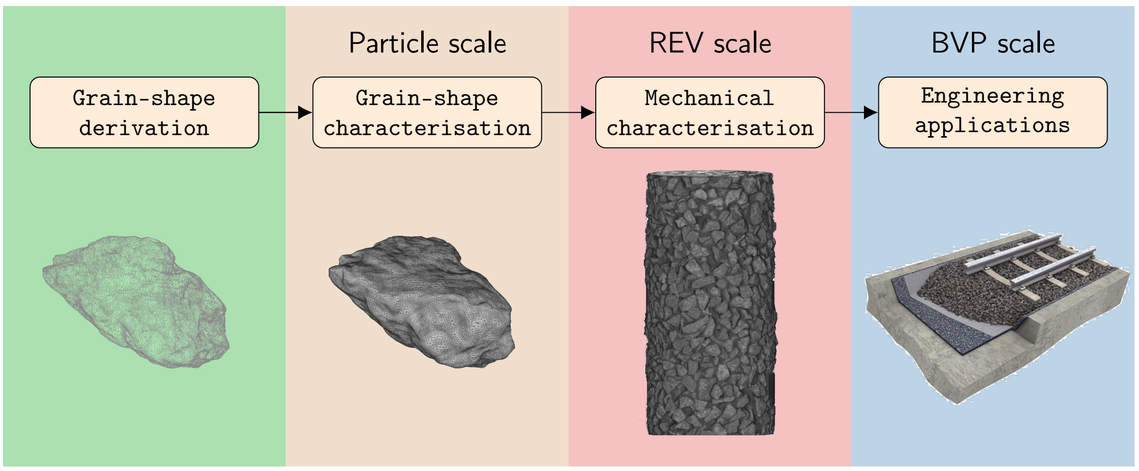


Figure 1.1: Scales for material characterisation.

Figure 1.2 demonstrates the morphology of a real ballast particle and two candidate methods for numerical simulation: (i) convex polyhedra and (ii) clumps of overlapping spheres. The former simulation approach suffers from the requirement of the particles to be convex, and requires a specialised and computationally intensive contact detection algorithm. The latter simulation approach does not require particles to be convex and the contact detection among spheres is computationally efficient; though, using a large number of sphere-members to represent a single physical particle increases the computational cost significantly. These two modelling approaches are of interest, as they allow the simulation of a wide spectrum of particle shapes, of various form (elongated, flat or compact) and roundness (angular or rounded) values. The flowchart of Figure 1.3 demonstrates a typical workflow to generate numerical models based on imaging data.

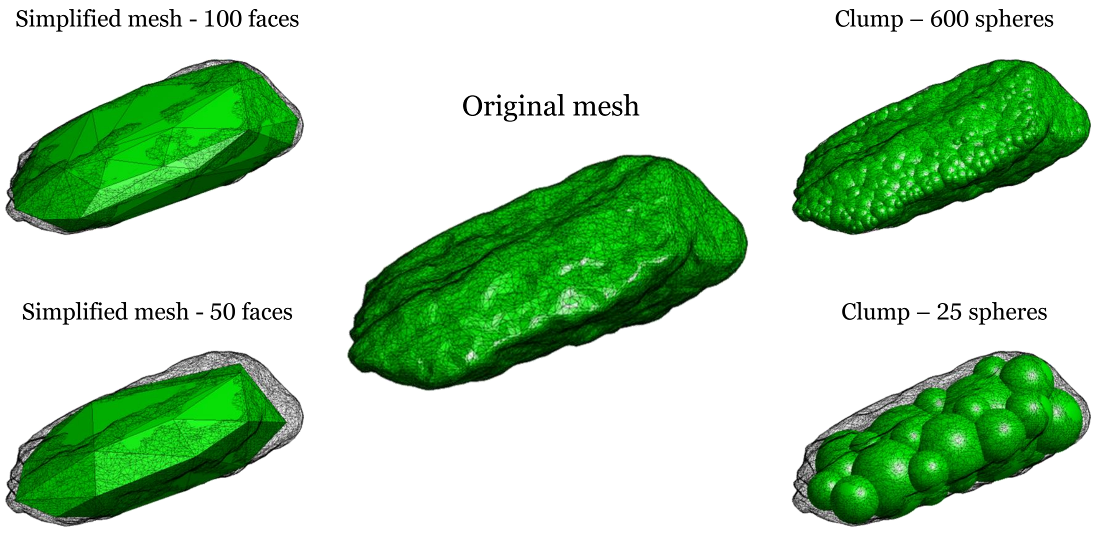


Figure 1.2: Original and simplified representations of particle morphology (left) convex polyhedra; (right) clumps of spheres.

1.3 Outline of the dissertation

Chapter 2 critically reviews various sets of indices proposed in the literature to characterise the form of three-dimensional particles. Then, a new set of indices for particle elongation, flatness and compactness is proposed, which overcomes limitations of similar existing indices, while possessing some of their merits. These new indices are used to propose a new classification system for particle form, which is in general agreement with the Zingg system, while it surmounts some of its limitations for very flat and for very elongated particles.

Chapter 3 introduces a novel open-source code to characterise the morphology of three-dimensional particles. The input particle geometries can have various formats, including labeled voxelated images, surface and tetrahedral meshes, point clouds and surface texture profiles, allowing compatibility with a variety of digital imaging techniques, such as computed tomography, laser scanning and white-light interferometry. Particle morphology is characterised in terms of three aspects, namely form, roundness and roughness. Outputs are offered in the format of the most prominent FEM

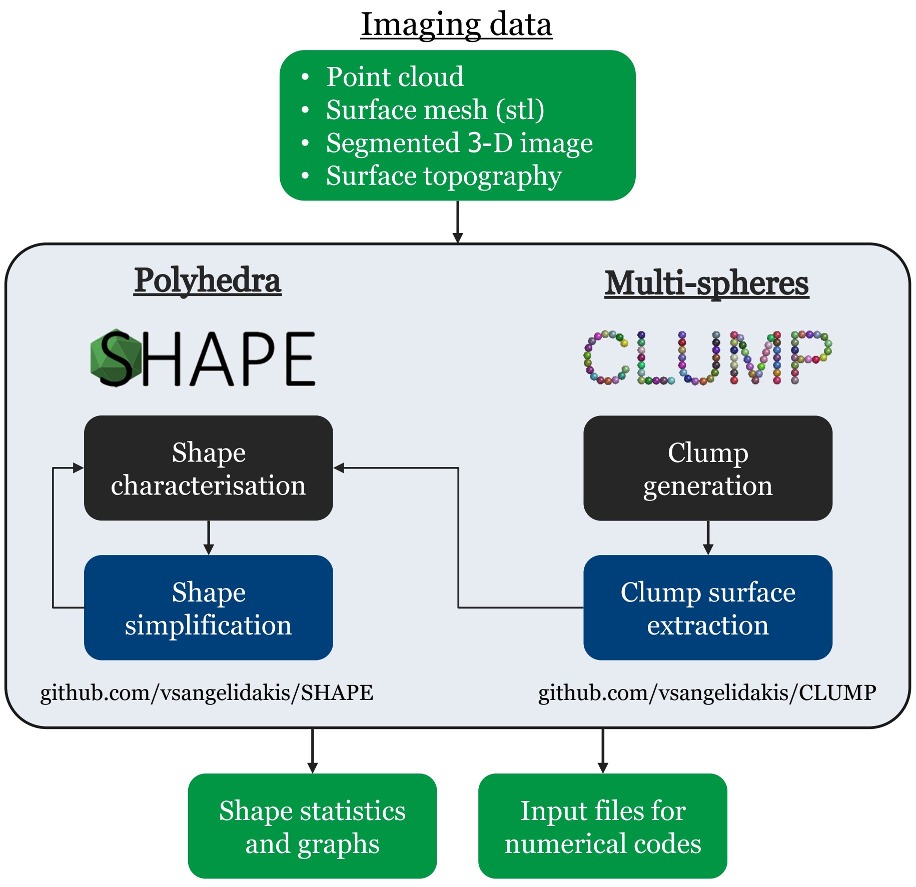


Figure 1.3: Typical workflow to generate multi-sphere particles and simplified polyhedra based on imaging data.

and DEM codes currently used in research and industrial practice.

Chapter 4 introduces a novel open-source code to generate multi-sphere particles, which can be used in Discrete Element simulations. This code, named `CLUMP`, comprises implementations of two algorithms proposed in the literature to approximate the geometry of three-dimensional particles using clumps of overlapping or clusters of non-overlapping spheres, while it proposes a new generation method for multi-sphere clumps and clusters. A routine to extract the surface of a multi-sphere particle was developed, allowing for a detailed characterisation of its morphology. Triaxial tests on sand and rice grains are performed using different particle generation methods, demonstrating how the clump generation method can affect the simulation results at the scale of a Representative Elementary Volume.

Chapter 5 discusses two codes to simulate non spherical particles in the Discrete

Element Method, namely the `Potential Blocks` and the `Potential Particles`, which were developed by Boon (2013) in the open-source code YADE by Smilauer et al. (2021) to simulate convex polyhedra and generalised three-dimensional convex particles, respectively. Points of interest are highlighted for both codes, along with some of their limitations, and applications are demonstrated where available. Part of this chapter has been shared in the documentation of YADE[1] by the doctoral student, while he was developing this part of the thesis.

Chapter 6 introduces a novel methodology to generate Representative Elementary Volumes using non-spherical particles in the Discrete Element Method, based on morphological characterisation of their overall form. A case study for railway ballast particles is performed, demonstrating the validity of the method via DEM simulations of multi-sphere particles.

Chapter 7 illustrates the conclusions of this thesis.

Appendix A presents some key algorithmic implementations of `SHAPE`, complemented with inline comments, aiming to enhance an easy overview of these scripts.

Appendix B presents the main functions of `CLUMP`, implementing three methods to generate multi-sphere particles and a proposed method to extract their surface, allowing for a detailed characterisation of their morphology.

1.4 Contributions

This section aims to clarify the contributions of the doctoral student to each chapter and corresponding piece of published work, where applicable, along with the contributions of the supervisory team (Dr Nadimi and Prof. Utili) and external collaborators.

In Chapter 2, the doctoral student visualised the values of the existing indices of particle form on a Zingg plot, proposed the new formulae for elongation, flatness and compactness and the new classification system. The supervisory team (Dr Nadimi and Prof. Utili) reviewed this work, provided feedback on the produced results and proposed the examples used to compare different sets of indices.

In Chapter 3, the doctoral student developed the majority of the scripts to calculate geometric parameters of three-dimensional particles, characterise their morphology and create an object-oriented data architecture to store these, allowing for an efficient automated processing of large amounts of particles. Dr Nadimi provided

[1] `https://yade-dem.org/doc/potentialparticles.html`

some functions relating to surface roughness. The full supervisory team (Dr Nadimi and Prof. Utili) reviewed the chapter and the accompanying manuscript in Computer Physics Communications (Angelidakis et al., 2021b), and proposed feedback and ideas regarding case studies which were used for code validation.

In Chapter 4, the doctoral student developed all algorithmic implementations of the clump-generating methods of Favier et al. (1999) and Ferellec and McDowell (2010), along with the newly proposed clump-generation method based on the Euclidean transform of three-dimensional images and a surface extraction module, which is used to facilitate shape characterisation of the produced multi-sphere particles. The student also generated the clumps used in the Illustrative examples section of the chapter, and characterised their morphology. Dr Nadimi reviewed the code in its early stages, providing feedback and much-needed alpha-testing (i.e. testing within the developers group) and provided the physical material of the analysed sand particle and performed micro Computed Tomography (µCT) scans of the analysed sand and rice particles and post-processing of the produced images, while he facilitated the link to collaborate with the group of Dr Otsubo. Dr Masahide Otsubo from the University of Tokyo provided the physical rice material used in the study and performed all the numerical triaxial simulations of the sand and rice particles using the Discrete Element Method and the granular package of the LAMMPS code, in particular, along with post-processing of the results. The full supervisory team (Dr Nadimi and Prof. Utili) and Dr Otsubo reviewed the final version of the manuscript and offered edits and feedback.

In Chapter 5, the described Potential Particles and Potential Blocks codes were developed in Boon (2013). The student has undertaken the role of maintaining these codes within the YADE developers' team during his doctoral project and any original contributions and improvements are pointed out to the reader of this thesis, aiming to clarify what constitutes an original contribution associated with this thesis and what belongs to the existing literature. Regarding the "Round Robin test of angle of repose", the project and full experimental work was organised by the Japanese branch of Technical Committee 105: Micro to Macro (TC105). The doctoral student developed all numerical models using the Potential Particles code and post-processed all the results. The supervisory team (Dr Nadimi and Prof. Utili) provided feedback on aspects of generating the numerical model and writing the report for the submission of the results produced in this thesis to the committee organising this Round Robin exercise. Improvements on the algorithmic development of these codes by the doctoral student are discussed utilising a piece of documentation [2] the student developed as part of this thesis, detailing the underlying concepts behind these codes and providing practical tutorials.

In Chapter 6, the full team (student and supervisors) agreed on the idea to create a method for the generation of Representative Element Volumes that reproduced to a prescribed degree the statistical distribution of particle form, for of a material with irregular particles. The doctoral student performed all tasks pertaining to shape characterisation and simplification of the ballast material, numerical simulations of triaxial compression tests, and post-processing. Scans of the ballast grains were generously provided by the research group of Prof. Junhua Xiao, from the College of Transportation Engineering of Tongji University, China. The supervisory team offered feedback on the morphological and mechanical characterisation results, along with overall guidance.

[2] `https://yade-dem.org/doc/potentialparticles.html`

Chapter 2

Characterisation of particle form

2.1 Introduction

A century after the first attempts of Wentworth to characterise the shape of cobbles, our understanding of particle morphology is still expanding. A plethora of shape indices has been proposed in the literature to characterise the morphology of individual particles. This chapter aims to shed light on the merits and limitations of the indices currently used to characterise particle form (elongation, flatness and compactness) adopting a unified classification framework. Second, new indices for elongation, flatness and compactness are proposed to address the identified shortcomings. Third, a new particle classification system derived from the proposed indices is illustrated. It is shown the new system overcomes the misclassification of a range of particles that are incorrectly classified as bladed in the Zingg system.

2.2 Literature review: Existing indices for particle form

Particle morphology is of interest in soils and rocks description, characterisation and classification and has been shown to bear significant influence on the mechanical and hydraulic behaviour of soils and rock aggregates (e.g. Cho et al., 2006; Boon et al., 2015b; Altuhafi et al., 2016; Kawamoto et al., 2018; Nguyen and Indraratna, 2020). Outside the scope of soil mechanics, particle morphology is believed to be the controlling factor in spreading of powder bed in additive manufacturing (Nan et al., 2018), wave velocity anisotropy (Otsubo et al., 2020), railway track substructure design (Le Pen et al., 2013) and pharmaceutical tableting (Abdel-Hamid et al., 2011), just to name a few. The morphology of a particle is characterised in terms of three aspects, namely form (first order approximation of shape), roundness (second order) and roughness (third order), illustrated in Figure 2.1. This chapter deals with form.

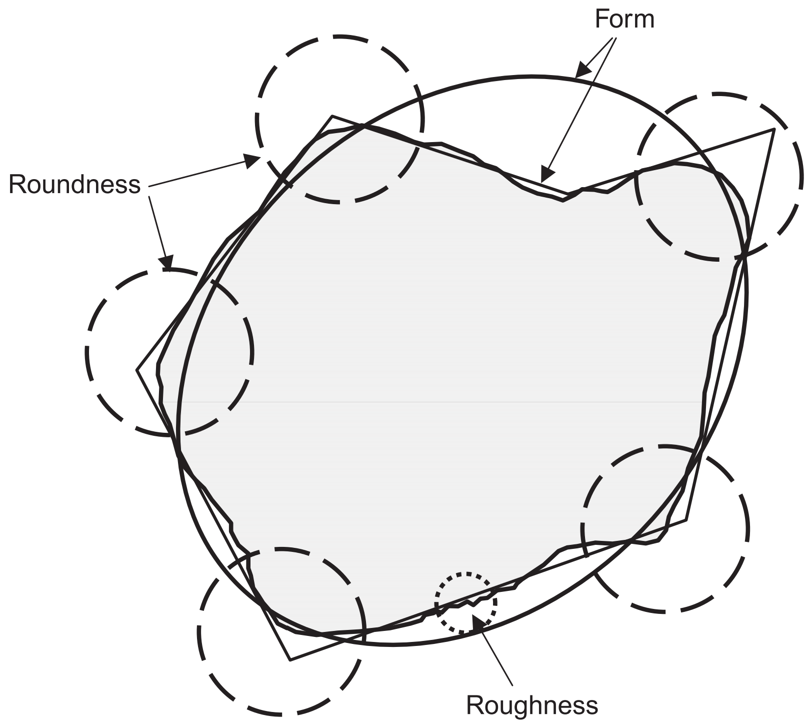


Figure 2.1: Aspects of particle shape as introduced in Barrett (1980): Form, roundness and roughness (after Clayton et al., 2009).

The characterisation of the form of three-dimensional particles is mostly performed via the calculation of two independent indices: flatness and elongation. Flatness is meant to express how flat (or flaky or platy) a particle is, while elongation is meant to express how elongated (or rod-like) it is. Some approaches consider a third index as well, compactness. However, compactness is calculated from flatness and elongation so the latter two indices are enough to fully characterise the form of a particle. Apart from flatness and elongation, several indices representing sphericity have been proposed in the literature (Rorato et al., 2019), although some consider sphericity a separate aspect of particle morphology, rather than another index related to particle form (Blott and Pye, 2008).

All traditional formulae for flatness and elongation make use of the three main dimensions of a particle, here called $a > b > c$ as in Zingg (1935). In the case of regular particle geometries (e.g. cuboids or ellipsoids), the main particle dimensions are unambiguously defined, whereas for irregular particles, the definition of these dimensions can be ambiguous. These dimensions provide a simplified description of the form of a particle, and can be calculated using a variety of methods. For instance, Fonseca (2011) assumes a,b,c as the dimensions of an oriented bounding box, calculated using Principal Component Analysis, while Potticary et al. (2015) assume the axes of an equivalent ellipsoid as the main particle dimensions.

This chapter first describes the classification system proposed by Zingg (1935) and then compares the indices proposed by Kong and Fonseca (2018), Bagi and Orosz (2020) and Potticary et al. (2015) in a Zingg plot. This enables a direct comparison of the indices for particles of the same aspect ratios. By clarifying the advantages and limitations of each set of indices, the case is laid out for the introduction of a new set of indices. The advantages of the new indices are illustrated. Then, a new particle classification system derived from the new indices which overcomes the shortcomings of the Zingg system is described.

2.2.1 Interpretations of sphericity

Comparing the morphological characteristics of irregular particles with the ones of idealised shapes has been used in the literature to produce indices associated with sphericity. Sphericity is meant to represent how closely the morphology of a particle resembles that of a sphere. Wadell (1932) defined the Degree of true sphericity as the ratio of the surface area of a sphere with the same volume as the particle, over the surface area of the particle. For a given volume, a sphere is the shape with the minimum surface area, and as a result the value of this index varies within the range (0-1) for non-spherical particles and is equal to unity for spheres. Equation 2.1 demonstrates the formula for the degree of true sphericity.

$$S = \frac{\sqrt[3]{36\pi \cdot V^2}}{A} \tag{2.1}$$

Blott and Pye (2008) report that the degree of true sphericity as proposed by Wadell (1932) is affected both by form and roundness, hence they consider it a separate aspect of particle form. To visualise this, Figure 2.2 shows how the degree of true sphericity varies for cuboids and ellipsoids of all possible aspect ratios, by plotting contour maps of its value on a Zingg plot. It becomes evident that cuboids and ellipsoids of the same aspect ratio do not demonstrate the same sphericity values.

In particular, the degree of true sphericity for a cube is equal to 0.804, while for a sphere it is 1.0. For both types of particles, it is observed that sphericity increases moving towards the upper-right region of the plot, which corresponds to compact particles.

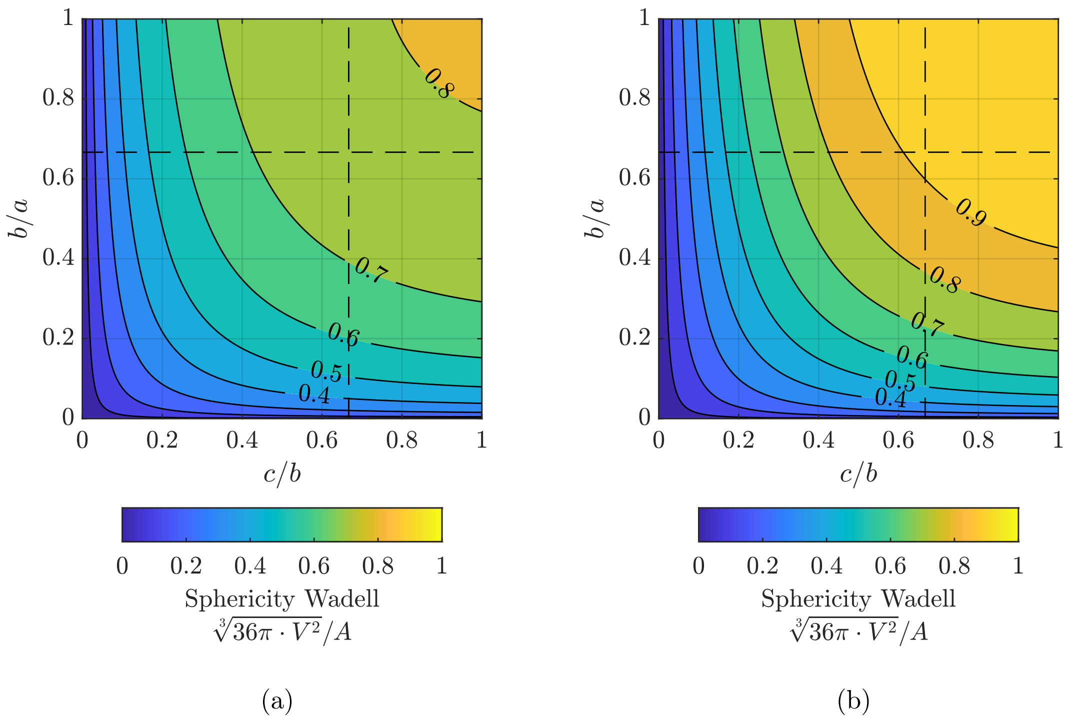


Figure 2.2: Degree of true sphericity (Wadell, 1932) for (a) cuboids and (b) ellipsoids.

Krumbein (1941) proposed an alternative definition of sphericity, named 'Intercept sphericity', aiming to introduce a more efficient way to measure particle form experimentally, using calipers, by expressing particle form as that of a triaxial ellipsoid. Using his approach, sphericity can be calculated just by determining three main orthogonal dimensions of the particle, without needing to compute the particle volume or surface area. The formula for intercept sphericity is shown in Equation 2.2.

$$S = \sqrt[3]{\frac{b \cdot c}{a^2}} \tag{2.2}$$

Sneed and Folk (1958) criticised the degree of true sphericity of Wadell (1932), observing that *"although geometrically valid, [it] is not a behavioristic parameter if one is concerned with the dynamics of particles under natural hydraulic conditions"*. To support this position, they compared two spheroids with the same degree of true sphericity, one elongated (dimensions 100 x 10 x 10 cm) and one flat (dimensions 100 x 100 x 1 cm), illustrated in Figure 2.3. They observed that the flat spheroid is more inequidimensional, even though it has the same degree of true sphericity, as it would settle more slowly if left to settle in water, due to its larger projection area. Aiming to produced a more "natural" measure of sphericity, which takes into account also the hydraulic behaviour of particles, they proposed the "Maximum projection sphericity", which is defined as the ratio of the maximum projection area of a sphere of the same volume as the particle, over the maximum projection area of the particle itself, demonstrated in Equation 2.3. The aforementioned spheroids take values of maximum projection sphericity of 0.464 for the elongated one and 0.046 for the flat one, while they both demonstrate a degree of true sphericity of 0.216.

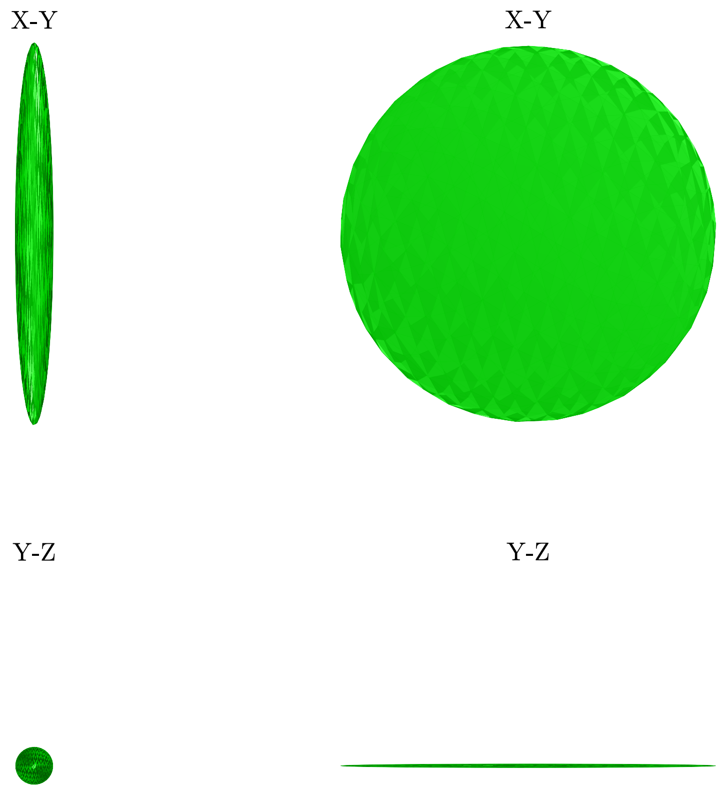


Figure 2.3: Spheroids with same degree of true sphericity (Wadell, 1932) from two different perspectives: (left) prolate spheroid with dimensions 100 x 10 x 10 cm; (right) oblate spheroid with dimensions 100 x 100 x 1 cm.

$$S = \sqrt[3]{\frac{c^2}{a \cdot b}} \tag{2.3}$$

Contrary to the sphericity of Wadell (1932), which is affected by both form and roundness, the formulae of Intercept sphericity (Krumbein, 1941) and Maximum projection sphericity (Sneed and Folk, 1958) are calculated based on a simplified representation of the particle form, which is thought to be described fully through the three main particle dimensions a, b, c and thus, an ellipsoid, a cylinder and a cuboid with the same dimensions are considered to have the same intercept and maximum projection sphericity values (Clayton et al., 2009). For example, a sphere, a cylinder with equal height and diameter, and a cube take values of intercept and maximum projection sphericity equal to unity. It can be asserted thus, that these indices do not measure the morphological resemblance of a particle to a sphere, like the degree of true sphericity. Instead, these indices quantify particle *equidimentionality*, which has also been called in the literature as *equancy* or *compactness*. Figure 2.4 illustrates contour maps for the intercept and the maximum projection sphericity on a Zingg plot, for particles of all possible aspect ratios. It becomes evident these two formulae produce similar values for the same aspect ratios, while their contour maps are symmetric along the bisect line of the Zingg plot, $c/b = b/a$. Indeed, both these sphericity indices take higher values as particles become more compact, comparing with the Zingg classification system (dotted lines on the graphs).

2.2.2 Interpretations of elongation, flatness and compactness

Zingg (1935) proposed one of the most popular classification systems for form, considering two aspect ratios to define the form of a particle: c/b being particle flatness and b/a being elongation (as shown in Equation 2.4).

$$\begin{aligned} el &= \frac{b}{a} \\ fl &= \frac{c}{b} \end{aligned} \tag{2.4}$$

Plotting the Zingg parameters c/b and b/a in a chart provides an intuitive visualisation of the form of a particle, in what is widely referred to as a *Zingg plot*. Zingg (1935) categorised particle morphology in four classes: oblate if $c/b < 2/3$ and $b/a > 2/3$; compact if $c/b > 2/3$ and $b/a > 2/3$; blade-like if $c/b < 2/3$ and $b/a < 2/3$ or prolate if $c/b > 2/3$ and $b/a < 2/3$, as shown in Figure 2.5a. Although Zingg (1935) did not use the term flatness nor elongation, b/a and c/b have been referred to as elongation and flatness respectively in the subsequent literature (e.g. Blott and Pye, 2008).

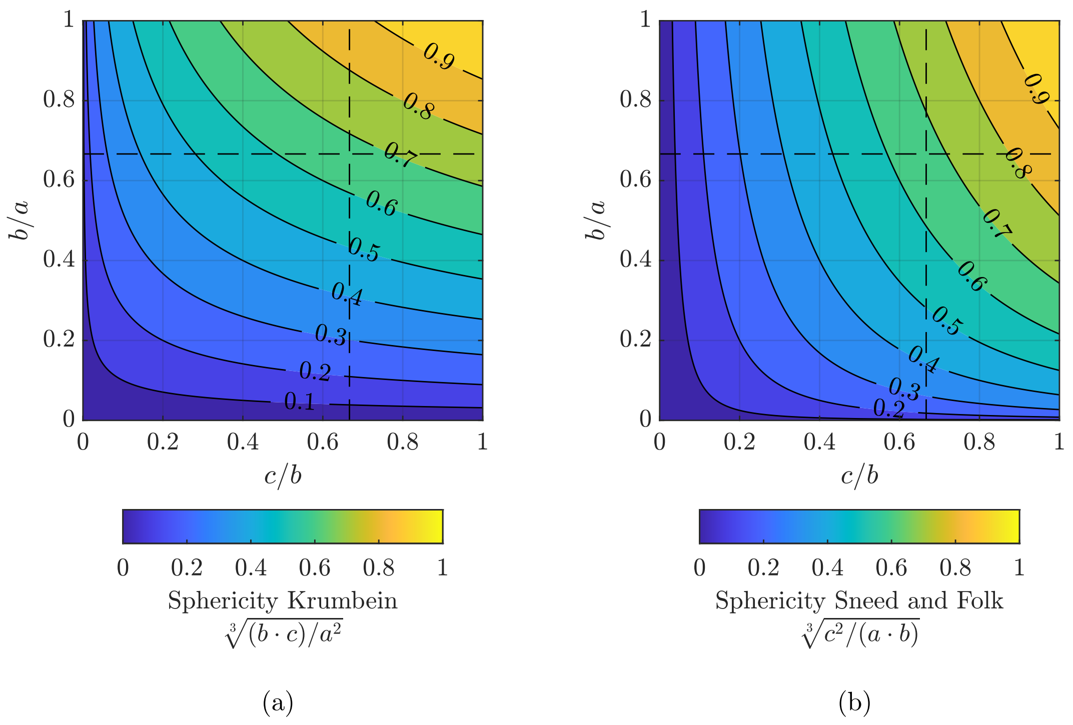


Figure 2.4: Indices for (a) Intercept sphericity (Krumbein, 1941) and (b) Maximum projection sphericity (Sneed and Folk, 1958).

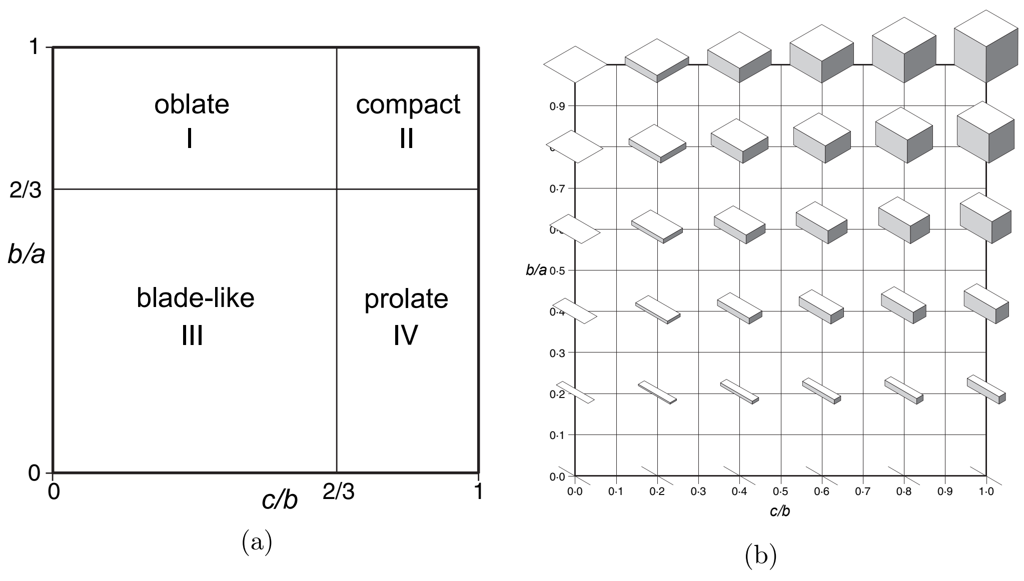
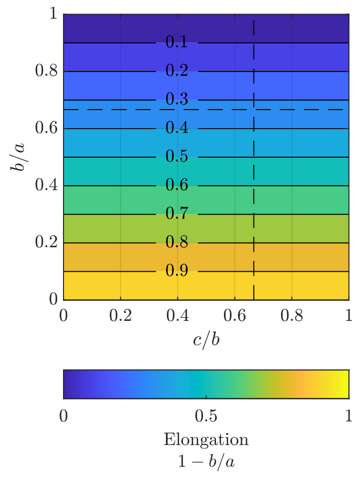


Figure 2.5: (a) The shape classification system of Zingg (1935); (b) cuboids with varying aspect ratios on a Zingg plot (modified from Blott and Pye (2008) with permission from Wiley & Sons).

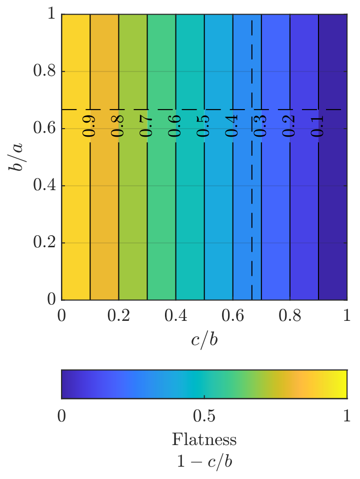
Kong and Fonseca (2018) proposed new formulae for flatness and elongation given in Equation 2.5. These indices can be thought of as the complementary to unity of the ratios in Equation 2.4.

$$
\begin{aligned}
el &= 1 - \frac{b}{a} \\
fl &= 1 - \frac{c}{b}
\end{aligned}
\tag{2.5}
$$

The advantage of the Kong and Fonseca (2018) formulae is that a flat particle takes a high value of flatness, an elongated particle takes a high value of elongation, a blade-like particle (both flat and elongated) takes a high value of both flatness and elongation, while a compact particle takes a low value for both indices. Also these indices are easy to understand and visualise (see the Zingg plots of Figure 2.6). However, a limitation of the indices is that they do not express flatness nor elongation as percentages of an overall form. For example, a particle with $fl = 0.8$, so $c/b = 0.2$, is not compact, but it could be either flat, if $el < 1/3$, so $b/a > 2/3$, or bladed (flat and elongated) otherwise (see Figure 2.5b). So, according to Kong and Fonseca (2018)'s definition of flatness and elongation, particles featured by the same value of fl may feature different degrees of flatness, depending on their elongation values.



(a)



(b)

Figure 2.6: Indices of (a) elongation and (b) flatness proposed by Kong and Fonseca (2018).

Bagi and Orosz (2020) proposed the use of a surface orientation tensor to characterise the form of a particle as a solution to the problem of defining the main dimensions of irregular particles. The tensor is used to calculate elongation, flatness and compactness of the tessellated surface of a particle. In case of cuboidal particles, calculation of the surface orientation tensor leads to the following formulae for elongation, flatness and compactness:

$$\begin{aligned} el &= \frac{c}{b} - \frac{c}{a} \\ fl &= 1 - \frac{c}{b} \\ co &= \frac{c}{a} \end{aligned} \tag{2.6}$$

Contour maps of these indices produced on a Zingg plot are shown in Figure 2.7. From Figure 2.7a emerges that the el index takes low values for particles on the left (both upper left and lower left) of the Zingg plot. Although the particles on the upper left part of the plot are indeed flat and not elongated, the particles on the lower left (i.e. for $c/b < 0.2$ and $b/a < 0.2$) are particles that are in fact flat and elongated. However, most geomaterials do not exhibit such high elongation and flatness values. Nevertheless a merit of these indices is that they add up to unity ($el+fl+co = 1$), so they can be interpreted as percentile representations of a particle overall form, i.e. a real three-dimensional particle will always have some degree of elongation, flatness and compactness even if it is predominantly elongated, flat or compact. Moreover, for values at the extremes of the spectrum, one of the indices is enough to determine the class of the particle form, e.g. a flatness of 0.8 guarantees that a particle is flat, regardless of its values of compactness or elongation. However Bagi and Orosz (2020) did not attempt to employ these indices to improve on the current particle classification systems.

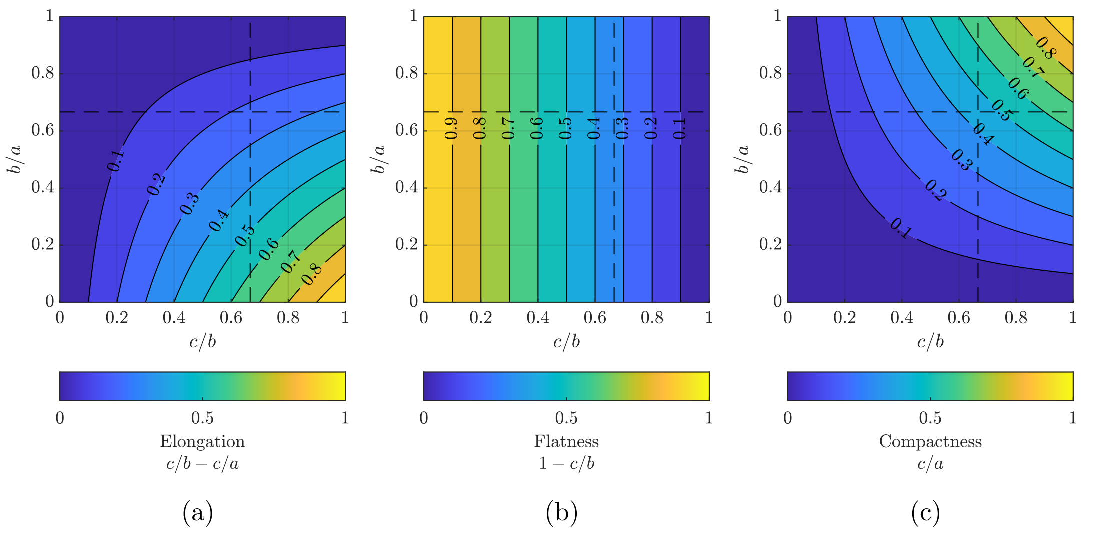


Figure 2.7: Indices of (a) elongation, (b) flatness and (c) compactness proposed by Bagi and Orosz (2020).

On the other hand Potticary et al. (2015) considered an equivalent scalene ellipsoid to define the main dimensions of irregular particles, arriving at a different definition for el, fl and co:

$$\begin{aligned} el &= \frac{a-b}{a+b+c} \\ fl &= \frac{2(b-c)}{a+b+c} \\ co &= \frac{3 \cdot c}{a+b+c} \end{aligned} \tag{2.7}$$

These indices can be seen as percentages of an overall form, while their distribution on a Zingg plot seems in agreement with the shape of the cuboids of varying aspect ratios of Figure 2.5b. With regard to very bladed particles (i.e. for $c/b < 0.2$ and $b/a < 0.2$), the indices show an opposite trend of the indices of Bagi and Orosz (2020), featuring high elongation and low flatness values for particles that are both flat and elongated. Contour maps of these indices produced on a Zingg plot are visible in Figure 2.8.

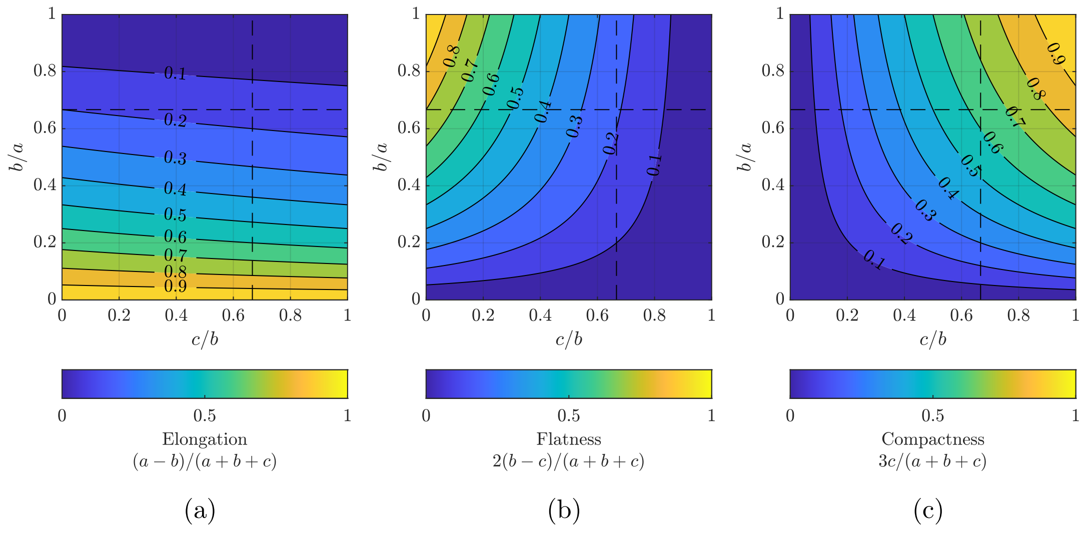


Figure 2.8: Indices of (a) elongation, (b) flatness and (c) compactness proposed by Potticary et al. (2015).

2.2.3 Effective form

The indices proposed by Potticary et al. (2015) belong to a small group of indices which have been shown to demonstrate a clear correlation with the shear strength of particulate materials at critical state, i.e. at a state of excessive shear deformation in which the material keeps deforming, without further increase in its internal stresses. Potticary et al. (2015) first demonstrated an almost linear trend of their flatness (or so called platyness) index with the tangent of the critical state friction angle, by conducting numerical triaxial compression tests of oblate spheroids (i.e. ellipsoids where their intermediate and long axes are equal) using the DEM, for particles of zero elongation and increasing flatness values. Potticary et al. (2016) conducted a similar study for prolate spheroids (i.e. ellipsoids where their small and intermediate axes are equal), focusing on particles with zero flatness and increasing elongation values, demonstrating that their elongation index also correlates almost linearly with the tangent of the critical state friction angle of the studied spheroids.

Combining these observations, Harkness and Zervos (2019) conducted triaxial compression tests for triaxial ellipsoids, i.e. particles that demonstrated both a degree of flatness and a degree of elongation, for various combinations of aspect ratios. Based on their analysis, they derived a new index called *effective form*, described by Equation 2.8, which correlates almost linearly to the tangent of the critical state friction angle of the studied ellipsoids. Figure 2.9 illustrates a contour map with values of this index for particles of all possible aspect ratios c/b and b/a, with the aid of a Zingg plot. This way of visualisation allows to explore how the values of the effective form vary for particles of various morphology types.

$$ef = \frac{a-c}{a+b+c} \tag{2.8}$$

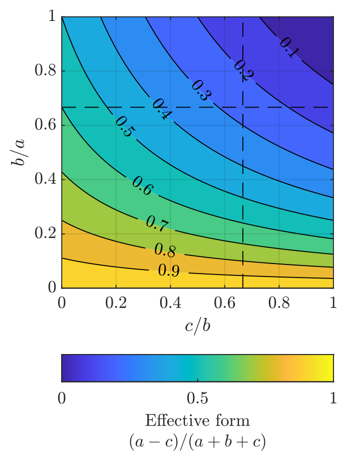


Figure 2.9: Contour plot of the effective form proposed by Harkness and Zervos (2019) on a Zingg plot.

Plotting the effective form on a Zingg plot makes apparent that the values of the effective form do not differentiate among the different regions of the Zingg classification system, i.e. a flat or an elongated particle can have the same effective form. Though, using the Zingg classification system, it can be inferred that a flat particle can take values of effective form within the range 0.15-0.6, a compact particle cannot have an effective form larger than 0.25, a bladed particle can have an effective form within the range 0.25-1.0, while for an elongated particle, the parameter can take values within the range 0.15-1.0. This happens, as the effective form of a particle can be seen as the sum of its elongation index plus half its flatness index, using the formulae of Potticary et al. (2015), i.e. $ef = el + fl/2$.

It is noteworthy, that the effective form shows roughly an inverse trend to the compactness index on the axes of a Zingg plot. Because of this, it can be argued that the critical state friction angle shows increased values for non-compact particles and lesser values for compact particles, which is also found by the analyses of Potticary et al. (2015, 2016); Harkness and Zervos (2019).

It should be noted that Figure 2.9 demonstrates high values of effective form -and thus of critical state shear strength- for very elongated particles, moderate values for flat particles and small values for compact particles. This distribution is true if particles stay undeformed and intact. When it comes to natural materials, such as silt and sand, very flat or very elongated particles (e.g. with $c/b \approx 0.01$ and $b/a \approx 0.01$, respectively) tend to demonstrate low survival rates, as in nature they fragment and evolve to more moderate particle forms. Buscarnera and Einav (2021) demonstrate this by introducing the concept of a "shape attractor", i.e. a moderate form type representing the shape of fragments which can evolve from initially flat, elongated, bladed or compact intact particles. The characterisation "moderate" is here used to describe a particle form that is not particularly flat, nor particularly elongated, nor particularly bladed, nor particularly compact. For instance, Buscarnera and Einav (2021) illustrate this point on a Zingg plot, near the cross-over point among the four classes of form, as defined by Zingg, highlighting a range within $0.65 < c/b \approx b/a < 0.8$ as the location of the "shape attractor" particle form.

2.3 New indices for elongation, flatness and compactness

A new set of indices to characterise particle elongation, flatness and compactness is here proposed. The indices (Equation 2.9) were designed to add up to unity and take 'reasonable' values for the whole range of particle aspect ratios, i.e. for c/b and b/a ranging within (0,1].

$$\begin{aligned} el &= \frac{a \cdot c}{a \cdot c + b^2} - \frac{c}{a + c} \\ fl &= \frac{b^2}{a \cdot c + b^2} - \frac{c}{a + c} \\ co &= \frac{2 \cdot c}{a + c} \end{aligned} \tag{2.9}$$

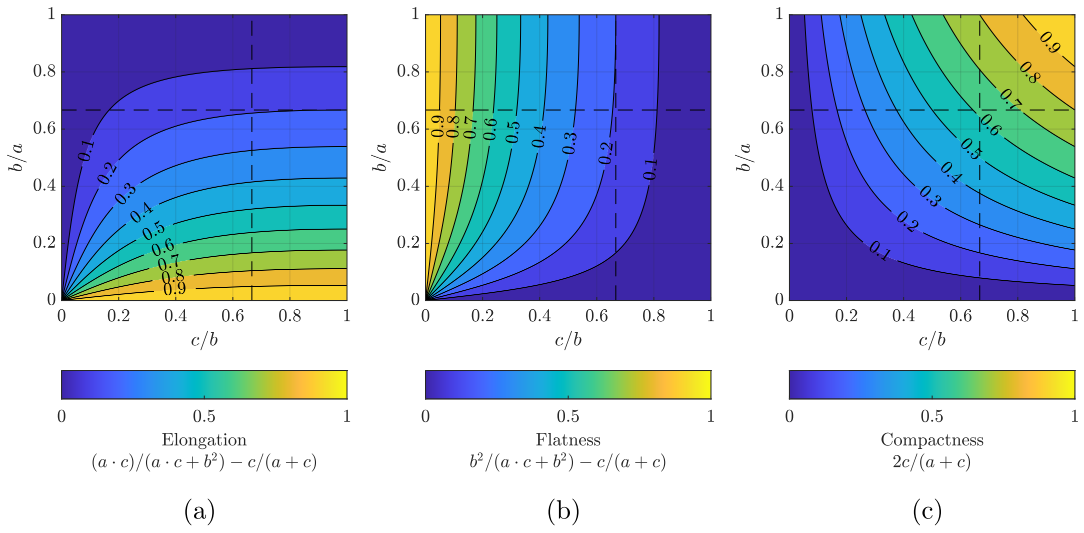


Figure 2.10: Indices of (a) elongation, (b) flatness and (c) compactness proposed in this study.

Mapping the indices on a Zingg plot (see Figure 2.10), makes it apparent that the new flatness and elongation indices are symmetric along the bisect line of the map, $c/b = b/a$, making their values easier to interpret. The proposed fl and el indices show a general agreement with the ones proposed by Bagi and Orosz (2020) and Potticary et al. (2015) in the regions I, II and IV of the Zingg plot (see Figure 2.5a), featuring flat, compact and elongated particles, respectively. Instead, this is not the case for the flatness and elongation values of particles falling in the bladed region of the Zingg classification system, i.e. $c/b < 2/3$ & $b/a < 2/3$ (lower left region bounded by dashed lines). For example, for particles featured by small c/b values the el index of Potticary et al. (2015) takes high values although they are predominantly flat. But their flatness is captured by the new proposed indices. For instance, a very flat particle is considered, featured by $c/b = 0.01$ & $b/a = 0.2$ (see the red circle in Figure 2.11d), meaning that its short axis c is 100 times smaller than the intermediate axis b, while the intermediate axis is only 5 times smaller than the long axis a. According to Potticary et al. (2015) $fl = 0.329$ & $el = 0.666$, whereas using the proposed new indices $fl = 0.950$ & $el = 0.046$. Moreover, for particles with $b/a = 0.2$ (see the green band in Figure 2.11d) their morphology becomes more elongated for increasing c/b values. This trend is well captured by the proposed fl and el indices whereas el of Potticary et al. (2015) takes nearly a constant value irrespective of the c/b values. Finally with regard to the proposed compactness index, it exhibits similar values to the ones taken by the indices of Bagi and Orosz (2020) and Potticary et al. (2015).

2.4 New classification system for particle form

As early as 1958, Sneed and Folk highlighted that the Zingg classification system underestimates the range of elongated and flat particles, while overestimates the range of bladed particles. The latter occupies the majority of the Zingg plot (Smalley, 1966) with a range of both flat and elongated particles misclassified as bladed (Sneed and Folk, 1958).

To address the current shortcoming of the Zingg classification system, a sought new classification system needs 1) to keep unchanged region II of the compact particles while 2) enlarging the regions of both flat and elongated particles to the expense of the region of the bladed ones. The first requirement, i.e. keeping region II unchanged, implies that the region needs to be delimited by the same boundaries as those of the Zingg classification system, i.e. $b/a = 2/3$ and $c/b = 2/3$. These values plugged into the new indices proposed in this chapter result into $fl = 0.2$ and $el = 0.2$. In Figure 2.11a the curves corresponding to $fl = 0.2$ and $el = 0.2$ are plotted as solid lines on a Zingg plot together with the modified regions I, II, III and IV of the new proposed classification system, where the boundaries between regions are now provided by $fl = 0.2$ and $el = 0.2$ instead of $b/a = 2/3$ and $c/b = 2/3$ (drawn as dashed lines in Figure 2.11a). Visually it is immediately evident that in the new system, regions I and IV corresponding to flat and elongated particles respectively are larger while the size of region III of the bladed particles is smaller, fulfilling the second aforementioned requirement. In Figure 2.11b and Figure 2.11c, ellipsoids and cuboids of various aspect ratios are superimposed on the plot of Figure 2.11a, so that some specific example cases of particles can be analysed in order to assess the accuracy of the new proposed classification system. Both ellipsoids and cuboids are visualised, to highlight that using the main particle dimensions to characterise particle form is not affected by roundness features. For example, a cuboid is considered, featured by $c/b = 0.6$ & $b/a = 0.2$, marked by a red circle in Figure 2.11d. This is clearly an elongated rather than a bladed particle as per the Zingg classification system. Second, a cuboid is considered, featured by $c/b = 0.01$ & $b/a = 0.2$. This is a predominantly flat particle rather than flat & elongated (i.e. bladed) as per the Zingg classification system. In summary, in the proposed new classification system and unlike the Zingg one, particles featured by $b/a < 2/3$ and small c/b values are correctly classified as flat and particles featured by $c/b < 2/3$ and small b/a values as elongated.

The proposed indices and classification system are well defined for particles of all possible aspect ratios ($c/b > 0$ and $b/a > 0$). Focusing on the lower-left region of a Zingg plot, which corresponds to particles with $c/b < 0.1$ and $b/a < 0.1$, three particle shapes will be considered: (i) a flat particle with $c/b = 0.01$ & $b/a = 0.1$, (ii) a blade-like particle with $c/b = 0.1$ and $b/a = 0.1$ and (iii) an elongated particle with $c/b = 0.1$ and $b/a = 0.01$, aiming to demonstrate the robustness of calculations for three extreme particle shapes. Cuboids of these aspect ratios are shown in Figure 2.12, where the morphological difference among these particles becomes apparent. Moreover, it should be noted that most natural materials, such as sands or silts, do not feature such extreme particle shapes, as these tend to break in nature, and evolve into smaller, more regular fragments (Buscarnera and Einav, 2021).

2.5 Comparison of form classification systems

Several classification systems have been proposed in the literature to categorise particles based on their form. The system of Zingg has become standard practice due to its simplicity, although as demonstrated in the previous section, it misclassifies very flat and very elongated particles as bladed, while particles with the same aspect ratio c/b or b/a can feature significantly different shapes, making the interpretation

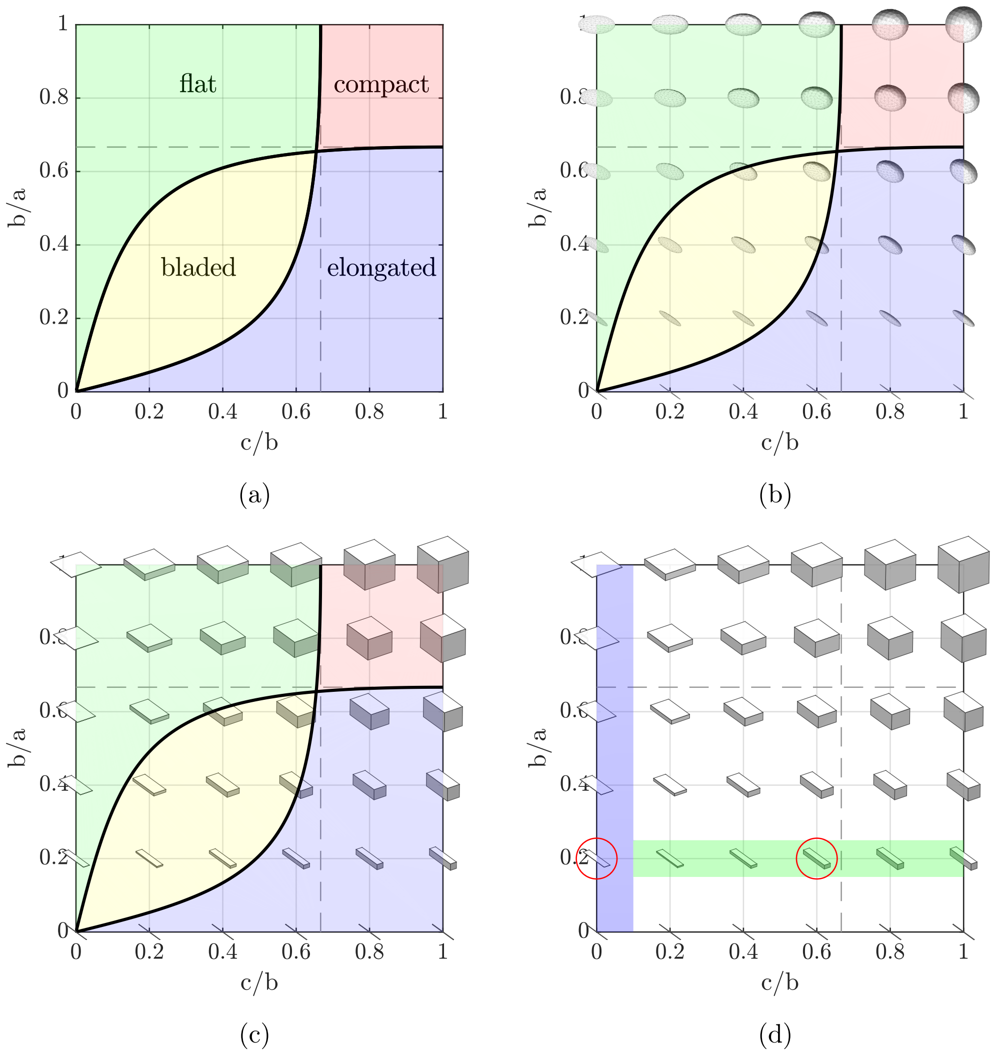


Figure 2.11: (a) The proposed classification system with the solid lines denoting the boundaries among bladed, elongated, compact and flat particles. The dashed lines denote the Zingg classification system; (b),(c) ellipsoids and cuboids with varying aspect ratios plotted over the newly proposed and Zingg's classification systems; (d) particles in the coloured bands and red circles were employed in the comparison exercise between form indices.

of these indices less intuitive.

Blott and Pye (2008) proposed an updated Zingg classification system, which entails a finer discretisation of the aspect ratio domain ($c/b - b/a$) to 25 categories of particle form, with the boundaries between categories being determined by the values of c/b and b/a for intervals of 0.2. Still, this more sophisticated version of the Zingg classification system suffers from the same limitations as the original Zingg system, when it comes to very flat and very elongated particles.

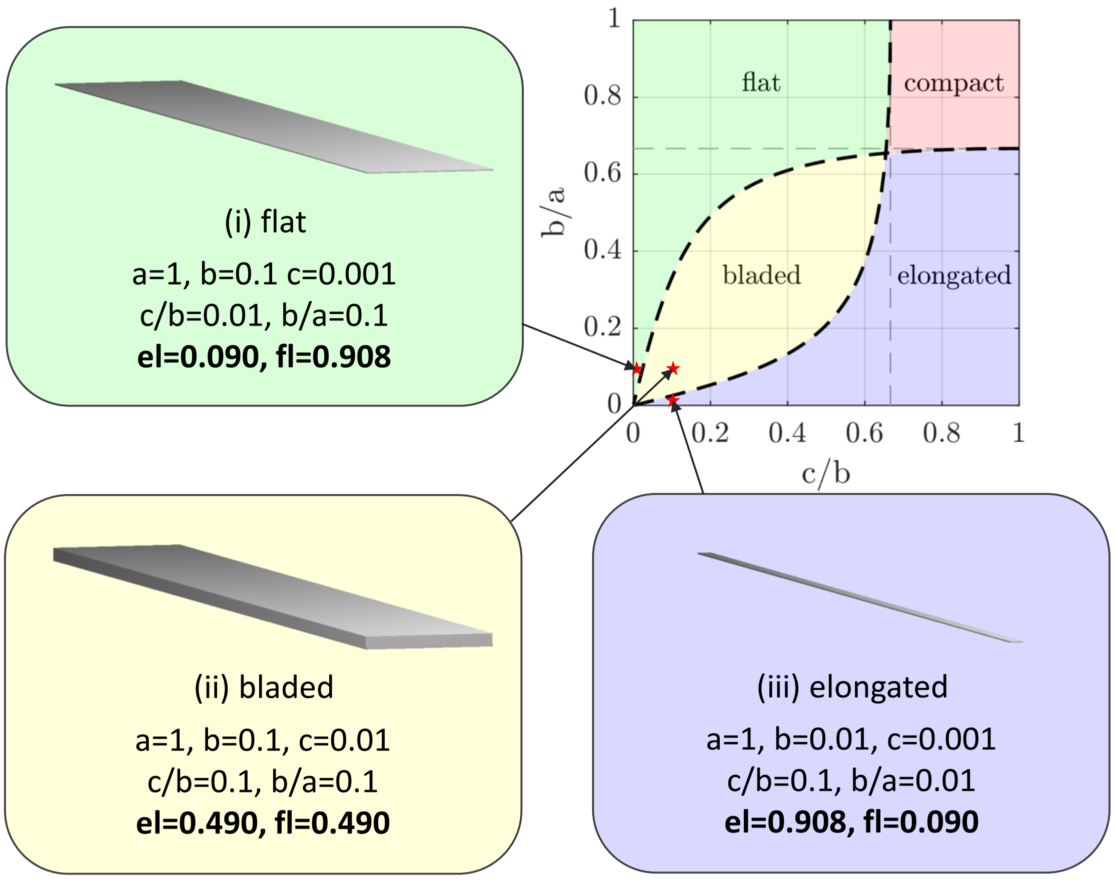
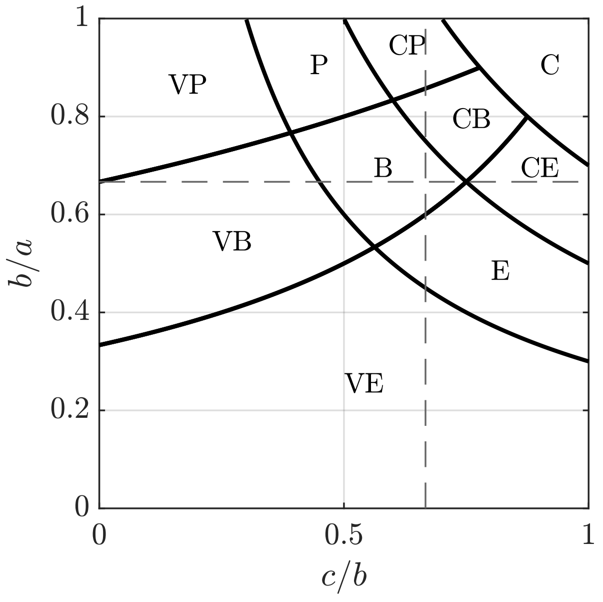


Figure 2.12: Classification of three cuboids with extreme aspect ratios using the proposed formulation.

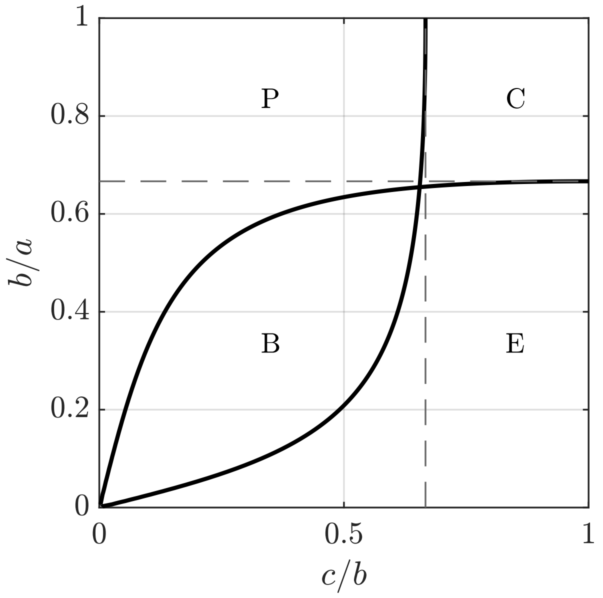
A more refined classification system was proposed by Sneed and Folk (1958), which considers ten classes of particle form: compact, compact platy, compact bladed, compact elongated, platy, bladed, elongated, very platy, very bladed, very elongated. The shape classes of Sneed and Folk (1958) are mapped on a Zingg plot, where it becomes evident that there is similarity between the trends of the proposed compactness and the compactness of Sneed and Folk (1958), which was equal to c/a, same as the compactness proposed by Bagi and Orosz (2020) considering the surface orientation tensor of cuboidal particles.

Figure 2.13 demonstrates the classification system of Zingg (1935) along with the proposed one, produced on a Zingg plot, to ease comparisons. The classification system of Sneed and Folk (1958) exhibits significant differences with the proposed system and with the system of Zingg (1935), although it agrees with the proposed system when it comes to classifying particles with low b/a values as elongated.

Figure 2.14 shows the Sneed and Folk classification system in its usual form,



(a)



(b)

Figure 2.13: (a) The classification system of Sneed and Folk (1958), visualised on a Zingg plot. The notations stand for C: Compact, P: Platy, B: Blade, E: Elongated, V: Very; (b) the classification system proposed in this study. The characterisation 'platy' is here considered interchangeable with the term 'flat'.

plotted on a ternary plot, where the parameters controlling the classes become visible. In particular, the parameter c/a classifies particles based on their degree of compactness at values $c/a = 0.3$, $c/a = 0.5$ and $c/a = 0.7$, while the parameter $(a - b)/(a - c)$, termed as the "disc-rod index" by Illenberger (1991), classifies particles of low compactness as platy (for values $(a - b)/(a - c) < 1/3$), bladed $(1/3 < (a - b)/(a - c) < 2/3)$ and elongated $((a - b)/(a - c) > 2/3)$. All particles with high compactness $(c/a > 0.7)$ are considered compact, regardless of their disc-rod index value.

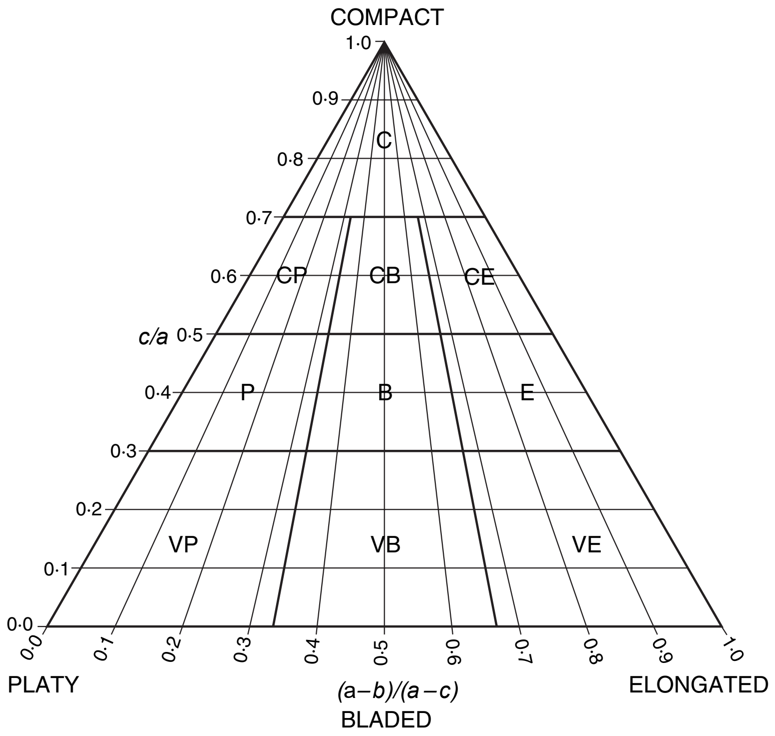


Figure 2.14: The shape classification system of Sneed and Folk (1958) (modified from Blott and Pye (2008) with permission from Wiley & Sons). The notations stand for C: Compact, P: Platy, B: Blade, E: Elongated, V: Very.

The proposed system classifies particle form in terms of flatness (fl) and elongation (el), while compactness (co) is not used directly, as only two of the proposed fl, el and co indices are independent, since $fl + el + co = 1$. A deficiency of the system is identified, as particles with the same value of compactness can be classified in different classes, e.g. for $co = 0.75$, a particle can be flat if $fl = 0.2$ and $el = 0$ (i.e. for $c/b = 0.6$ and $b/a = 1.0$), compact if $fl = 0.125$ and $el = 0.125$ (i.e. for $c/b = 0.775$ and $b/a = 0.775$) or elongated if $fl = 0.0$ and $el = 0.25$ (i.e. for $c/b = 1.0$ and

$b/a = 0.6$). This reveals a limitation of the proposed classification system, as particles with $co > fl$ can be classified as flat, and particles with $co > el$ can be classified as elongated, rather than compact. Sneed and Folk (1958) avoided this limitation by introducing more classes in their system, allowing for combination of features, such as *compact-platy*, *compact-bladed* and *compact-elongated* particles. This could be a way forward for the improvement of the future development of the proposed system, as at the moment the system is in close agreement with the widely-accepted classification system of Zingg (1935) at these regions of high compactness, as seen in Figure 2.13b, which also considers four classes. Determining exact boundaries between classes for any classification system is not trivial, as objective criteria do not exist to define such boundaries, and the design of the system depends on the experience and perspective of the researcher designing the system.

2.6 Concluding Remarks

In this chapter, the most popular sets of indices from the literature employed to classify particle form were critically reviewed with the aid of a Zingg plot, where all possible combinations of aspect ratios are represented.

New indices for elongation, flatness and compactness were proposed in the attempt to overcome the limitations identified for the indices currently in use. The newly proposed elongation and flatness indices give rise to Zingg plots which are symmetric along the bisect line of the plot, making their values easier to interpret. Also, they take values complementary to unity so they can be interpreted as percentages of an overall form. Examples of cuboids are analysed to illustrate the appropriateness of the indices to describe particle form.

Finally, a new particle classification system derived from the proposed indices which overcomes the misclassification of a range of particles that are incorrectly classified as bladed in the Zingg system, is illustrated. Overall, the proposed indices combine merits of all the other demonstrated sets of indices, as:

- They add up to unity, allowing them to be seen as percentages of an overall form, thus making their values intuitive and easy to follow.
- The distribution of their values of a Zingg plot allows for them to be used in a classification system, as in improvement of the Zingg system.
- They provide reasonable fl, el values for very elongated and very flat particles.

Chapter 3

SHape Analyser for Particle Engineering (SHAPE): Seamless characterisation and simplification of particle morphology from imaging data

3.1 Introduction

The mechanical and rheological behaviour of particulate and granular assemblies is significantly influenced by the shape of their individual particles. This chapter presents SHAPE (Angelidakis et al., 2021b), a code to implement shape characterisation of three-dimensional particles in an automated and rigorous manner, allowing for the processing of samples composed of thousands of irregular particles within affordable time runs. The input particle geometries can be provided in one of the following forms: segmented labelled images, three-dimensional surface meshes, tetrahedral meshes or point-clouds. These can be complemented with surface texture profiles. Shape characterisation is implemented for three key aspects of shape, namely surface roughness, roundness and form. Also, simplified particle shapes are generated by the code which can be used in numerical simulations to characterise the mechanical behaviour of particulate assemblies, using numerical approaches such as the Discrete Element method and Molecular Dynamics. Combining these two features in one automated framework, the code allows not only to characterise the original granular material but also to monitor how its morphological characteristics change as the shape of the particles is simplified according to the chosen fidelity level for the application of interest.

3.2 Motivation and significance

Particulate and granular materials are omnipresent in nature, industry and day-to-day life. They play a crucial role in powder handling (Alizadeh et al., 2017), additive manufacturing (Nan et al., 2018), landslide hazard assessment (Boon et al., 2015b) and rock avalanche modelling (Mollon et al., 2015), but to name a few. The vast majority of particles are featured by a variety of irregular shapes, far from any idealised geometries. Their morphological characterisation is of interest to a plethora of research disciplines which study the behaviour of particulate matter, namely Physics (e.g. Nguyen et al., 2014; Domokos et al., 2015), Civil Engineering (e.g. Santamarina and Cho, 2004), Pharmaceutical Engineering (e.g. Champion et al., 2007; Gamble et al., 2015; Hare et al., 2018), Chemical Engineering (e.g. Hodges et al., 2010), Agricultural Engineering (e.g. Pasha et al., 2016) and Geosciences (e.g. Williams and Caldwell, 1988; Blott and Pye, 2008). Several studies of particulate materials have demonstrated the significant influence of particle morphology (i.e. form, roundness and roughness) on the static and dynamic mechanical behaviour of assemblies of particles such as shear banding (e.g. Iwashita and Oda, 1999; Kawamoto et al., 2018), jamming (e.g. Nan et al., 2018), flowability (e.g. Pasha et al., 2016) and processability (e.g. Shah et al., 2017).

In this study, particle morphology is categorised using three independent aspects of shape, namely form, roundness and roughness, as defined in Barrett (1980). Form is a first order morphological property, reflecting the relative proportions of a particle. Roundness is a second-order aspect of particle morphology, related to the sharpness of corners and edges, and is therefore appended on top of the form features. Surface roughness is a third-order aspect of shape, related to asperities which are appended on top of roundness features. It should be noted that roughness is scale dependent, so the level at which this aspect of shape is measured and studied depends on the problem of interest.

Regarding the characterisation of particle form, the classification system proposed by Zingg (1935) is being predominantly used in both research and practice, mainly due to its simplicity. Based on two independent ratios of the three main dimensions of the particle, Zingg classified the shape of pebbles in four distinct categories, namely: oblate (or flat), compact (or equant), prolate (or elongated) and blade-like (or triaxial). The ratios of these dimensions -short (S), intermediate (I) and long (L)- are used in the Zingg plot of Figure 3.5b). The ratio S/I is related to how flat a particle is, while the ratio I/L is related to how elongated it is. According to the Zingg classification system: a particle is oblate if $S/I < 2/3$ and $I/L > 2/3$; a particle is compact if $S/I > 2/3$ and $I/L > 2/3$; a particle is blade-like if $S/I < 2/3$ and $I/L < 2/3$; a particle is prolate if $S/I > 2/3$ and $I/L < 2/3$.

Particle shape characterisation has been an active research topic since the beginning of the 20^{th} century (Wentworth, 1919; Wadell, 1932). Traditionally, a set of shape indices were employed to describe the main morphological aspects of particulate materials, mostly in 2-D (Powers, 1953). However, the results of 2-D shape analyses are largely influenced by the plane chosen for the projection of the real particle (Fonseca et al., 2012; Nadimi and Fonseca, 2017a). In the last two decades, the field has made important progress thanks to new experimental techniques, e.g. micro Computed Tomography (µCT) (Nadimi and Fonseca, 2018; Mehrabi et al., 2021), laser scanning, white light interferometry and more advanced algorithms combining 2-D images to reconstruct 3-D particle geometries, which in general allowed more precise particle shape analyses. On the other hand, this progress has made apparent the limitations of the indicators proposed in the 20th century, so, that several new indicators have been proposed in the last decade (Clayton et al., 2009; Altuhafi et al., 2016). At the same time, no consensus has yet been reached in the scientific community about a set of universally accepted shape indicators to fully characterise a particle shape. Therefore, a software able to calculate the main indicators in the literature for particles imaged from a variety of experimental techniques is a potentially transformative tool since it provides the community with the opportunity to estimate shape indicators efficiently and conveniently for current and future datasets of particles. In this way, researchers will be able to identify over time the best set of shape indicators for their specific particulate material of interest. Also, this software has the potential to significantly benefit industry, which still heavily relies on 2-D image analyses (British Standards Institution, 2006, 2008, 2012, 2014).

The numerical modelling of particulate materials for engineering applications is currently dominated by the conventional Discrete Element Method (DEM) with particles modelled as spheres, since detecting contact is mathematically and computationally vastly easier, compared to non-spherical particles. Nevertheless, various approaches have been proposed to model real 3-D particles accounting for their shape, the main ones being: polyhedral particles (Cundall, 1988; Hart et al., 1988), superquadrics (Williams and Pentland, 1992), clusters of spheres held together (Jensen et al., 1999; Garcia et al., 2009) or Non-Uniform Rational Basis-Splines (NURBS) as in Andrade et al. (2012). In principle, any particle can be discretised as a polyhedron and the more complex the shape, the higher the number of faces required for a faithful representation. A new mathematical formulation extending the concept of potential particle (Houlsby, 2009) to 3-D polyhedra (Boon et al., 2012, 2013) allows the efficient DEM modelling of hundred thousands of particles of any convex shape (Gardner et al., 2017). In this framework, a concave particle can also be simulated as a cluster of convex polyhedra linked by unbreakable bonds (Boon et al., 2015b). For the successful modelling of a particulate material, the right balance between computational efficiency and geometric and mechanical accuracy has to be struck. For instance, the geometry of a single particle, imaged by a laser scanner, can be very accurately captured by thousands of faces, but if a DEM simulation of some thousand particles was attempted, the computational time would be unaffordable. Therefore, in the context of DEM analysis of particulate matter, particle shapes need to be simplified, so that the numerical simulation of interest can be performed without compromising the representativeness of the sample, i.e. the mechanical behaviour of the numerical sample still needs to be a faithful replica of the real sample in terms of the experimental response observed.

Various mathematical techniques can be used to simplify the morphology of a particle. Zhou et al. (2015) demonstrated the capacity of Spherical Harmonics, a generalisation of the Fourier transformation in the 3D space, to generate particles of various fidelity levels. Ouhbi et al. (2017) used statistical techniques and the Proper Orthogonal Decomposition in particular, which is a variant of Principal Component Analysis, to generate particles of controlled morphological accuracy. In addition, mesh-reduction techniques, such as edge-collapse, can be employed to generate simplified particle morphologies. Comparisons among these methods are currently missing from the existing literature.

The proposed code is expected to be of interest to the community, since it can be used to link particle shape to mechanical behaviour, through the calculation of various shape indices, not only for the original material but also for the simplified particles it generates and which can be used in numerical analyses. To achieve this, a down-sampling of the original particle geometries is proposed in this section, at discrete resolution intervals, i.e. fidelity levels, with simultaneous monitoring of the inescapable alteration of some shape features, to ensure the preservation of the main morphological profile.

This approach ensures that the selected particle shapes are not oversimplified and thus that they retain an association to the real particles, morphologically-wise. With this, the modeller can make an informed choice on the trade-off between accuracy and computational speed for numerical simulations of particulate materials. A variety of shape descriptors is available, while outputs are provided in multiple formats, compatible with the syntax of some widely used DEM and FEM codes.

3.3 Software description

The code comprises two main modules, implementing particle shape characterisation and particle shape simplification, with the latter aiming to generate polyhedral particles for numerical simulations. An additional auxiliary module provides utility functions, calculating geometric properties of the polyhedral particles and performing basic geometric transformations.

3.3.1 Software architecture

`SHAPE` provides a particle shape characterisation routine for all aspects of particle shape, namely surface roughness, roundness and form (Barrett, 1980). In addition, the code includes a module to generate simplified geometries of three-dimensional particles derived from imaging data, for different levels of simulation fidelity. The particle geometries are analysed using the particle shape characterisation module, allowing the user to prescribe the level of particle simplification on the basis of target shape descriptor values. `SHAPE` can work for large assemblies of particles, providing a fully automated framework from digital imaging to numerical simulation.

Figure 3.1 shows the main architectural features of `SHAPE`. Available imaging data of a particle can be inserted in the code either as a point cloud, a surface or tetrahedral mesh (Micó, 2020), or a segmented volumetric image. If a point cloud is given as input, the particle surface is reconstructed either using the Delaunay triangulation algorithm or using the Crust method, proposed in Amenta et al. (1998) and implemented in Matlab by Giaccari (2020). If a segmented volumetric image is used, it is transformed into a surface mesh using an implementation of the refined Delaunay triangulation algorithm developed in The CGAL Project (2020), which is provided by the `Iso2Mesh` code (Fang and Boas, 2009). In particular, the functions `surf2vol` and `vol2surf` of `Iso2Mesh` are used for transformations between surface and voxelated representations of the particles.

Geometrical characteristics of the particle can be calculated, such as its centroid, volume, surface area inertia tensor, radius of its largest possible inscribed sphere and radius of its smallest bounding sphere. Shape characterisation follows, which is detailed in the next section. If only characterisation is of interest, outputs of the shape analysis can be extracted and statistics can be provided for the original geometry of the material at this point.

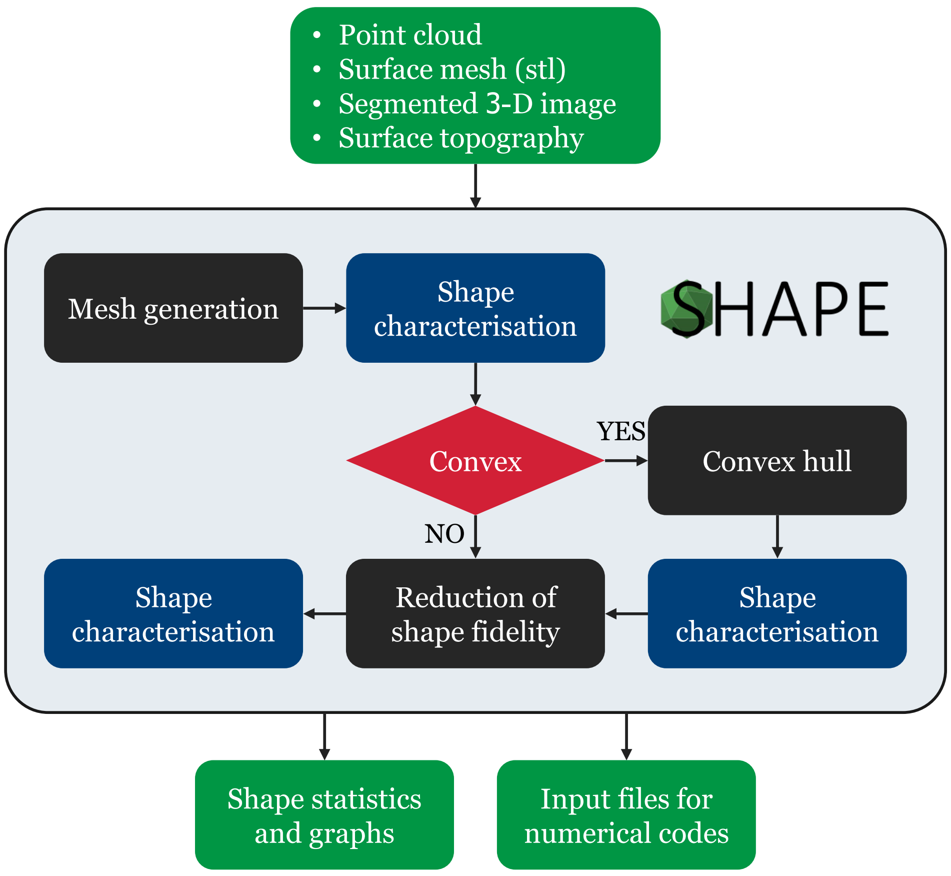


Figure 3.1: Main architecture of SHAPE.

What comes next is a shape simplification procedure, employing mesh-reduction techniques. The user can choose at this stage whether the simplified particle needs to be convex, and the convex hull of the particle is employed in such a case. Preserving geometrical characteristics during the simplification of a particle can be of interest. To this end, `SHAPE` offers the option to apply an isochoric transformation of the simplified particle, in order to match the volume of the original one. The isochoric transformation operates a homothetic scaling of the simplified particle, multiplying the coordinates of its vertices by a scale factor of $\sqrt[3]{V_{original}/V_{simplified}}$, where V represents the volume of the particle for both the original and the simplified fidelity levels.

Applying this scaling law on the simplified geometry preserves the volume of the particle across fidelity levels, albeit the same is not achieved for the surface area nor the inertia tensor of the particle. It should be noted, that scaling the particle homothetically to achieve an overall isochoric transformation from the input object to the simplified particle does not affect the parameters related to shape characterisation, like sphericity, convexity, flatness and elongation, but it does change the size of the particle, so the user should be careful in prescribing scale factors which might deviate significantly from unity, as the particle size distribution may be affected.

Then, simplified particles are generated using the `Iso2Mesh` code (Fang and Boas, 2009), which includes binaries of The CGAL Project (2020) and `TetGen` (Si, 2015) to implement mesh manipulations, including mesh-reduction. It should be noted that only some mesh-generation and editing tasks are outsourced in `Iso2Mesh` and this external dependency is not used to its full extent.

The outputs of the code include shape characteristics for each particle and statistics of their values if a whole sample is analysed. Additionally, the simplified particle geometries are exported in various formats, supported by some of the most widely used FEM and DEM codes.

3.3.1.1 Shape characterisation

To date, particle shape characterisation is not a straightforward procedure. A plethora of indices (or so-called descriptors) exist, aiming to measure some basic aspects of shape, which often provide contradicting results if compared. It becomes evident that the outputs of such a characterisation are highly dependent on the definition of the chosen indices. To this end, `SHAPE` calculates a variety of widely accepted indices, aiming to offer a comparison among their values and provide the user with an integrated view over the morphological characteristics of the material of interest.

In many studies, shape characterisation is often limited to the form of the particles, while in reality the importance of roundness and surface roughness are reported to be influencing the mechanical behaviour as well (Otsubo et al., 2017; Yang et al., 2016; Harkness and Zervos, 2019; Nadimi et al., 2019; Marzulli et al., 2021). Roundness can be seen here as the complementary percentage of angularity. `SHAPE` supports two definitions of roundness and two definitions of angularity.

In cases where the particle size is relatively small and its morphology is derived through laser scanning or computed tomography, the available imaging resolution is not satisfactory to measure its surface roughness. It is common practice to achieve this by employing tools like white light interferometry. `SHAPE` calculates five indices related to surface roughness (texture), based on available data of the surface profile.

Surface roughness data are given in the format of a point cloud. The input can contain some larger local features (depending on the size of the region of interest), which are related to the roundness or form of the particle. These features can be decoupled from roughness through the application of filtering, to remove the smaller curvatures (corresponding to the largest radii), associated to roundness and form. In this filtering process, it is integral to choose a cut-off wavelength wisely, in order not to filter out features related to roughness. To this end, Li et al. (2021) demonstrate a method to estimate the cut-off wavelength via measurements of roughness, which can be used to decouple roughness from roundness.

Figure 3.2 demonstrates the main structure of the shape characterisation module. A list of all the supported shape indices can be found in Table 3.1.

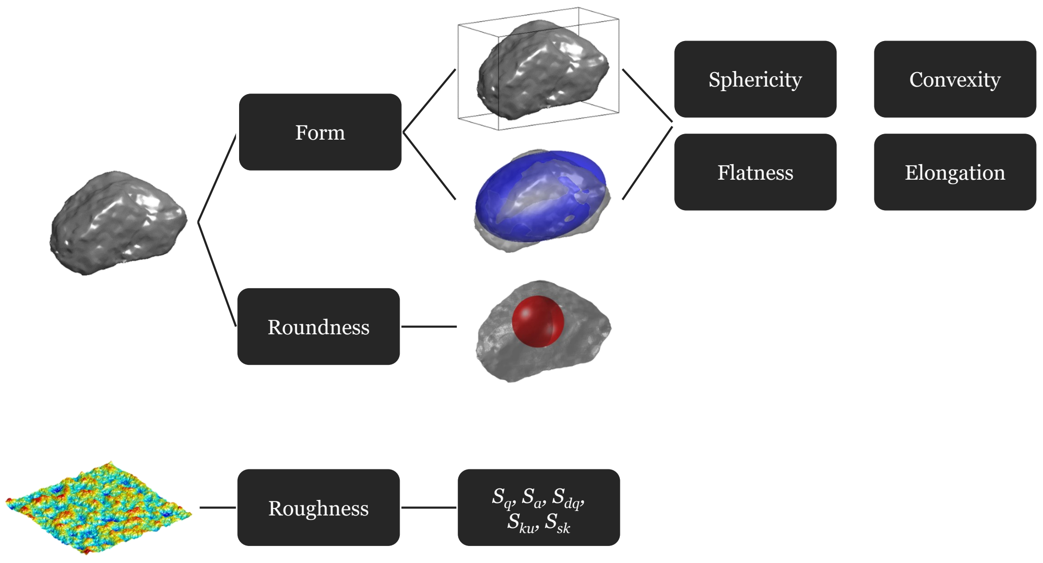


Figure 3.2: Shape characterisation module.

Table 3.1: Supported shape descriptors.

Shape descriptor	**Formula**	**Range**	**Parameters/Comments**
Form			
Convexity	$\frac{V}{V_{CH}}$	$[0,1]$	V_{CH}: Volume of convex hull
Degree of true sphericity (Wadell, 1932)	$\frac{\sqrt[3]{36\pi V^2}}{A}$	$[0,1]$	V: Volume A: Surface area
Intercept sphericity (Krumbein, 1941)	$\sqrt[3]{\frac{IS}{L^2}}$	$[0,1]$	S: Short axis
Flatness (Zingg, 1935)	$\frac{S}{I}$	$[0,1]$	I: Intermediate axis L: Long axis
Elongation (Zingg, 1935)	$\frac{I}{L}$	$[0,1]$	
Flatness (Kong and Fonseca, 2018)	$\frac{I-S}{I}$	$[0,1]$	
Elongation (Kong and Fonseca, 2018)	$\frac{L-I}{L}$	$[0,1]$	
Flatness (Potticary et al., 2015)	$\frac{2(I-S)}{L+I+S}$	$[0,1]$	
Elongation (Potticary et al., 2015)	$\frac{L-I}{L+I+S}$	$[0,1]$	
Flatness (Bagi and Orosz, 2020)	$\frac{f_1-f_2}{f_1}$	$[0,1]$	f_1, f_2, f_3: Eigenvalues of surface orientation tensor
Elongation (Bagi and Orosz, 2020)	$\frac{f_2-f_3}{f_1}$	$[0,1]$	
Compactness (Bagi and Orosz, 2020)	$\frac{f_3}{f_1}$	$[0,1]$	
Flatness (Angelidakis et al., 2022)	$\frac{I^2}{S\cdot L+I^2}-\frac{S}{S+L}$	$[0,1]$	
Elongation (Angelidakis et al., 2022)	$\frac{S\cdot L}{S\cdot L+I^2}-\frac{S}{S+L}$	$[0,1]$	
Compactness (Angelidakis et al., 2022)	$\frac{2S}{S+L}$	$[0,1]$	
Roundness			
Roundness (Wadell, 1932)	$\frac{\left(\sum_{i=1}^{N} r_i\right)/N}{R_{in}}$	$[0,1]$	r_i: Radius of corner i N: Number of corners
Roundness (Wentworth, 1919)	$\frac{r}{R_{in}}$	$[0,1]$	r: Radius of sharpest corner R_{in}: Inradius
Angularity (Harkness and Zervos, 2019)	$\frac{V\cap V_{el}}{V_{el}}$	$[0,1]$	V_{el}: Volume of fitted ellipsoid
Angularity (Kong and Fonseca, 2018)	$\frac{\sum A_j\cdot\max\left[0,\operatorname{sign}\left(k_{m,j}-k_{in}\right)\right)}{\sum A_j}$	$[0,1]$	$k_{m,j}$: Mean curvature of triangle j; $k_{in}=1/R_{in}$
Roughness			
Root mean square height (S_q)	$\sqrt{\frac{1}{A}\iint\limits_A z^2(x,y)\,\mathrm{d}x\,\mathrm{d}y}$	$[0,+\infty)$	Units of length
Arithmetical mean height (S_a)	$\frac{1}{A}\iint\limits_A \lvert z(x,y)\rvert\,\mathrm{d}x\,\mathrm{d}y$	$[0,+\infty)$	Units of length
Root mean square gradient (S_{dq})	$\sqrt{\frac{1}{A}\iint\limits_A\left[\left(\frac{\partial z(x,y)}{\partial x}\right)^2+\left(\frac{\partial z(x,y)}{\partial y}\right)^2\right]\mathrm{d}x\,\mathrm{d}y}$	$[0,+\infty)$	Unitless
Kurtosis (S_{ku})	$\frac{1}{{S_q}^4}\left[\frac{1}{A}\iint\limits_A z^4(x,y)\,\mathrm{d}x\,\mathrm{d}y\right]$	$[0,+\infty)$	Unitless
Skewness (S$_{sk}$)	$\frac{1}{{S_q}^3}\left[\frac{1}{A}\iint\limits_A z^3(x,y)\,\mathrm{d}x\,\mathrm{d}y\right]$	$(-\infty,+\infty)$	Unitless

3.3.2 Software functionalities

`SHAPE` is meant to provide an integrated framework which links particle shape characterisation with the generation of polyhedral geometries in numerical simulations, supporting the creation of simplified particles at measurable fidelity levels affordable by some of the most popular state-of-the-art numerical solvers. Its main functionalities can be summarised as particle shape characterisation of 3-D particles for all shape aspects and generation of simplified polyhedral geometries.

The supported output formats for the simplified particles are compatible with the following numerical tools:

- the DEM solver YADE (Smilauer et al., 2021), supporting the formats of the particle classes `Polyhedra`, `PotentialBlock` and `PotentialParticle`;
- the FEA solver Abaqus (2014), supporting the `C3D4` element type;
- the DEM solvers 3DEC (Itasca Consulting Group, Inc., 2016) & PFC 6.0 (Itasca Consulting Group, Inc., 2018);
- the DEM solver LMGC90 (Dubois and Jean, 2003);
- the DEM solver BlazeDEM (Govender et al., 2016).

In addition, several utility functions are provided, calculating some key geometric parameters, detailed in Table 3.2. Last, a set of functions is available, dedicated to provide graphs and statistics of the particle shape characterisation analysis of a sample.

3.3.3 Sample code snippet

The morphological analysis of large samples, composed of hundreds or thousands of particles, necessitates the implementation of a robust data structure, to make post-processing of the results easier. To this end, `SHAPE` has been designed following an object-oriented structure, briefly demonstrated in Figure 3.3.

Table 3.2: Geometric parameters.

Geometric parameter	**Method of calculation**
Volume	Discretisation to tetrahedra
Centroid	Discretisation to tetrahedra
Surface area	Sum of the areas of the triangles comprising the particle surface
Principal inertia tensor & principal axes	Superposition of inertia tensor for each tetrahedron and eigenvalue analysis to calculate the principal tensor and axes
Axis-aligned bounding box (AABB)	Extreme coordinates for the current particle orientation
Fitted ellipsoid (Petrov, 2020)	Least squares fit to point cloud
Oriented bounding box (OBB)	Two options: 1. Principal component analysis, using Singular Value Decomposition 2. Heuristic calculation, with options whether to calculate the bounding box with minimum volume, surface area or total length of edges (Korsawe, 2020)
Particle dimensions (Short, Intermediate, Long)	Two options: 1. Axes of the least squares fitted ellipsoid 2. Axes of the oriented bounding box
Maximum inscribed sphere	The particle geometry is transformed into a volumetric image, where the maximum Euclidean distance is equal to the inradius
Minimal bounding sphere (Semechko, 2020)	Using Welz's and Ritter's algorithms
Mean and Gaussian curvatures of surface mesh	Following a methodology for triangular surface meshes (Dong and Wang, 2005)

```
Particle % e.g. 1, 2, 3, etc.
  Particle_type % e.g. Original, Convex_hull, Face_No_100, Face_No_50, etc.
    Mesh % Surface_mesh, Tetrahedral_mesh, Voxelated_image, Surface_texture
    Auxiliary_geometries % AABB, OBB, Fitted_ellipsoid, Minimal_bounding_sphere,
         Maximal_inscribed_sphere
    Geometrical_features % Volume, Centroid, Surface_area, Current_inertia_tensor,
         Principal_inertia_tensor, Principal_orientations
    Morphological_features % Form, Roundness, Roughness
```

Figure 3.3: Object-oriented structure of `SHAPE`.

3.4 Illustrative examples

Example 1: Single grain simplification

Figure 3.4 shows the reduction of shape resolution of a real soil grain for five fidelity levels, using the simplification module of the code. The simplified shapes in this paradigm are created starting from the convex hull of the original particle, meant to be used by DEM codes, where the particles may need be convex. Fidelity is quantified in this particular case in terms of number of triangular faces of the mesh constituting the particle surface, while in a wider perspective, the user can choose a different measure of simulation fidelity to represent the detail of resolution of the particles.

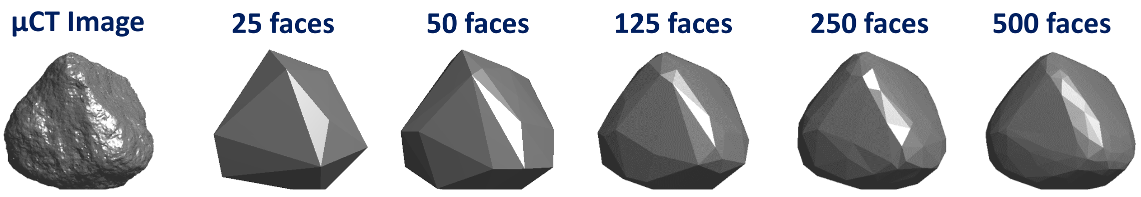


Figure 3.4: Simplification module: Reduction of shape resolution for five user-defined fidelity levels.

Table 3.3 presents various shape parameters for the original fidelity level and for simplified levels of 25, 50, 125, 250 and 500 triangular faces per particle. The high convexity value of the particle at its original fidelity level (equal to 0.933) indicates that considering its convex hull as the starting point of the shape simplification process can lead to representative simplified shapes, not very different from the original one. Small discrepancies are observed for the degree of true sphericity (less than 8%) and the shape indices proposed by Bagi and Orosz (2020) (less than 25%) across fidelity levels, which ensure that the simplified particles retain a high level of similarity to the original particle from a morphological standpoint. To further examine this, more shape indices were calculated in `SHAPE`, including the intercept sphericity (Krumbein, 1941), and the form indices of Kong and Fonseca (2018), Potticary et al. (2015) and Angelidakis et al. (2022). These indices rely on the three main particle dimensions, which are in this case calculated both using a minimal-volume oriented bounding box and a fitted ellipsoid, in an effort to assess the sensitivity of calculations that take place in these approaches. In both cases, all indices demonstrate relatively small differences between the original and the simplified fidelity levels, with the lower fidelity levels exhibiting larger discrepancies. For this sand particle, the fidelity level of 50 faces exhibits errors in the region of 25% for most of the analysed form indices, compared to the original fidelity level. Regarding usage of these particles in numerical simulations, e.g. using the DEM, the fitness of a simplified particle to be considered representative of the real material depends on the error of shape indices that is considered acceptable by the modeller.

Table 3.3: Characterisation of particle form of sand particle for various fidelity levels.

Shape index	**Fidelity level**					
	Original	**25 faces**	**50 faces**	**125 faces**	**250 faces**	**500 faces**
Convexity	0.933	1.000	1.000	1.000	1.000	1.000
Degree of true sphericity	0.871	0.901	0.915	0.928	0.933	0.934
Flatness (Bagi and Orosz, 2020)	0.144	0.126	0.163	0.158	0.154	0.152
Elongation (Bagi and Orosz, 2020)	0.116	0.145	0.122	0.127	0.133	0.128
Compactness (Bagi and Orosz, 2020)	0.739	0.729	0.715	0.715	0.713	0.720
	Particle dimensions from minimal oriented bounding box					
Intercept sphericity (Krumbein, 1941)	0.901	0.892	0.907	0.904	0.902	0.905
Flatness (Kong and Fonseca, 2018)	0.036	0.059	0.046	0.036	0.026	0.026
Elongation (Kong and Fonseca, 2018)	0.129	0.131	0.116	0.125	0.132	0.127
Flatness (Potticary et al., 2015)	0.023	0.038	0.030	0.023	0.017	0.017
Elongation (Potticary et al., 2015)	0.047	0.049	0.043	0.046	0.049	0.047
Compactness (Potticary et al., 2015)	0.929	0.913	0.928	0.931	0.934	0.937
Flatness (Angelidakis et al., 2022)	0.018	0.030	0.023	0.018	0.013	0.013
Elongation (Angelidakis et al., 2022)	0.069	0.070	0.062	0.067	0.071	0.068
Compactness (Angelidakis et al., 2022)	0.913	0.900	0.915	0.915	0.916	0.919
	Particle dimensions from fitted ellipsoid					
Intercept sphericity (Krumbein, 1941)	0.900	0.902	0.889	0.887	0.895	0.900
Flatness (Kong and Fonseca, 2018)	0.091	0.192	0.114	0.082	0.082	0.085
Elongation (Kong and Fonseca, 2018)	0.105	0.047	0.110	0.127	0.116	0.108
Flatness (Potticary et al., 2015)	0.060	0.134	0.076	0.054	0.054	0.056
Elongation (Potticary et al., 2015)	0.039	0.017	0.041	0.048	0.043	0.040
Compactness (Potticary et al., 2015)	0.901	0.849	0.883	0.898	0.903	0.904
Flatness (Angelidakis et al., 2022)	0.047	0.106	0.060	0.043	0.043	0.044
Elongation (Angelidakis et al., 2022)	0.055	0.024	0.058	0.068	0.062	0.057
Compactness (Angelidakis et al., 2022)	0.898	0.870	0.882	0.889	0.896	0.899

Example 2: Shape characterisation results for a sample of 50 railway ballast particles

In this example, results are demonstrated from a particle shape characterisation analysis of 50 railway ballast grains. The particle geometries have been scanned by Xiao et al. (2017) using a hand-held laser scanner. Figure 3.5a shows the alteration of the degree of true sphericity for the 50 ballast grains, starting from the original particle, the convex hull and then for descending fidelity levels from 200 to 25 triangular faces on the surface of each particle. Figure 3.5b demonstrates a Zingg plot for the original particle shapes (Zingg, 1935), along with isolines of intercept sphericity values (Krumbein, 1941), allowing for a classification of the particles' morphology for the whole sample.

To further quantify the loss of fidelity introduced by the simplification of the ballast particle shapes, the values of more indices of particle form are plotted against

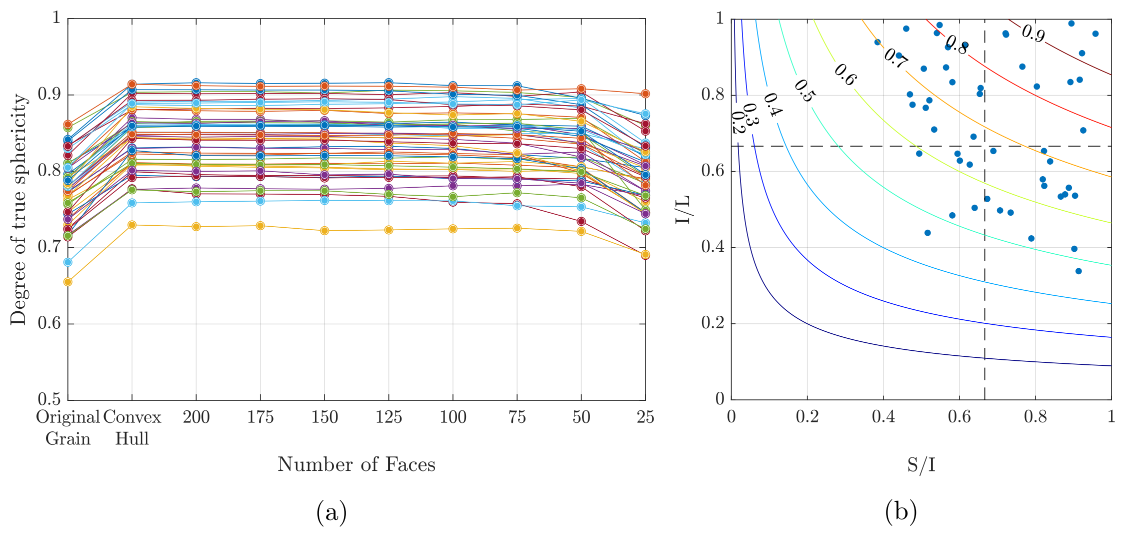


Figure 3.5: Shape characterisation of 50 railway ballast grains: (a) Degree of true sphericity (Wadell, 1932) for various fidelity levels; (b) Zingg plot for the original fidelity level (Zingg, 1935). The contour lines represent the values of Intercept Sphericity (Krumbein, 1941).

the number of faces, i.e. across fidelity levels. In this example, particle shape is simplified employing a *quadric edge-collapse decimation technique*, available within the `Computational Geometry Algorithms Library` (The CGAL Project, 2020), a dependency of `SHAPE`. These indices are calculated based on the main particle dimensions, which in this case are estimated using both an oriented bounding box (OBB) of minimal volume and a fitted ellipsoid (ELL), using non-linear least square fitting, aiming to quantify the sensitivity of each method for different fidelity levels. Figure 3.6 shows values of Intercept Sphericity (Krumbein, 1941) for both cases and for all fidelity levels. It becomes evident that for most particles, considering the oriented bounding box led to higher values of intercept sphericity, compared to the values corresponding to the fitted ellipsoid, for the analysed particles, by about 10%. The oriented bounding box results show smaller differences comparing the original fidelity level to the convex hull and the 200 faces, while for the fitted ellipsoid results, an increase of intercept sphericity is observed comparing the convex hull to the simplified fidelity level with 200 faces on the surface of the particle.

Although the values of sphericity can indicate if a particle is compact or not, as discussed in Chapter 2, it does not provide information on whether non-compact particles are flat or elongated. To this end, two measures of flatness and elongation, proposed by Potticary et al. (2015) and Angelidakis et al. (2022) are calculated across fidelity levels, as shown in Figure 3.7 and Figure 3.8, respectively.

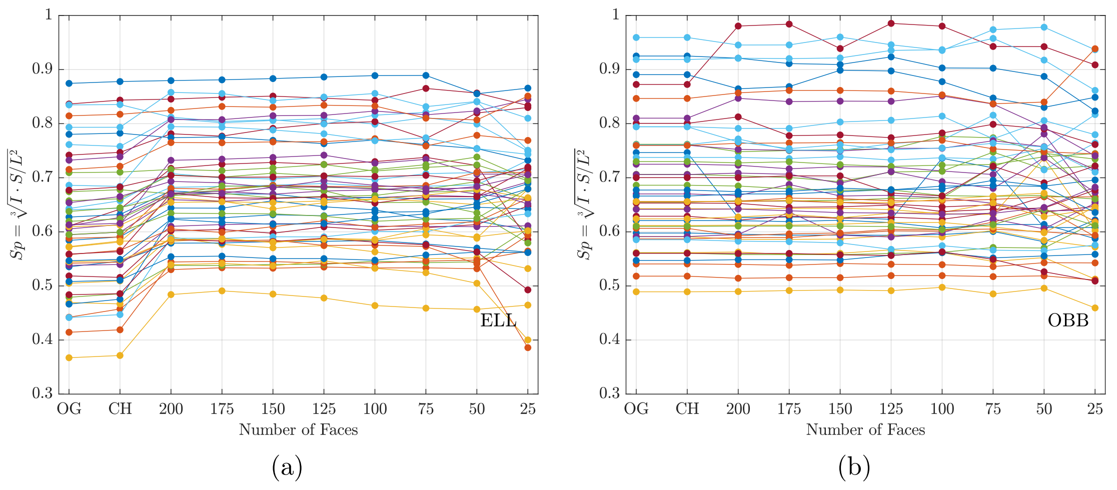


Figure 3.6: Intercept Sphericity (Krumbein, 1941) of 50 railway ballast grains. Main dimensions calculated using a: (a) fitted ellipsoid (ELL); (b) oriented bounding box (OBB). The first two fidelity levels correspond to: *OG: Original Grain, CH: Convex Hull.*

For both sets of indices, it can be observed that using an oriented bounding box to estimate the main particle dimensions leads to better preservation of particle form (both flatness and elongation) in most cases, comparing the original fidelity level to the 200 faces and down to 25 faces on the surface of each particle. The OBB-derived results show high error levels for a small subset of particles (less than 5% of the full sample), indicating either a limitation of the oriented bounding box to capture the main particle dimensions consistently or an insufficiency of the employed simplification technique to preserve the main morphological characteristics pertaining to the form of the analysed particles. As a result, if a modeller was to use these particles in a numerical simulation, they could either attempt a supervised pre-processing before particle generation in the DEM, to eliminate the simplified particles that did not preserve their form or else to set a threshold for an acceptable level of error, effectively automating the elimination process for oversimplified particles. The indices of Potticary et al. (2015) and Angelidakis et al. (2022) show similar levels of sensitivity across fidelity levels, with same trends regarding the consideration of a fitted ellipsoid or an oriented bounding box, to measure the main particle dimensions.

Example 3: Characterisation of images with noise

The products of digital imaging are always influenced to some degree by the existence of numerical noise. Regardless of the technique used to digitise the geometry

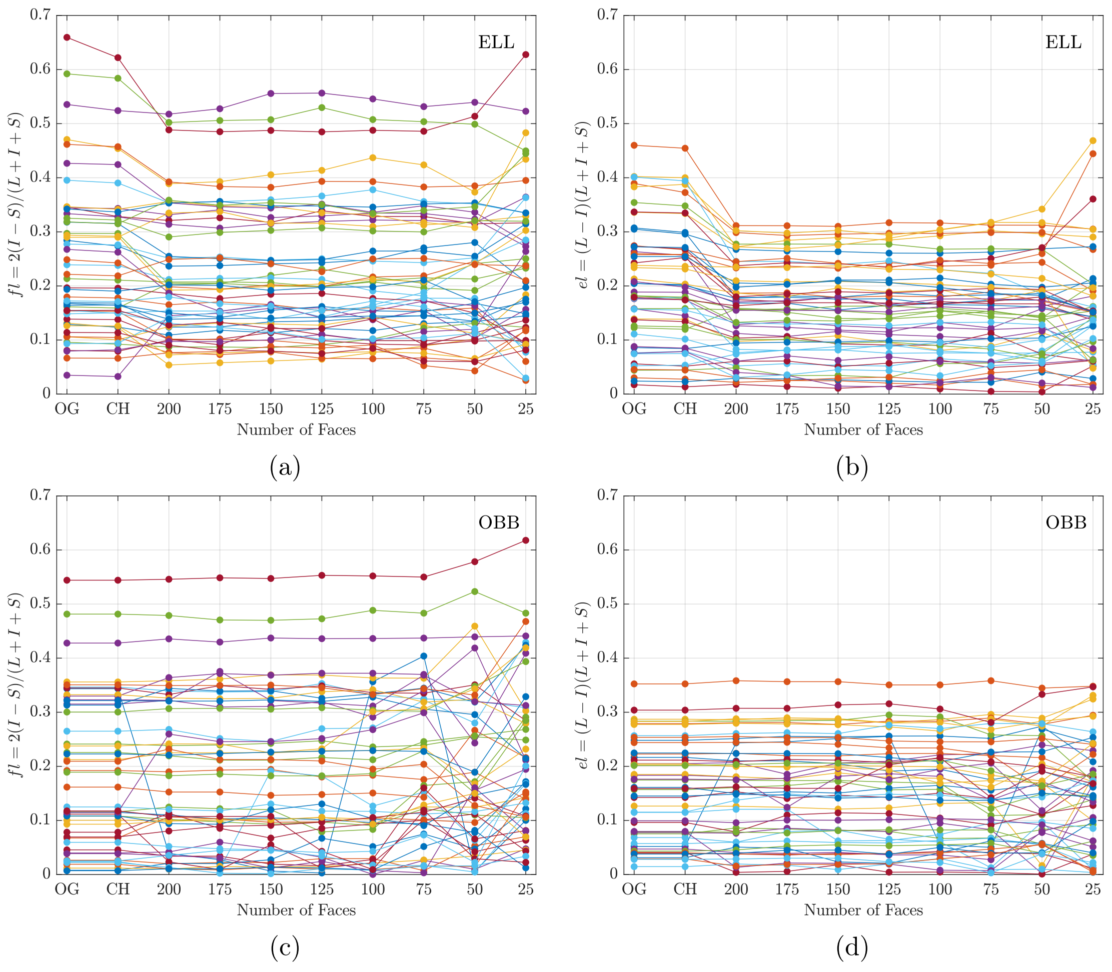
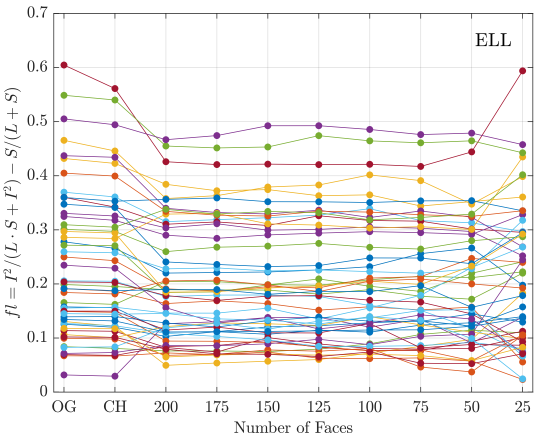


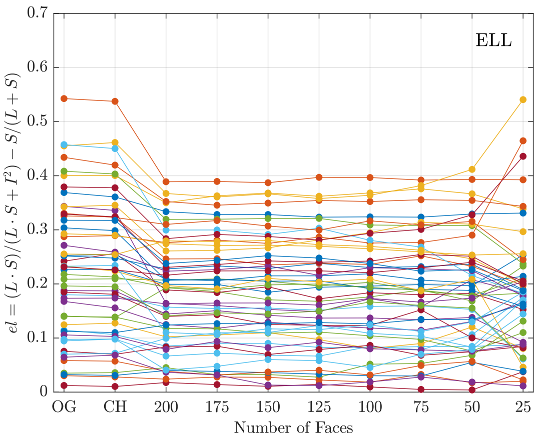
Figure 3.7: Flatness and elongation (Potticary et al., 2015) of 50 railway ballast grains. Main dimensions calculated using a: (a,b) fitted ellipsoid (ELL); (c,d) oriented bounding box (OBB). The first two fidelity levels correspond to: *OG: Original Grain, CH: Convex Hull.*

of a particle, (e.g. computerised tomography or laser scanning) or the format of the output (3D images and point clouds, respectively), defects are always present. Preparing imaging data at a pre-processing stage is typical before using them for shape characterisation or numerical simulations of any type, including tasks such as: filtering, removing of isolated elements (e.g. pixels or points) or applying smoothing and refining techniques.

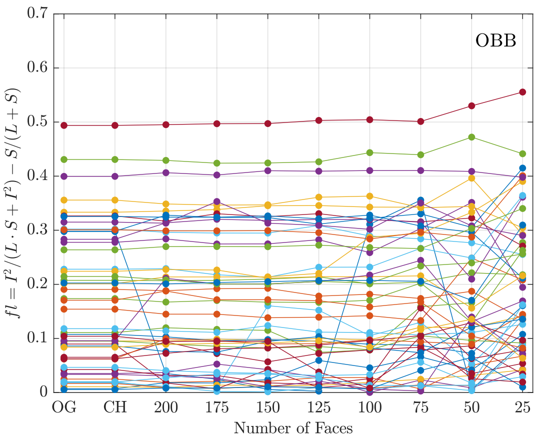
Wiebicke et al. (2017) studied how defects of realistic images derived using computerised tomography can affect the segmentation of individual particles. Starting from computer-generated synthetic-images of spherical assemblies, they generated realistic-like images, with the addition of Gaussian noise and blur. Gaussian noise



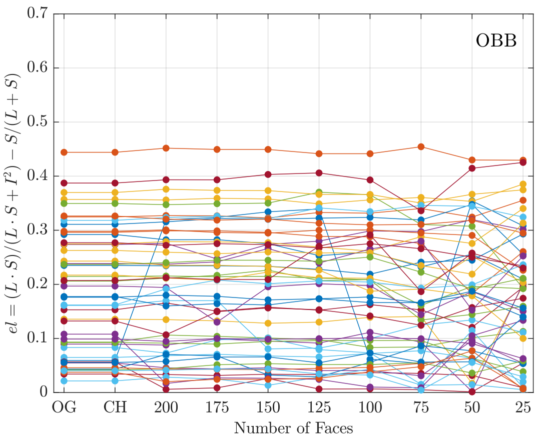
(a)



(b)



(c)



(d)

Figure 3.8: Flatness and elongation (Angelidakis et al., 2022) of 50 railway ballast grains. Main dimensions calculated using a: (a,b) fitted ellipsoid (ELL); (c,d) oriented bounding box (OBB). The first two fidelity levels correspond to: *OG: Original Grain, CH: Convex Hull.*

and blur are the two elements present in every real image captured using computerised tomography.

This example demonstrates a sensitivity analysis, where a 3D image of an ellipsoidal particle is analysed using `SHAPE`. Then, blur and noise of different levels are added, to identify the effect of the induced noise on the form characterisation of the overall image. Blur is created by applying Gaussian filtering of varying standard deviations. Figure 3.9 demonstrates an ellipsoid with radii 150$\times$ 75 $\times$ 105 mm, where increasing levels of blur and Gaussian noise are added.

Figure 3.10 shows the absolute error values of several shape parameters, for all levels of blur and noise. All examined parameters correspond to the characterisation of particle form, including Convexity, Intercept sphericity (Krumbein, 1941) and Flatness and Elongation (defined as in Kong and Fonseca, 2018). Two ways of calculating the main particle dimensions are employed, corresponding to the axes of an Oriented Bounding Box (noted with "OBB" in the figure), and the axes of a fitted ellipsoid (noted with "ELL" in the figure), aiming to quantify the sensitivity of each method to noise and blur. For all shape parameters, the maximum value of error is limited below 7.5%, 15% and 20%, for blur levels with standard deviations σ_{noise} equal to 1, 3 and 5, respectively.

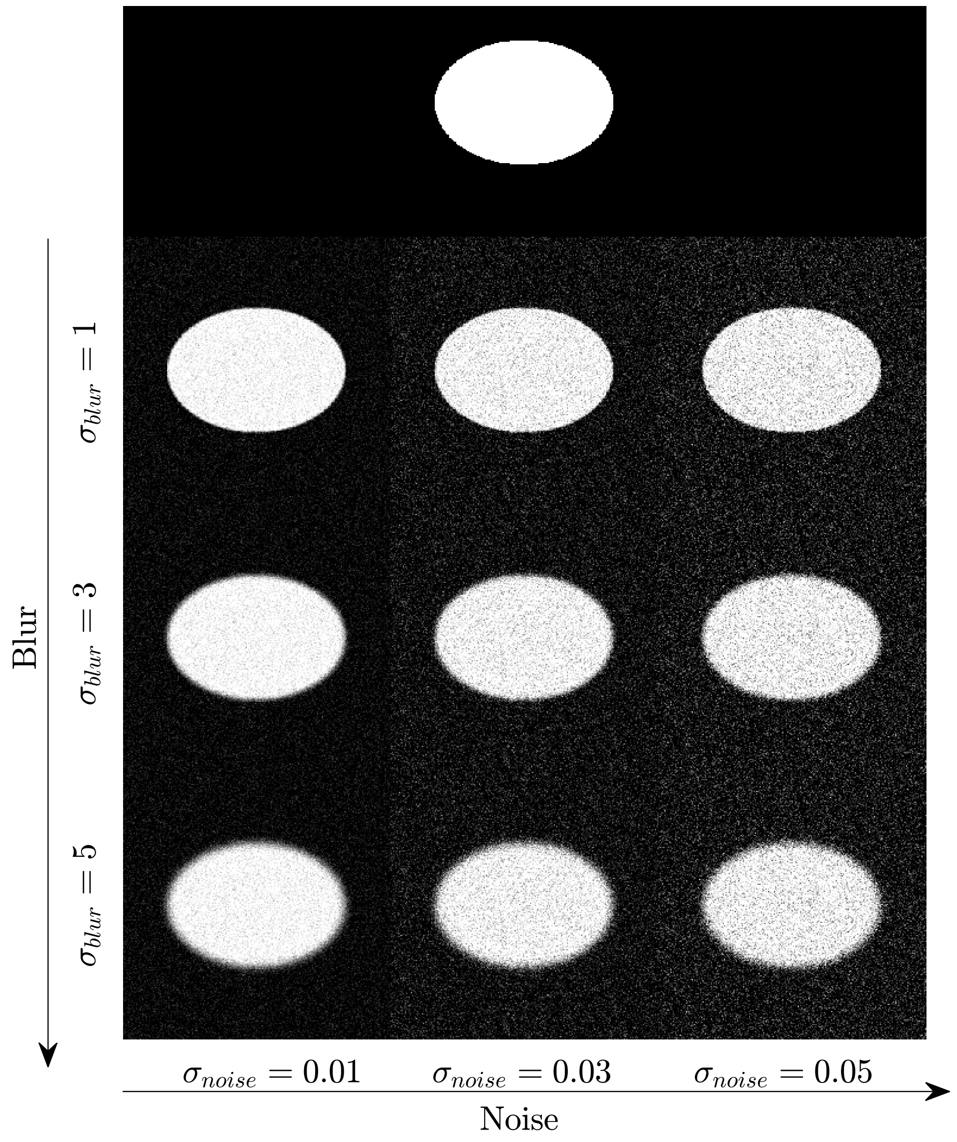
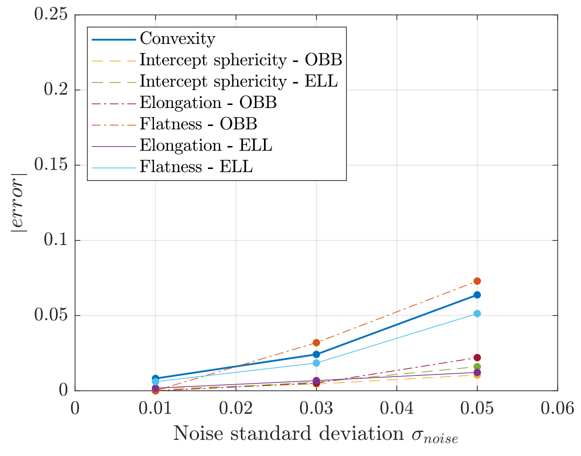
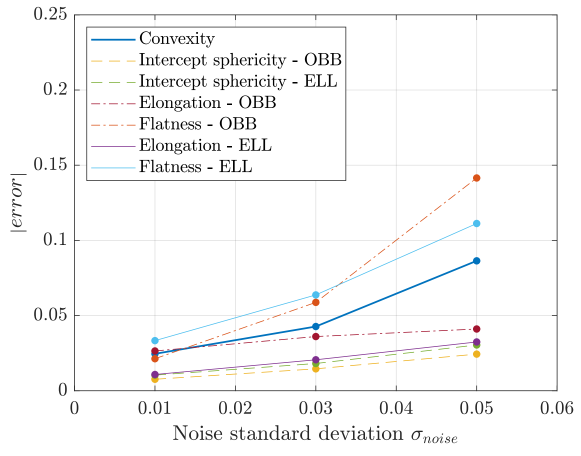


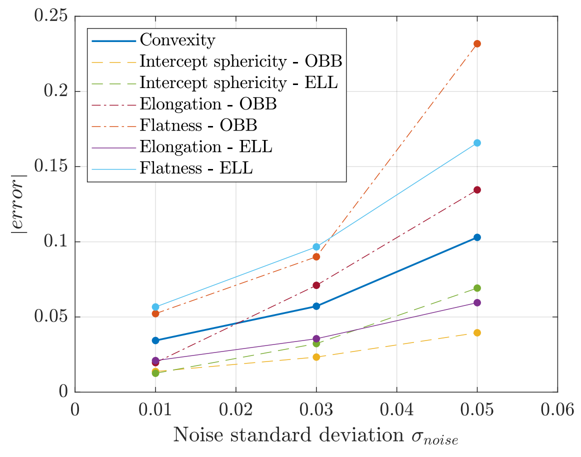
Figure 3.9: 2-D slices of ellipsoids for various levels of blur and noise.



(a)



(b)



(c)

Figure 3.10: Absolute error of shape parameters for images with increasing levels of blur and noise: (a) $\sigma_{blur} = 1$; (b) $\sigma_{blur} = 3$; (c) $\sigma_{blur} = 5$. Flatness and elongation are calculated as in Kong and Fonseca (2018). The particle dimensions are calculated using an oriented bounding box (OBB) and a fitted ellipsoid (ELL).

Example 4: Characterisation of concave particles

This example means to demonstrate the capacity of `SHAPE` to characterise the morphology of particles with concavities. To establish a meaningful comparison, the morphology of a ring torus was characterised, where analytical expressions exist for several of its geometrical characteristics.

The analysed torus shown in Figure 3.11 has a horizontal ring of radius 3.0 cm and a vertical section of radius 1.0 cm. To tessellate the surface of this particle, 300 subdivisions are considered along its horizontal ring and 100 subdivisions along

Table 3.4: Morphological characteristics of a torus. Validation of `SHAPE` against theoretical values and `Meshlab`.

Shape parameter	Theory	SHAPE	**error**$=1-\frac{SHAPE}{Theory}$	Meshlab	**error**$=1-\frac{SHAPE}{Meshlab}$
Volume (cm^3)	59.21760	59.17430	0.0731%	59.17434	0.0001%
Surface area (cm^2)	118.43530	118.41040	0.0210%	118.40588	-0.0038%
Inradius (cm)	1.0000	1.0002	-0.0200%	-	-
Circumradius (cm)	4.0000	4.0011	-0.0275%	-	-
	303.49030	303.22310	0.0880%	303.22311	0.000005%
Inertia values (cm^5)	303.49030	303.22310	0.0880%	303.22311	0.000005%
	577.37190	576.87850	0.0855%	576.87848	-0.000004%
Convexity	0.655453	0.655200	0.0387%	0.655200	0.0000%
Sphericity	0.620350	0.620200	0.0242%	0.620202	0.0003%
Outer ring					
Gaussian curvature (cm^{-2})	0.250000	0.251100	-0.4400%	0.250491	-0.2431%
Mean curvature (cm^{-1})	0.625000	0.625600	-0.0960%	0.625345	-0.0408%
Top ring					
Gaussian curvature (cm^{-2})	0.000000	0.000227	-	0.000228	0.3947%
Mean curvature (cm^{-1})	0.500000	0.501000	-0.2000%	0.501560	0.1117%
Inner ring					
Gaussian curvature (cm^{-2})	-0.500000	-0.497900	0.4200%	-0.500564	0.5322%
Mean curvature (cm^{-1})	0.250000	0.251300	-0.5200%	0.249488	-0.7263%

each of its sections, resulting in a particle with 30,000 vertices and 60,000 faces.

The geometrical parameters of the torus calculated analytically and by `SHAPE`, along with the corresponding errors are reported in Table 3.4. The errors are consistently below 0.5%. To further investigate the source of errors, the generated mesh of the torus was reanalysed using `Meshlab` (Cignoni et al., 2008). It emerges that `Meshlab` and `SHAPE` provide very similar results and therefore, the errors can be attributed to the tessellation of the torus into a discrete mesh.

3.5 Software impact

In practice, particle shape characterisation is typically carried out based on a handful of selected shape indices, while different interpretations of these indices exist in literature, aiming to describe the same shape aspects. The effectiveness of these indices to characterise the particle shape is still an open discussion and researchers and practitioners use the indices they find more representative subjectively, since to date, a consensus does not exist.

This piece of software is meant to facilitate a framework where the user can have easy access to a -growing- variety of different shape indices, while they can monitor

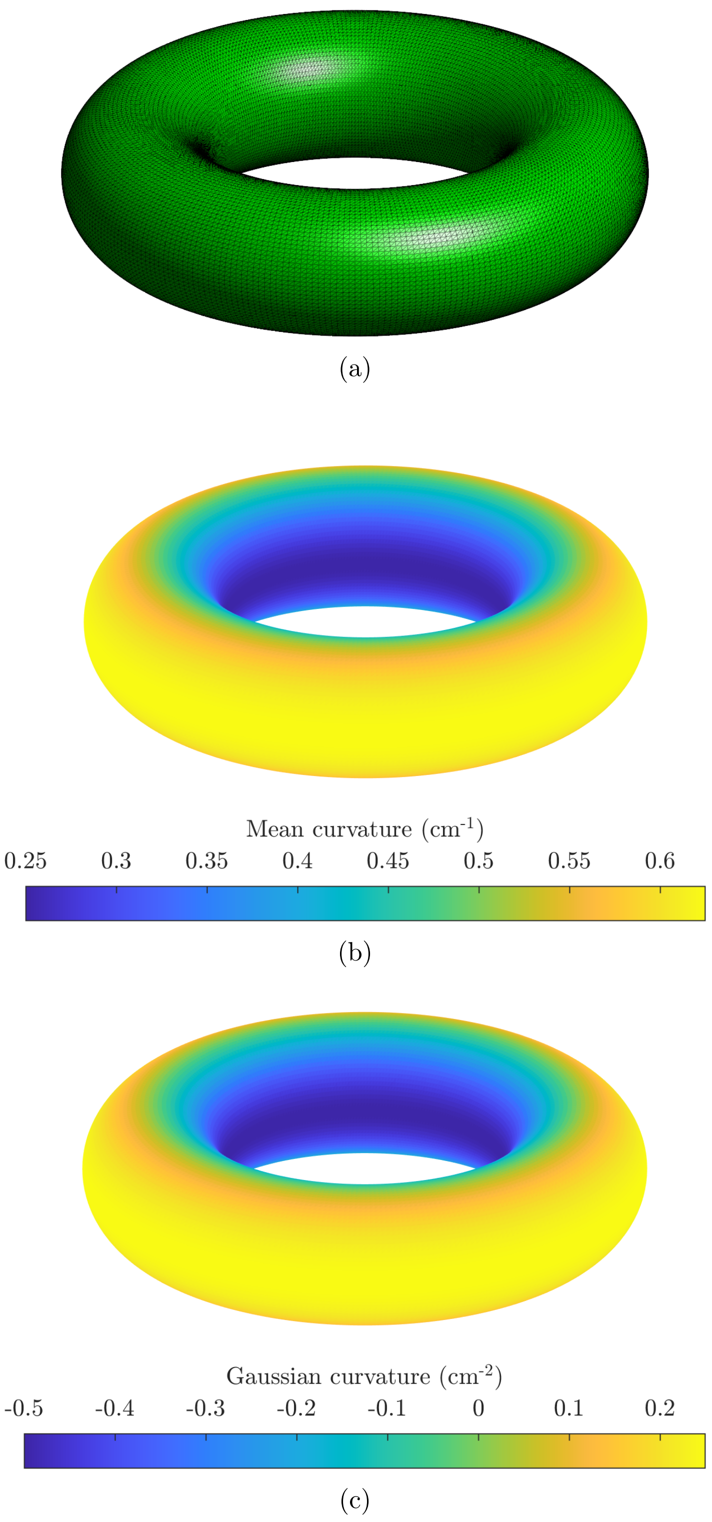


Figure 3.11: Analysed torus:(a) surface mesh; (b) mean curvatures; (c) Gaussian curvatures.

possible differences between indices which are meant to characterise the same shape aspects. For instance, different interpretations of sphericity can lead to different results, while all of them are meant to represent how closely a particle resembles a sphere. That is, this software is expected to provide a better comprehension in the study of particle shape characterisation, providing a pool of available shape indices. Users are encouraged to request or further develop the implementation of more shape indices they might be using in their work.

Utilising the ability of this software to process not just single particles, but whole particulate assemblies of three-dimensional particles, the authors intend to provide the community with a comprehensive tool to study the effect of particle shape in a quantitative and effective manner. This will facilitate the incorporation of 3D shape characterisation in industrial standards and guidelines.

3.6 Conclusions

A new open-source software is presented to perform shape characterisation of three-dimensional non-spherical particles. The following conclusions about the significance and reach of the work can be drawn:

1. `SHAPE` provides seamless characterisation and simplification of particle morphology, considering three key aspects of particle shape, namely form, roundness and surface roughness. This allows the characterisation of large particulate assemblies composed of thousands of particles, without human intervention.

2. For each aspect of shape, the code can calculate the most relevant shape descriptors, along with multiple approaches to calculate the main particle dimensions. This means the code can be employed for the many practical applications involving granular materials across the spectrum of the engineering and physical sciences.

3. The user is in control of the morphological simplification of each particle processed by setting acceptability thresholds for the error measured by the code for each shape index.

Having developed a shape analyser, which can also generate simplified shapes to be used in numerical simulations, this tool has the potential to be of use to researchers and practitioners alike investigating the influence of particle shape on the mechanical behaviour of granular assemblies at any scale of interest, including but not limited to powders, sands, silts, ballast, rockfills, and the many byproducts of process engineering.

`SHAPE` provides a user-friendly platform where future shape indices developed by the research community can be tested and pave the way for the development of new industrial standards based on 3-D particle characterisation.

Chapter 4

CLUMP: a Code Library for Universal Multi-sphere Particles

4.1 Introduction

The numerical simulation of particulate assemblies typically involves the use of Molecular Dynamics, where spheres are the predominant particle shape, and the Discrete Element Method (DEM), where more choices exist to model non-spherical particles. Clumps and clusters of spheres have been used to simulate non-spherical particles, primarily due to the simplicity of contact detection among spheres and their ability to approximate practically any irregular geometry. Various approaches have been proposed in the literature to generate such clumps or clusters, while open-source numerical codes applying these are scanty. The `CLUMP` code, proposed in Angelidakis et al. (2021a) , provides a unified framework, where a particle morphology can be approximated using different clump-generation approaches from the literature. This framework allows comparing the representations of the particle generated by the different approaches both quantitatively and qualitatively, providing the user with the tools to decide which approach is more appropriate for their application. Also, one novel generation technique is proposed. Outputs are provided in formats used by some of the most popular DEM codes. Moreover, the resulting clumps can be transformed into surface meshes, allowing for easy characterisation of their morphology. Finally, the effect of clump-generation techniques on the mechanical behaviour of granular assemblies is investigated via triaxial compression tests.

4.2 Previous work

Modelling real particles as clumps of overlapping spheres or clusters of non-overlapping spheres (Orefice and Khinast, 2020) is common practice in both academic and industrial studies, when particle morphology is of interest. Clustering spheres to represent a single particle offers versatility within a simulation, as the fine details of particle morphology such as local roundness, roughness or concavities can be introduced. Moreover, the computational cost is affordable due to the simplicity of performing contact detection among spheres, given the total number of spheres considered. An abundance of methods has been proposed in the literature to perform such a feat, including Favier et al. (1999), Matsushima and Saomoto (2002), Bradshaw and O'Sullivan (2004), Price et al. (2007), Wang et al. (2007), Garcia et al. (2009), Ferellec and McDowell (2010), Taghavi (2011), Gao et al. (2012), Li et al. (2015), Zheng and Hryciw (2016), Haeri (2017), and Katagiri (2019). These methods have been employed in multiple research fields, namely physics (Markauskas et al., 2010), pharmaceutical engineering (Tamadondar and Rasmuson, 2020), chemical engineering (Nan et al., 2017), agriculture (Pasha et al., 2016), civil engineering (Wang et al., 2007; Suhr and Six, 2020), mining engineering (Cho et al., 2007) and geosciences (Kodicherla et al., 2020). However, only a limited number of implementations are available open access, such as Haeri (2017): `https://github.com/sihaeri/DEM-ClumpedSphere` and Bradshaw and O'Sullivan (2004): `https://github.com/mlund/spheretree`.

In a typical numerical study employing clumps, the researcher employs an available method to approximate the real particle morphology. Most commercial DEM software offer one specific approach, developed in-house, to assist users in defining such clumps. For example, PFC3D (Itasca) provides the Taghavi (2011) method and EDEM (Altair Engineering) offers an in-house method in their most recent version. Open-source DEM codes (such as YADE or LAMMPS) often do not provide any methods to generate clumps, although YADE offers functions to fill a target geometry with non-overlapping spheres. Note that the quality of the particles generated by the available approaches varies with the type of application / problem investigated and with particle morphologies. At present, a quantitative analysis of the effect of clump generation methods is missing from the literature. Also, no or very little information is provided in the publications that introduce a new method for generating a clump about the performance of the method in preserving morphological characteristics. To this end, `CLUMP` provides a unified framework for the comparison of different multi-sphere particle generation methods and it facilitates morphology characterisation of these particles. The code is easy to use and it can provide an automated workflow to generate thousands of particles within reasonable time runs.

4.3 Software description

The code comprises three main modules: (1) Functions to generate clumps, (2) Functions to export clumps and (3) Transformation of clumps to surface meshes, allowing for their morphological characterisation and comparison with the original particle. The first module is the main core of the software, while the second and third ones have an auxiliary role, functioning as utility scripts.

4.3.1 Software architecture

This section details the main structure of the code, which is also visualised in Figure 4.1.

Module I: Generate Clump `CLUMP` supports three approaches to generate clumps, namely Favier et al. (1999), the first multi-sphere approach proposed in the literature; Ferellec and McDowell (2010), a widely used approach; and a newly proposed approach based on the Euclidean distance transform of 3D images. Figure 4.2 shows the main objects of the clump generation module.

Module II: Export Clump The generated clumps represented by the centroids and radii of multi-spheres can be exported in various formats (*.py, *.csv), compatible with some of the most mainstream commercial and open-source DEM codes, including EDEM, PFC3D, YADE, and LAMMPS.

Module III: Characterise Clump A surface extraction module is developed, transforming each clump into a surface mesh, which in turn can be imported in SHAPE (Angelidakis et al., 2021b), to perform shape characterisation.

`CLUMP` relies on several external functions available within the Matlab FEX community. In particular, the outsourced tasks include the handling of stereolithography

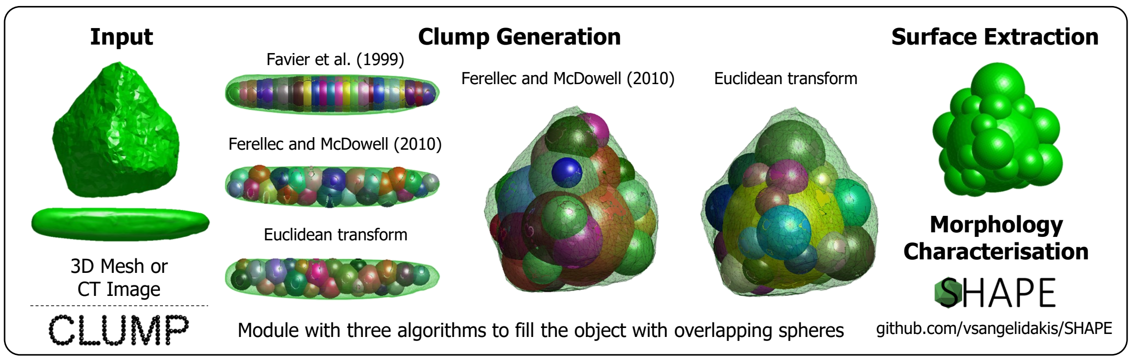

Input
Clump Generation
Surface Extraction
Favier et al. (1999)
Ferellec and McDowell (2010)
Euclidean transform
Ferellec and McDowell (2010)
Euclidean transform
3D Mesh or CT Image
CLUMP
Module with three algorithms to fill the object with overlapping spheres
Morphology Characterisation
SHAPE
github.com/vsangelidakis/SHAPE

Figure 4.1: The main modules of `CLUMP`. The surface of a rice grain and a sand grain are used to generate clumps of overlapping spheres. The surface of each clump can be extracted, allowing for a quantitative characterisation of their morphology.

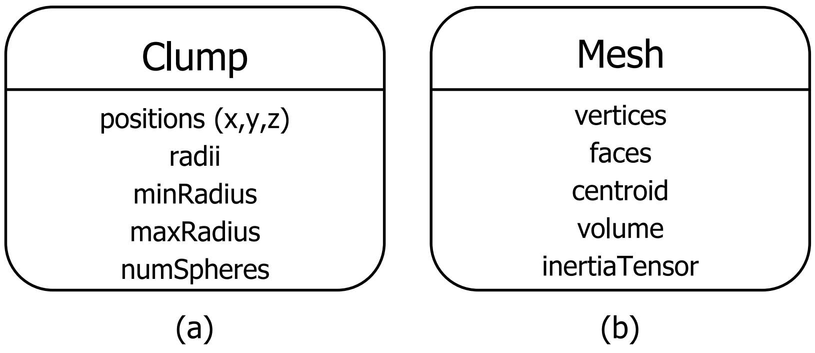


Figure 4.2: Main objects of the code structure (a) clump (b) mesh.

files (Micó, 2020), mesh manipulations (Fang and Boas, 2009), and calculation of rigid body parameters (Semechko, 2021).

4.3.2 Software functionalities

`CLUMP` supports the following clump-generation approaches:

4.3.2.1 Favier et al. (1999)

This method only applies to symmetrical particles which are constructed of spheres whose centres are located on the particle axis of symmetry. Spheres may overlap and may vary in diameter along the length of the axis of symmetry. The surface of a particle is approximated by inscribing spheres such that the surface of each sphere is tangent to the surface of the particle at the point of contact. The position of each element sphere is fixed relative to the other elements within a particle.

4.3.2.2 Ferellec and McDowell (2010)

The clump generation methodology of Ferellec and McDowell (2010) selects random points of the mesh of the particle surface and generates tangent spheres of increasing radii, until they intersect a different vertex of the particle surface. This methodology was designed to eliminate artificial asperities of flat faces, through the consideration of large numbers of spheres per particle. Due to this, the approach of Ferellec and McDowell (2010) will not work as well if a small number of spheres is sought, e.g. 2 or 3 spheres per clump, as the selection of the initial points is random. This happens because the methodology lacks a deterministic criterion of which vertex to choose every time to generate spheres. In the implementation of the Ferellec and McDowell (2010) method within `CLUMP`, a `seed` parameter is introduced in the selection of vertices to achieve reproducible clumps. Though, this `seed` parameter is not guaranteed to lead in any way to the best-possible clump, for the given number of spheres, and is only introduced to initialise the random number generation algorithm used to select the vertices where sphere generation takes place in each iteration.

In more detail, the volume of the real particle is filled optimally with spheres of different sizes: from a point chosen randomly on the surface, a sphere is grown internally along the normal vector at that point to the maximum extent possible inside the boundary of the particle. In other words, the expansion of the sphere continues until it reaches another point on the surface of the particle. Then, the process is repeated for other points on the surface of the particle, which must be farther by a distance $\mathtt{d_{min}}$ to any existing sphere. The number of spheres inside the clump is directly related to the number of points on the surface, by monitoring a percentage $\mathtt{p_{max}}$, which if met, terminates the generation process. If the density of points on the surface of the particle is high enough, then the process can theoretically lead to a clump composed of thousands of spheres, although Ferellec and McDowell (2010) set criteria that can reduce local crowding of spheres, using a minimal radius $\mathtt{r_{min}}$, or the aforementioned $\mathtt{d_{min}}$ and $\mathtt{p_{max}}$ variables. If a large number of spheres per clump is computationally acceptable, this approach can lead to smaller artificial asperities of flat faces, compared to other methods, as Ferellec and McDowell (2010)report. This approach was designed to be applied on surface meshes, but is also applicable directly on point clouds, since techniques do exist to estimate the normal vector of each member point of a cloud.

Figure 4.3 shows several multi-sphere particles of the sand particle geometry employed in this study, using the Ferellec and McDowell (2010) approach. These clumps were generated using the same parameters d_{min}, p_{max}, r_{min} and only the `seed` parameter was varied, demonstrating the randomness of the sphere-generation points. This issue is minimised for larger numbers of spheres, as in the studies of Ferellec and McDowell (2010).

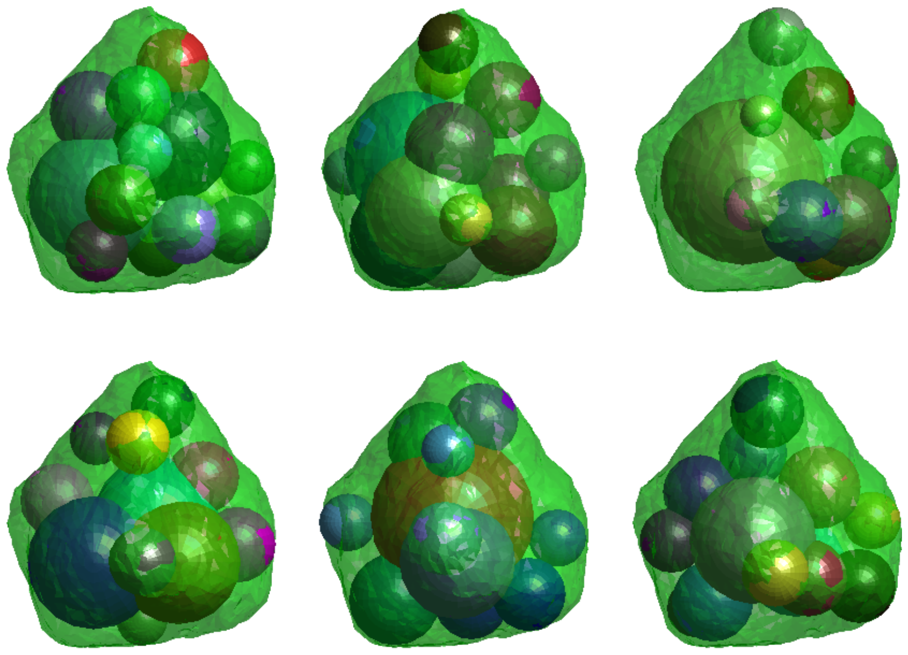


Figure 4.3: Sand particle: Clumps generated using the approach of Ferellec and McDowell (2010) with the same parameters, changing only the `seed` value.

4.3.2.3 Euclidean distance transform

A new approach is presented in this chapter to generate multi-sphere particles, based on the concept of the Euclidean distance transform of 3D images. Initially, the particle morphology is transformed from a surface mesh to a voxelated representation of a minimum dimension `div`, where the value of all voxels belonging to the particle is set equal to zero. Then, the inscribed sphere of the particle is found as the maximum value of the Euclidean distance transform of the voxelated image. The voxels corresponding to the inscribed sphere are then set equal to one and the Euclidean distance map is calculated for the new voxelated image. This process is repeated until a user-defined number of spheres is found (`N`) or until the user-defined minimum radius ($\mathtt{r_{min}}$) has been reached, as the spheres are generated in decreasing sizes. This methodology can also generate overlapping spheres, if only a percentage of the voxels making each new sphere is set equal to one, instead of all of them, expressed by a variable `overlap`, which takes values within the range $[0, 1)$.

A simple example is shown in Figure 4.4 to demonstrate the logic of generating clumps utilising the Euclidean transform of images. Assuming the triangular 2D particle of Figure 4.4, the Euclidean transform is applied, and its maximum value is equal to the radius r of the largest inscribed circle of the particle. A circle with radius r is generated and the pixels within a circle with radius $r \cdot (1 - overlap)$ are turned to zero in the next iteration. The procedure continues, applying the Euclidean transform for each new image of the particle and turning the respective pixels to zero for each new circle, until the required number of circles is generated, or the optional, user-defined value for a minimum radius r_{min} has been reached. Figure 4.4a demonstrates a step-by-step generation of a clump with three members with $overlap = 0.0$, while Figure 4.4b shows the same procedure for $overlap = 0.2$.

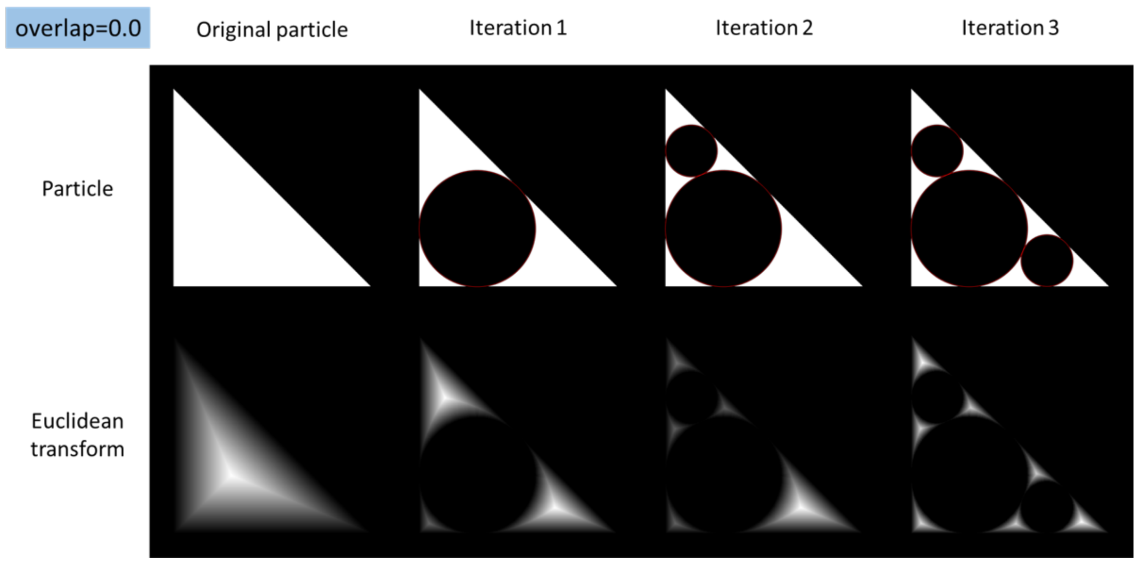
Figure 4.5 shows the algorithms behind the Ferellec and McDowell (2010) and Euclidean distance transform implementations. The output of the above approaches is a set of spheres, represented by their centroids and radii, as needed in commercial and open-source codes.

4.3.2.4 Comparison of clump generation methods for idealised particles

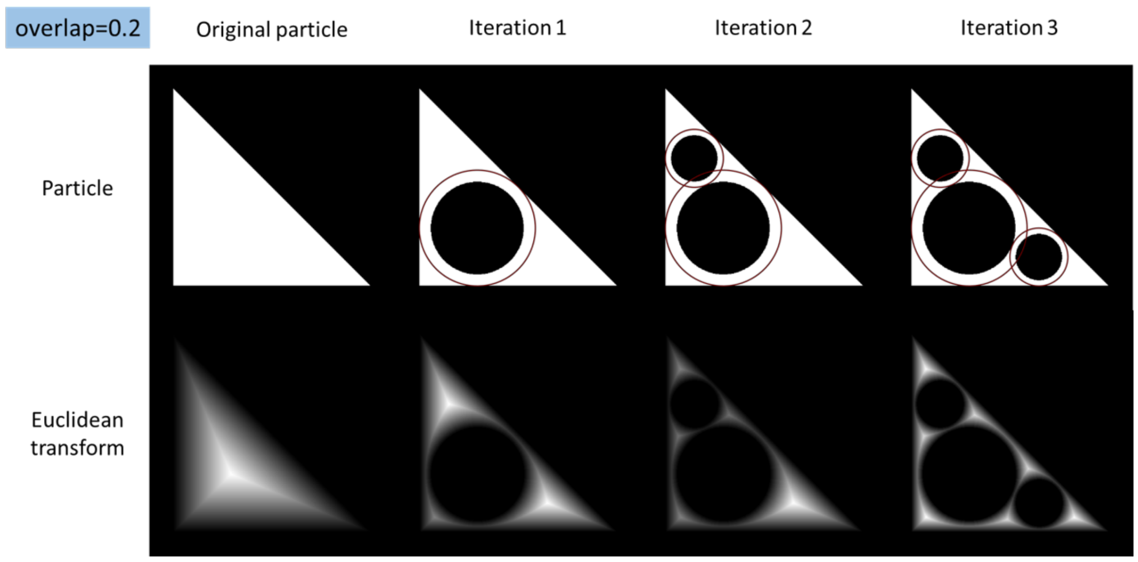
The clump generation method of Ferellec and McDowell (2010) and the proposed method based on the Euclidean transform are used to generate clumps for the five platonic solids, as shown in Figure 4.6. It becomes evident that the Euclidean transform method can lead to clumps that represent the volume and inertia tensor of the particles with relatively small numbers of spheres (Figure 4.6b). On the other hand, the method proposed by Ferellec and McDowell works well for larger numbers of spheres, where it can minimise artificial asperities and lead to almost flat faces (Figure 4.6a). For small numbers of spheres, the method of Ferellec and McDowell cannot guarantee a valid representation of the volume or inertia of the particle, due to the randomness of choosing the generation points of the tangent spheres.

4.3.2.5 Particle surface extraction

One of the main objectives of `CLUMP` is to allow for a direct comparison among the morphologies of multi-sphere particles generated using different approaches. To achieve a quantitative comparison, particle shape-characterisation tools can be used,



(a)



(b)

Figure 4.4: Clump generation using the Euclidean transform for (a) $overlap = 0.0$ and (b) $overlap = 0.2$.

such as SHAPE (Angelidakis et al., 2021b). To this end, the surface of each clump has to be extracted in the form of a surface mesh, allowing for compatibility between the clump generation and the shape characterisation codes.

Extracting the surface of a multi-sphere particle is not a trivial task. The problem can be tackled using many different approaches. In this implementation, the surface of each clump is extracted following the steps below:

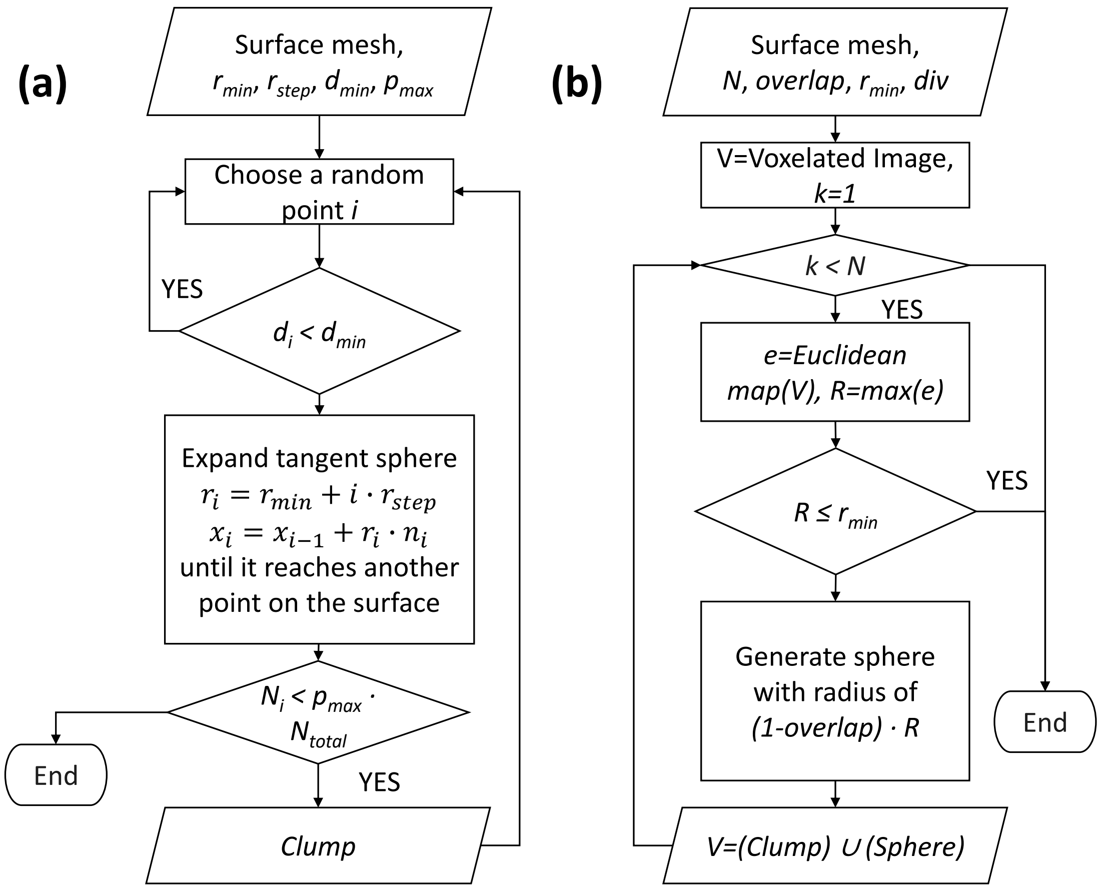

(a)
Surface mesh, r_{min}, r_{step}, d_{min}, p_{max}
Choose a random point i
YES
$d_i < d_{min}$
Expand tangent sphere $r_i = r_{min} + i \cdot r_{step}$ $x_i = x_{i-1} + r_i \cdot n_i$ until it reaches another point on the surface
$N_i < p_{max} \cdot N_{total}$
YES
End
Clump
(b)
Surface mesh, N, overlap, r_{min}, div
V=Voxelated Image, k=1
$k < N$
YES
e=Euclidean map(V), R=max(e)
$R \leq r_{min}$
YES
Generate sphere with radius of (1-overlap) · R
End
V=(Clump) ∪ (Sphere)

Figure 4.5: Flowchart of the main algorithmic steps behind each clump generating approach (a) Ferellec and McDowell (2010), (b) Euclidean distance transform.

1. Contact detection is performed between all possible pairs of the spheres making the clump, identifying the pairs with geometric overlap.

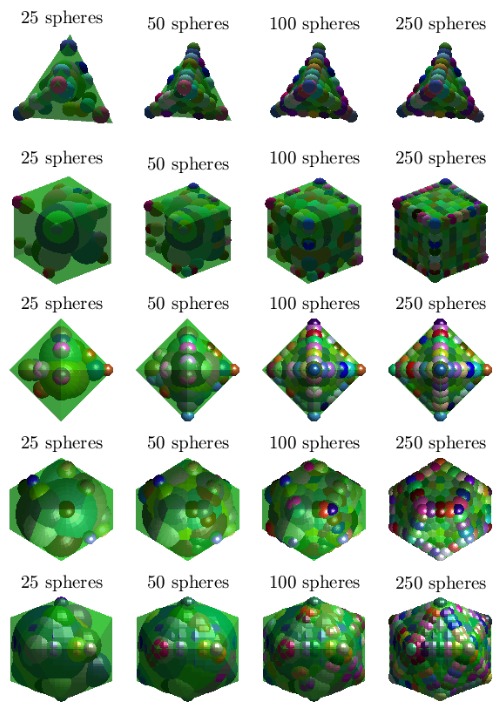
2. A point cloud is generated on the surface of each sphere.

3. For each pair of overlapping spheres, the points within the overlap region are deleted (Figure 4.7).

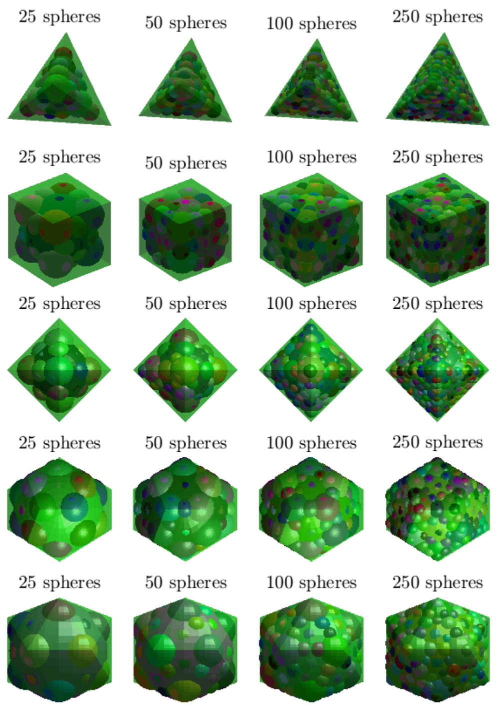
4. A circle is found as the intersection of each pair of overlapping spheres and points are generated along each circle, which are added to the point cloud.

5. The point cloud is then tessellated using a surface reconstruction technique, making the surface of the clump.

It should be noted that generating points on the surface of each sphere introduces mesh-dependency in the calculation of local shape parameters, like roundness and



(a)



(b)

Figure 4.6: Generation of clumps with increasing numbers of spheres (from left to right: 25, 50, 100, 250) for the five platonic solids (from top to bottom: tetrahedron, hexahedron, octahedron, dodecahedron, icosahedron), using the methods of (a) Ferellec and McDowell (2010) and (b) the Euclidean transform.

roughness. Therefore a dense mesh should be chosen when these shape aspects are of interest. The local morphology can also be affected by the technique chosen to perform surface reconstruction, which in this case is an implementation of the Crust method (Amenta et al., 1998) provided by Giaccari (2020). The points on the circles calculated as the overlap of each pair of spheres are added to the point cloud, aiming to minimise the numerical noise of the clump surface near the region of an overlap and provide a smooth transition from the surface of a sphere to its adjacent ones. Figure 4.7 shows the surface of a clump made of two spheres, where the points inside the overlap region (removed from the point cloud) and on the intersection circle (added to the point cloud) can be observed.

4.3.3 Sample code snippets

This section highlights two implementation details of `CLUMP`. First, Figure 4.8a demonstrates how the newly introduced approach based on the Euclidean distance

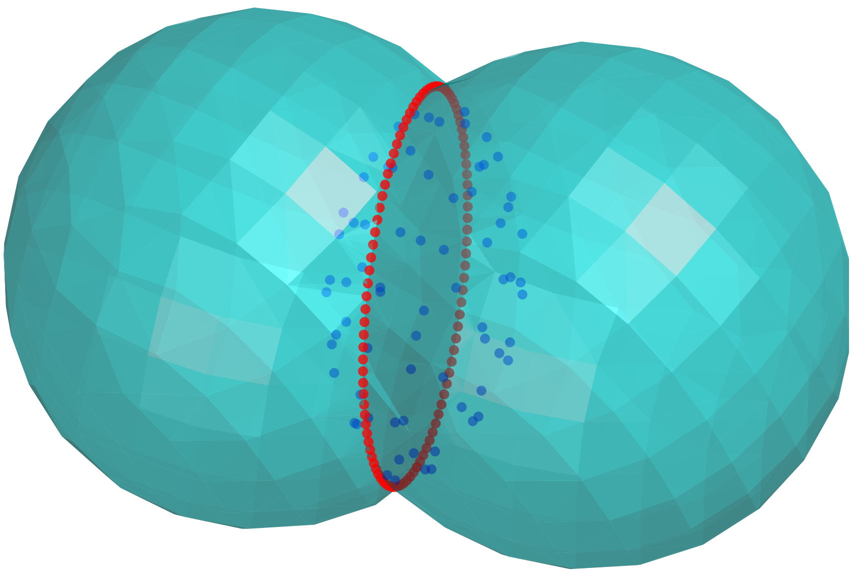


Figure 4.7: Tessellating the surface of a clump to transform it into a surface mesh. The red points are generated on the circle defined as the intersection of two overlapping spheres and added to the tessellation to aid surface reconstruction. The blue points correspond to vertices on the surface of each sphere which fall within the overlap region with a neighbouring sphere, i.e. they do not belong to the clump surface and are removed before tessellation.

transform allows the generation of overlapping spheres, while Figure 4.8b shows a typical loop of how a tangent sphere is grown in the methodology of Ferellec and McDowell (2010).

4.4 Illustrative examples

This example demonstrates the mechanical behaviour of assemblies of rice grains and soil grains under triaxial compression. The morphological characteristics of clumped spheres used in this illustrative example are presented in Table 4.1. Hereinafter, EU, FM and FA refer to grains represented by 25 spheres using the Euclidean transform, Ferellec and McDowell (2010) and Favier et al. (1999) methods, respectively. The samples consist of (1) rice grains using all three approaches, and (2) sand grains using the Euclidean transform and Ferellec and McDowell (2010) approaches. The surface meshes of the rice and sand particles, along with surface meshes of the produced clumps can be seen in Figure 4.9 and Figure 4.10. The major axis length of the rice grain and silica sand was approximately 18 mm and 6 mm, respectively. The particle shear modulus and particle Poisson's ratio were considered as 20 MPa

```
% vox: voxelated representation of particle
[dx,dy,dz] = meshgrid(1:size(vox,2), 1:size(vox,1), 1:size(vox,3));

for k=1:N % N: number of spheres
    edtImage = bwdist(~vox); % Euclidean map
    radius = max(edtImage(:)); % Inradius in voxel units
    [yCenter, xCenter, zCenter]= ind2sub(size(vox),find(edtImage == radius));
    dists = sqrt(sum(bsxfun(@minus,centroid,[xCenter,yCenter,zCenter]).^2,2));
    [~,i]=max(dists); % Index of the inscribed sphere

    sph=zeros(length(dy),length(dx),length(dz));
    sph=sqrt( (dx-xCenter(i)).^2 + (dy-yCenter(i)).^2 + (dz-zCenter(i)).^2 ) > (1-overlap)*radius; %
         Sphere
    vox=and(vox,sph); % Append the new sphere in the particle
end
```

(a)

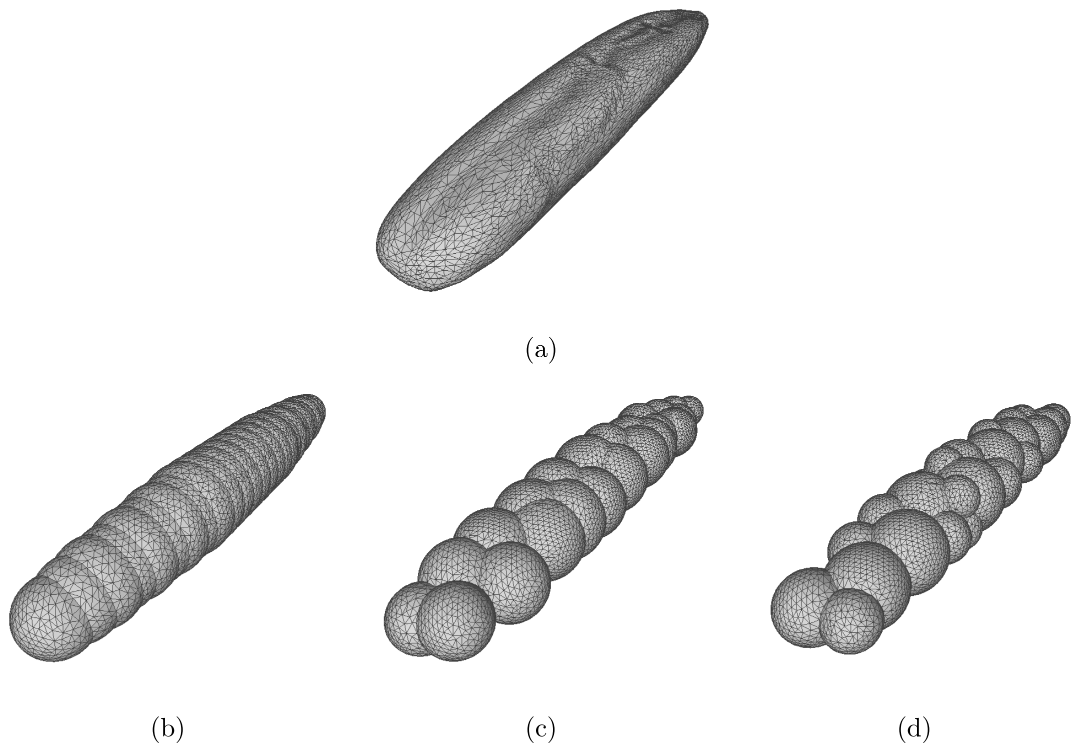
```
x=P(:,1); y=P(:,2); z=P(:,3);
for k=1:length(P) % P: vertices
    r=rmin;
    while reachedMaxRadius==false % while the sphere has not reached the surface
        while sphMin>-tol
            xC=x+r*n(1); yC=y+r*n(2); zC=z+r*n(3);
            % n: normal vector of vertex k
            distance=sqrt((x-xC).^2+(y-yC).^2+(z-zC).^2);
            % Distances of all points to the center of the sphere
            sph=(distance/r).^2-1;
            % Potential function (negative for points inside the sphere)
            sphMin=min(sph);
            r=r+rstep; % Grow radius for next step
        end
        reachedMaxRadius=true;
    end
end
```

(b)

Figure 4.8: Code snippets (a) How the overlap of spheres is imposed using a Euclidean distance map (b) Growing tangent spheres in Ferellec and McDowell (2010).

and 0.1 for rice grains and 29.1 GPa and 0.23 for sand grains, respectively. The particle density was 1470 kg/m^3 for the rice grains and 2650 kg/m^3 for the sand grains; these values were increased by 1000 times to save computational time.

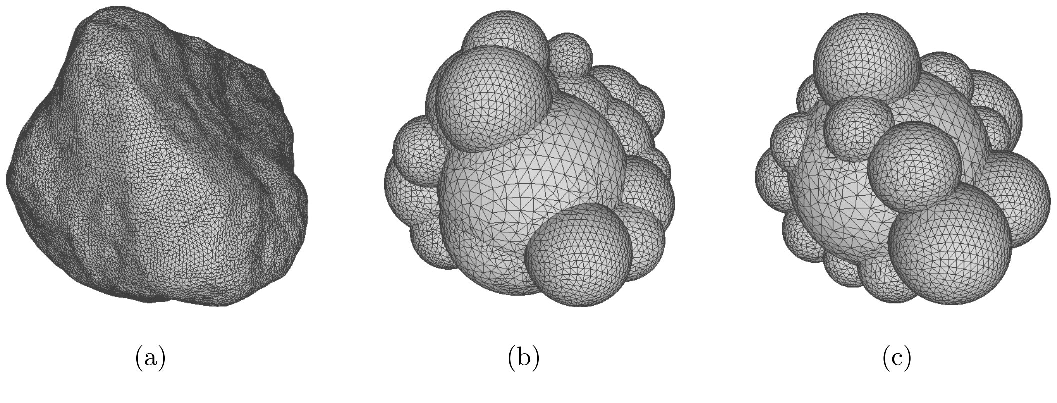
The LAMMPS molecular dynamics code (Plimpton, 1995; Nguyen and Plimpton, 2019) is employed with a modified servocontrol, as used by Hanley et al. (2015) and Morimoto et al. (2021). Each sample is composed of 5000 clumped grains, which are generated randomly without initial contacts with other particles, under no gravity conditions. The sample is subjected to an initial isotropic compression by moving the boundaries with a controlled velocity, to achieve an isotropic stress of 50 kPa. A simplified Hertz-Mindlin contact model with Coulomb friction is adopted. Dense, medium and loose samples are prepared using inter-particle friction coefficients (μ) of



(a)

(b) (c) (d)

Figure 4.9: (a) Reconstructed rice particle; (b) clump of the rice particle using 25 spheres and the method of Favier et al. (1999); (c) clump of the rice particle using 25 spheres and the method of Ferellec and McDowell (2010); (d) clump of the rice particle using 25 spheres and the Euclidean transform method, proposed in this chapter.



(a) (b) (c)

Figure 4.10: (a) Reconstructed sand particle; (b) clump of sand particle using 25 spheres and the method of Ferellec and McDowell (2010); (c) clump of sand particle using 25 spheres and the Euclidean transform method, proposed in this chapter.

0.0001, ≈0.1 and 0.35, respectively, during the isotropic compression, giving relative density (Dr) values of 100%, 50% and 0%, following the approach of Morimoto et al.

(2021). In the subsequent triaxial compression, the μ value was set to 0.35 for all the cases as a material constant. In the present simulations, no damping was used.

The cubical samples composed of rice grains in Figure 4.11a were subjected to quasi-static axial compression in the Z-axis while keeping the lateral stresses in the X- and Y-axes at 50 kPa. The overall trend of stress-strain relationships is similar for each density level (Figure 4.11b); however, the FA Dr=100% case leads to lower peak and residual stresses compared to the FM and EU cases, while the FA cases lead to higher initial stiffness compared to the other two cases, i.e. the initial increase in deviator stress (Figure 4.11c). A clear difference between FA and the other approaches is evident in the variation of mean coordination number (Figure 4.11d). A representative sample composed of sand grains is illustrated in Figure 4.12a. Similar variations of stress-strain responses are observed between the EU and FM approaches (Figure 4.12b), although measurable differences in the variation of void ratio are evident (Figure 4.12c).

For the illustrative examples presented here, the EU and FM approaches gave comparable results, while the results of the FA approach differed. Referring to Table 4.1, this discrepancy can be attributed to the difference in the shape parameters of the generated multi-sphere particles. EU and FM provide similar shape parameters, indicating that quantifying shape parameters of the generated clumps is recommended as an effective measure to ensure the quality of modelling non-spherical grains. On the other hand, FA cannot represent particles with flatness, as the axial symmetric clumps it generates can only result in compact or elongated particle shapes, a fact that shows a distinct effect on the macro-mechanical behaviour of the simulated triaxial tests on rice. Interesting to note that apart from flatness, the rest of the studied morphological parameters between FA, and EU and FM are similar.

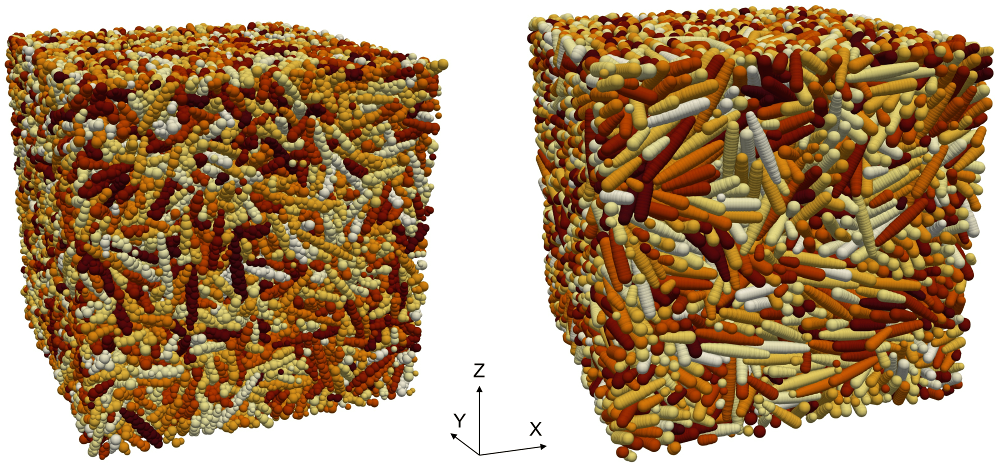
Overall, a single clump-generation method has not prevailed as objectively better than the rest for all particles and for all applications. Different methods will perform better for different particle shapes and depending on which aspect of shape is sought to be preserved during clump generation. For instance, the method of Favier et al. (1999) provides a simple and efficient generation method, but only applies to axisymmetric particles and cannot be used to approximate particles that demonstrate flat features or asymmetrical morphology. For irregular particles, the clump-generation method of Ferellec and McDowell (2010) can generate clumps with minimised artificial surface asperities, via the consideration of tangent spheres at the particle surface.

Table 4.1: Morphological and inertial characteristics of clumped spheres used in the illustrative examples.

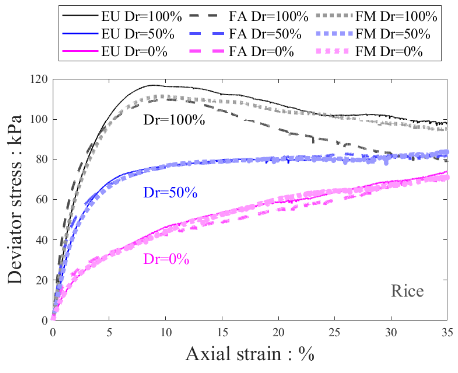
Shape Parameters	Original particle	Favier (1999) (25 spheres)	Ferellec and McDowell (2010) (25 spheres)	Euclidean transform (25 spheres)
Rice				
Sphericity*	0.6759	0.6831	0.591	0.5872
Convexity	0.9785	0.9645	0.8037	0.7804
Flatness†	0.2238	0.0002	0.2642	0.2195
Elongation†	0.812	0.826	0.7993	0.7964
Centroid (mm)	[0,0,0]	[0.110,0,0]	[-0.028,0,0.001]	[-0.248,0.089,0.04]
Volume (mm^3)	104.034	97.612	86.259	74.722
Surface area (mm^2)	158.263	150.094	159.742	146.105
Principal inertia values (mm^5)	[110,2085,2114]	[94,1888,1888]	[79,1753,1778]	[64,1359,1378]
Sand				
Sphericity*	0.881	-	0.806	0.782
Convexity	0.935		0.857	0.845
Flatness†	0.101		0.105	0.072
Elongation†	0.076		0.111	0.024
Centroid (mm)	[-0.04,0.558,0.538]	-	[-0.043,0.542,0.541]	[-0.048,-0.582,0.54]
Volume (mm^3)	65.235		54.486	52.458
Surface area (mm^2)	88.966		86.186	86.639
Principal inertia values (mm^5)	[158,176,193]		[118,133,149]	[113,124,137]

* Sphericity here corresponds to the degree of true sphericity as in Wadell (1932).
† The flatness and elongation indices were calculated as in Kong and Fonseca (2018).

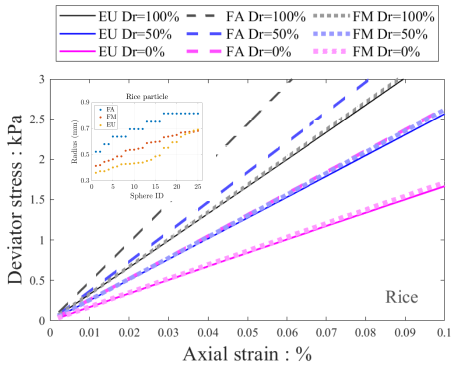
But, this method requires a large number of spheres per clump in order to achieve this minimisation of asperities, while for small numbers of spheres the method is not robust, as spheres are generated tangent to random points of the particle surface. On the other hand, the Euclidean distance-based method proposed in this chapter can generate representative clumps even using small numbers of spheres per clump, compared to the target particle geometry, as spheres are generated where mass is most under-represented in each iteration, but it does not mitigate the creation of artificial asperities like Ferellec and McDowell (2010) do. In the Euclidean method, every new sphere is smaller than the previous, allowing for a systematic prediction of the balance between morphological fidelity and computational efficiency. The examples and discussion demonstrated in this section highlight the strengths and weaknesses of the methods described, analysing the results through the prism of quantitative shape characterisation.



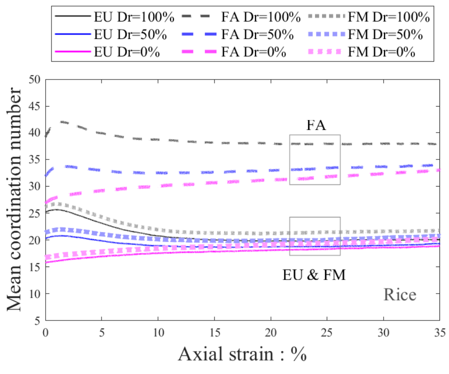
(a)



(b)

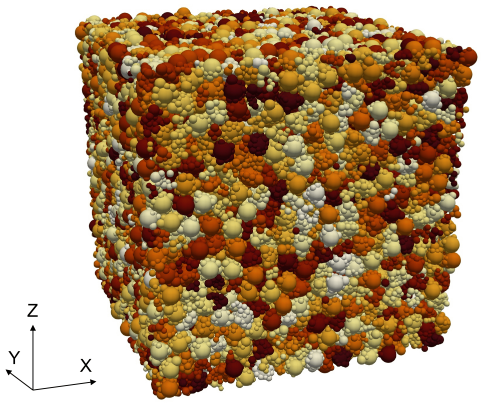


(c)

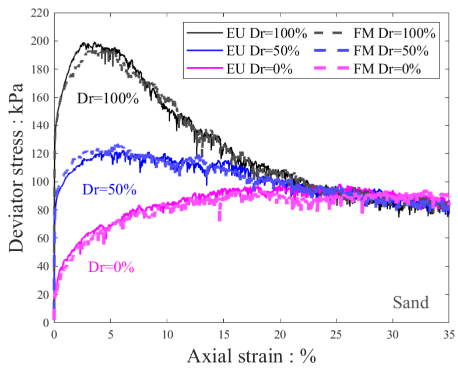


(d)

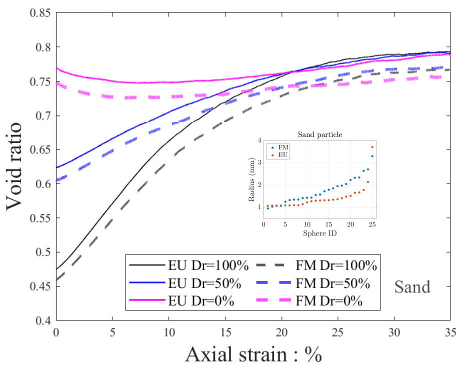
Figure 4.11: (a) Layout of triaxial test on rice grains generated using: (left) the Euclidean distance transform and (right) Favier et al. (1999); (b), (c) stress-strain behaviour of rice samples at three different relative densities, Dr = 0%, 50% and 100% (inset: radii of individual spheres); d) mean coordination number versus axial strain of the rice samples. EU, FA and FM stand for Euclidean transform, Favier et al. (1999) and Ferellec and McDowell (2010), respectively.



(a)



(b)



(c)

Figure 4.12: (a) Layout of triaxial test on sand grains generated with the Euclidean transform approach; (b) stress-strain behaviour of sand samples at three different relative densities, Dr = 0%, 50% and 100% (c) void ratio versus axial strain of the sand samples (inset: radii of individual spheres). EU and FM stand for Euclidean transform and Ferellec and McDowell (2010), respectively.

4.5 Software impact

`CLUMP` enables engineers, researchers and scientists from a variety of disciplines and industries to examine the dependency of their application to particle shape effects, through the generation of irregular particle shapes. Recent advances in granular physics demonstrate a clear need for consideration of particle shape in numerical modelling. In engineering practice, the clumped-sphere approach has been adopted in additive manufacturing, railway engineering, pharmaceuticals, *etc.* However, there is currently little evidence regarding how many spheres need to be generated. This is, mainly, due to the limited accessibility of open-source implementations of different clump-generation approaches. To this end, `CLUMP` offers two widely-used approaches to generate clumps and proposes a new one, aiming to provide an extendable library of available scripts. The implementations are computationally efficient, i.e. it takes few minutes to generate hundreds of spheres per grain.

`CLUMP` provides an option to extract the surfaces of the generated multi-sphere particles, allowing for the direct calculations of their shape descriptors, using available, open-source shape-characterisation codes. Comparing the morphological characteristics of the generated clumps to those of the original particle morphology, one can decide the number of spheres and method of application based on a quantitative and rigorous particle-generation approach.

4.6 Conclusions

This chapter presented `CLUMP`, a code library to approximate three-dimensional particle morphologies using different multi-sphere generation methods. The software allows exporting the results in formats compatible with several commercial and open-source DEM codes. `CLUMP` enables to assess the suitability of a clump-generation method for different applications and characterise the degree of manipulation of the original morphology in a quantitative manner.

The examples illustrated provide insight into the effect of the clump generation methods on the mechanical behaviour of two granular assemblies subjected to triaxial compression. In this chapter, it was found that the adoption of particles with near zero flatness leads to lower values of peak and residual stresses, while the adoption of larger contacting spheres leads to higher initial stiffness.

It is hoped that code developers will find the framework provided here useful so that other existing clump-generating approaches and future ones will be implemented too.

Chapter 5

Potential Particles and Potential Blocks

5.1 Introduction

This chapter discusses two codes to simulate (i) non-spherical particles using the concept of the Potential Particles (Houlsby, 2009), with the solution procedures in Boon et al. (2013) for 3-D, utilising convex optimisation techniques to perform contact detection and (ii) polyhedral blocks using planar linear inequalities, based on linear programming concepts (Boon et al., 2012). These codes define two shape classes in the open-source software YADE (Kozicki and Donze, 2008; Caulk et al., 2019), namely `PotentialParticle` and `PotentialBlock`. Besides some similarities in syntax, they have distinct differences, concerning morphological characteristics of the particles and the algorithms employed for contact detection.

Part of this chapter's content is sourced from the official manuals of YADE (Smilauer et al., 2021), where a dedicated documentation section[1] discussing these codes was written during this project.

The `Potential Particles` code (abbreviated herein as `PP`) is detailed in Boon et al. (2013), where non-spherical particles are assembled as a combination of 2nd degree polynomial functions and a fraction of a sphere, while their edges are rounded with a user-defined radius of curvature. The `Potential Blocks` code (abbreviated herein as `PB`) is used to simulate polyhedral particles with flat surfaces, based on the work of Boon et al. (2012), where a smooth, inner potential particle is used

[1] `https://www.yade-dem.org/doc/potentialparticles.html`

to calculate the contact normal vector. This code is compatible with the Block Generation algorithm defined in Boon et al. (2015a), in which `Potential Blocks` can be generated by intersections of an initial intact block with discontinuity planes.

These two codes are independent, in the sense that either one of them can be compiled/used separately, without enabling the other, while they do not interact with each other (i.e. currently, contact cannot be established between a PP and a PB). Enabling the PB code causes an automatic compilation of the Block Generation algorithm.

5.2 Potential Particles

The concept of Potential Particles was introduced and developed by Houlsby (2009). The problem of contact detection between a pair of potential particles was cast as a constrained optimisation problem, where the equations are solved using the Newton-Raphson method in 2-D. In Boon et al. (2013) it was extended to 3-D and more robust solutions were proposed, formulating contact detection as a Second Order Conic optimisation Problem (SOCP). Many numerical optimisation solvers generally cannot cope with discontinuities, ill-conditioned gradients (Jacobians) or curvatures (Hessians), and these obstacles were overcome in Boon et al. (2013), by re-formulating the problem and solving the equations using conic optimisation solvers. Previous versions of the code in the literature made use of MOSEK (using its academic licence), whereas currently an in-house code written by Boon et al. (2013) is used to solve the conic optimisation problem. A potential particle is defined as in Equation 5.1 (Houlsby, 2009; Boon et al., 2013, in 2-D and 3-D, respectively):

$$f = (1-k)\left(\sum_{i=1}^{N}\langle a_i x + b_i y + c_i z - d_i\rangle^2 - r^2\right) + k(x^2+y^2+z^2-R^2) \tag{5.1}$$

where:

(a_i, b_i, c_i)	= is the normal vector of the i^{th} plane in local particle coordinates;
d_i	= is the distance of the plane to the local origin;
$\langle\ \rangle$	= are Macaulay brackets, i.e. , $\langle x\rangle = x$ for $x > 0$; $\langle x\rangle = 0$ for $x \leq 0$.

The planes are assembled such that their normal vectors point outwards. They are summed quadratically and expanded by a distance r, which is also related to the radius of the curvature at the corners. Furthermore, a 'shadow' spherical particle is added; R is the radius of the sphere, with $0 < k \leq 1$, denoting the fraction of sphericity of the particle. The geometry of some cuboidal potential particles is displayed in Figure 5.1, for different values of the parameter k.

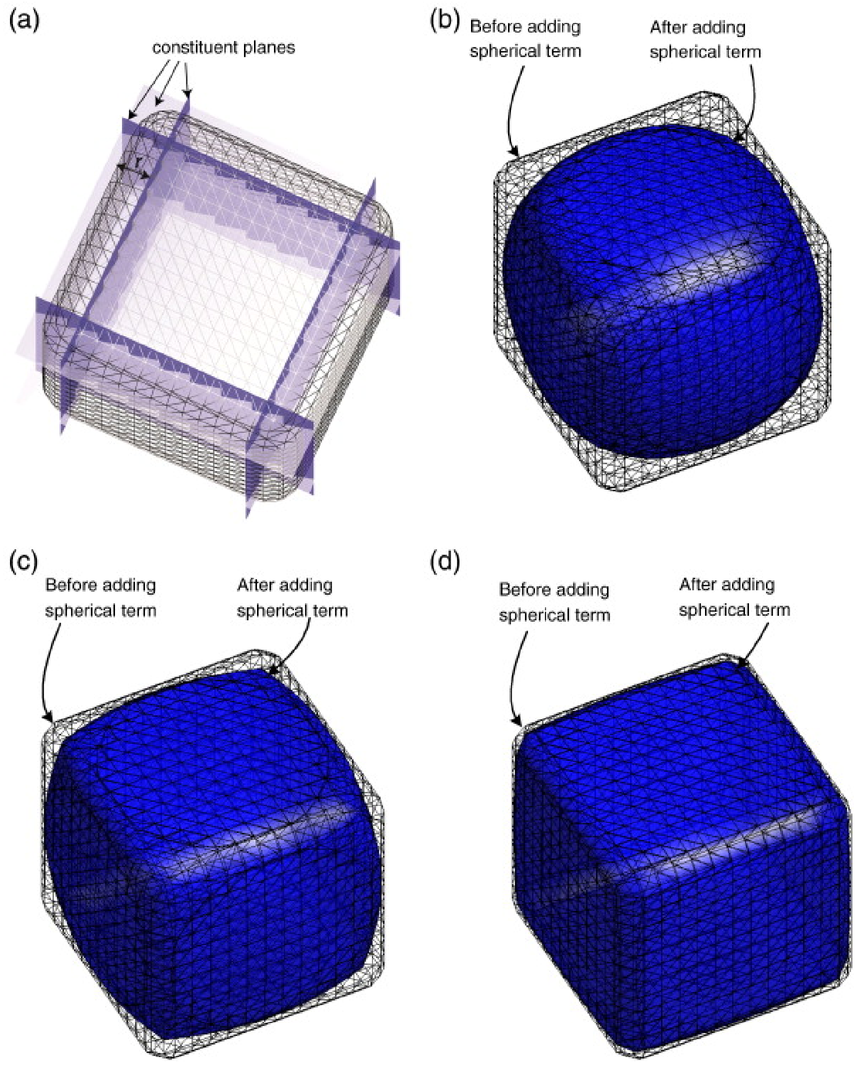

(a)
constituent planes
r
(b)
Before adding spherical term
After adding spherical term
(c)
Before adding spherical term
After adding spherical term
(d)
Before adding spherical term
After adding spherical term

Figure 5.1: Construction of potential particles (a) constituent planes are squared and expanded by a constant r. A fraction of sphere is added. Particles with the spherical term are visible in (b) k=0.9, (c) k=0.7, and (d) k=0.4 (after Boon et al., 2013).

The potential function is normalized for computational reasons in the form of Equation 5.2 (Houlsby, 2009):

$$f = (1-k)\left(\sum_{i=1}^{N} \frac{\langle a_i x + b_i y + c_i z - d_i \rangle^2}{r^2} - 1\right) + k\left(\frac{x^2+y^2+z^2}{R^2} - 1\right) \tag{5.2}$$

This potential function takes values:

- $f = 0$: on the particle surface;
- $f < 0$: inside the particle;
- $f > 0$: outside the particle.

To ensure numerical stability, it is not advised to use values approaching $k \approx 0$. In particular, the extreme value $k = 0$ cannot be used from a theoretical standpoint, since the `Potential Particles` were formulated for strictly convex shapes (curved faces).

5.3 Potential Blocks

The Potential Blocks code was developed during the D.Phil. thesis of CW Boon (Boon, 2013) and discussed in Boon et al. (2012). It was developed originally for rock engineering applications, to model polygonal and polyhedral blocks with flat surfaces. The blocks are defined with linear inequalities only and unlike the `PotentialParticle` shape class, no spherical term is considered .

For a convex particle defined by N planes, the space that it occupies can be defined using the following inequalities of Equation 5.3:

$$a_i x + b_i y + c_i z \ \leq \ d_i, i = 1 : N \tag{5.3}$$

where (a_i, b_i, c_i) and d_i defined as in the previous section. Figure 5.2 illustrates a Potential Block.

According to Boon et al. (2012), an inner, smooth potential particle is used to calculate the contact normal, formulated as in Equation 5.4:

$$f = \sum_{i=1}^{N} \langle a_i x + b_i y + c_i z - d_i + r \rangle^2 \tag{5.4}$$

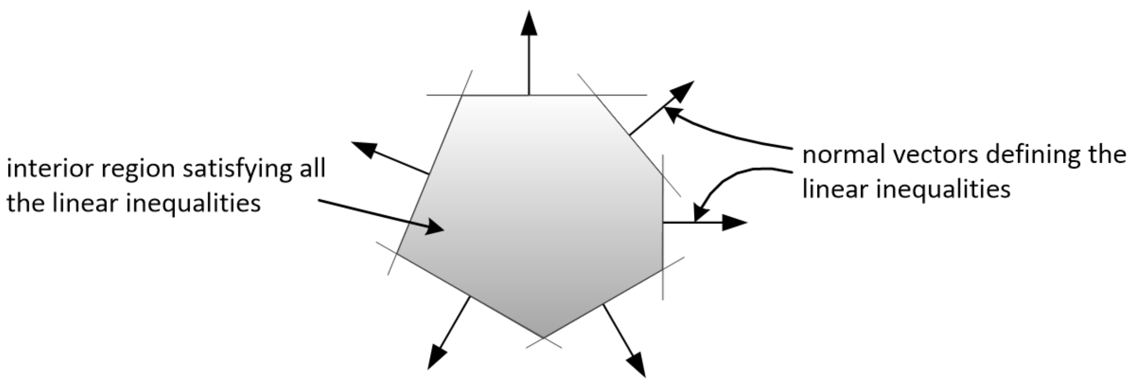

interior region satisfying all the linear inequalities
normal vectors defining the linear inequalities

Figure 5.2: A potential block. The normal vectors of the faces point outwards (after Boon, 2013)

This potential particle is defined inner by a distance r inside the actual particle, with edges rounded by a radius of curvature $\approx r$ (equal to r for corners meeting at a right angle), as well (see Figure 5.3). This inner potential particle is a shrinked, rounded version of the actual particle, with rounded edges and corners, so that the calculation of the contact normal becomes robust even for vertex-face and vertex-vertex contacts. The contact normal is calculated as the value of the gradient of the inner potential particle at the contact point.

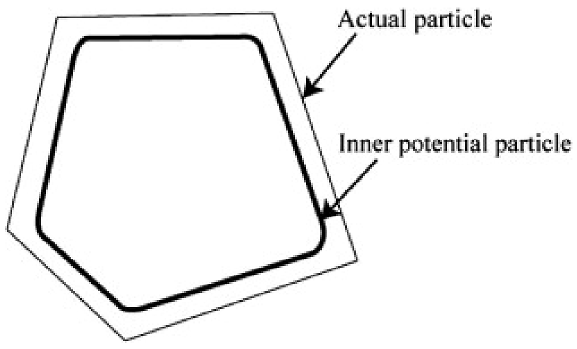


Figure 5.3: A potential particle is defined inside the actual particle. The normal vector of the particle at any point can be calculated from the first derivative of the potential particle (after Boon et al., 2012).

An implementation detail: In YADE, the Potential Blocks have a slightly different mathematical expression, since their shape is generated as an assembly of planes as in Equation 5.5:

$$a_i x + b_i y + c_i z - d_i - r = 0, i = 1 : N \tag{5.5}$$

while the inner Potential Particle used to calculate the contact normal is defined as in Equation 5.6:

$$f = \sum_{i=1}^{N} \langle a_i x + b_i y + c_i z - d_i \rangle^2. \tag{5.6}$$

Now, the Potential Block surface is at a distance of $(d_i + r)$ from the local particle center, while the inner potential particle is at a distance d from the local particle center.

It is worth emphasising that the shape of a Potential Block is defined as the convex, common space, bounded by a group of planes corresponding to the particle faces and not by a single, implicit potential function, like in the Potential Particles code. The inner potential particle in the Potential Blocks code is only used to calculate the contact normal vectors.

The problem of establishing intersection between a pair of blocks is cast as a standard linear programming problem of finding a feasible region which satisfies all the linear inequalities defining both blocks. The contact point is calculated as the analytic centre of the feasible region (see Figure 5.4), i.e. the inequalities defining the interior of both particles, inwards from their faces, a well-known concept of interior-point methods in convex optimisation calculations, found using linear programming. From a geometrical point of view, the analytic center is the point which maximizes the product of the distance among the considered planes, a property also demonstrated by the centroid of solid bodies.

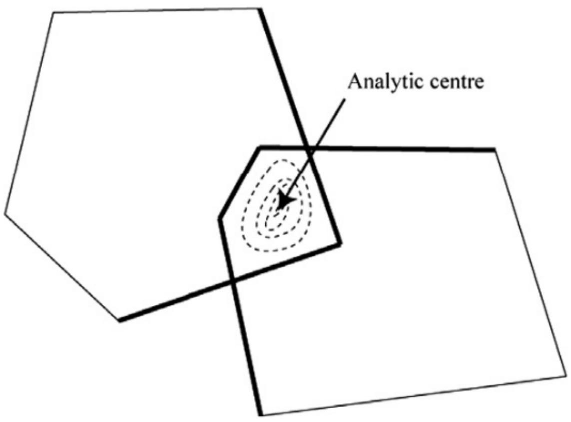


Figure 5.4: The analytic center for the inequalities defining both particles. The size of the overlap region is exaggerated for clarity (after Boon et al., 2012).

The linear programming solver for Potential Blocks was originally CPLEX, but was updated (in Boon, 2013) to CLP, developed by COIN-OR, since the latter can be downloaded from Ubuntu or Debian's distributions without requiring an academic licence.

5.4 Calculation of contact forces

5.4.1 Normal contact force

The normal contact force between two contacting `Potential Blocks` or `Potential Particles` is calculated as:

$$\mathbf{F_n} = k_n \cdot (u_n - \delta_0) \cdot \mathbf{n} = k_{n,vol} \cdot A_c \cdot (u_n - \delta_0) \cdot \mathbf{n} \tag{5.7}$$

where:

$\mathbf{F_n}$	= the normal component of the contact force (units: force);
$k_{n,vol}$	= the normal volumetric stiffness coefficient (units: stress/distance);
A_c	= the contact area (units: length^2);
u_n	= the relative displacement along the normal direction (units: length);
δ_0	= gap (units: length);
$\mathbf{n}$	= the contact normal vector.

The relative normal displacement is calculated heuristically, using a bracketed bisection search algorithm along the contact normal direction, to find two opposite points on the overlap region, starting from the contact point, as shown in Figure 5.5. The distance between these points is considered to be the relative normal displacement. It should be noted that these codes follow the *soft-particle approach* to calculate contact forces (Cundall and Strack, 1979), i.e. the particles are treated as rigid and any deformations during a contact are expressed by equivalent, phenomenal overlaps of the contacting region.

The gap δ_0 represents an initial gap, offering the possibility to simulate contacts with strength in tension. When $u_n > \delta_0$ the tension is compressive, otherwise when $0 < u_n < \delta_0$ the contact is under tension. This feature is not used in the simulations performed in this thesis, where all contact forces were compressive.

The normal stiffness of each contact (units: force/distance) is thus $k_n = k_{n,vol} \cdot A_c$, where A_c is updated in every time-step. The contact area is calculated using a heuristic algorithm to geometrically detect points of the overlap volume, searching along the contact shear direction. In essence, the contact area is calculated as the

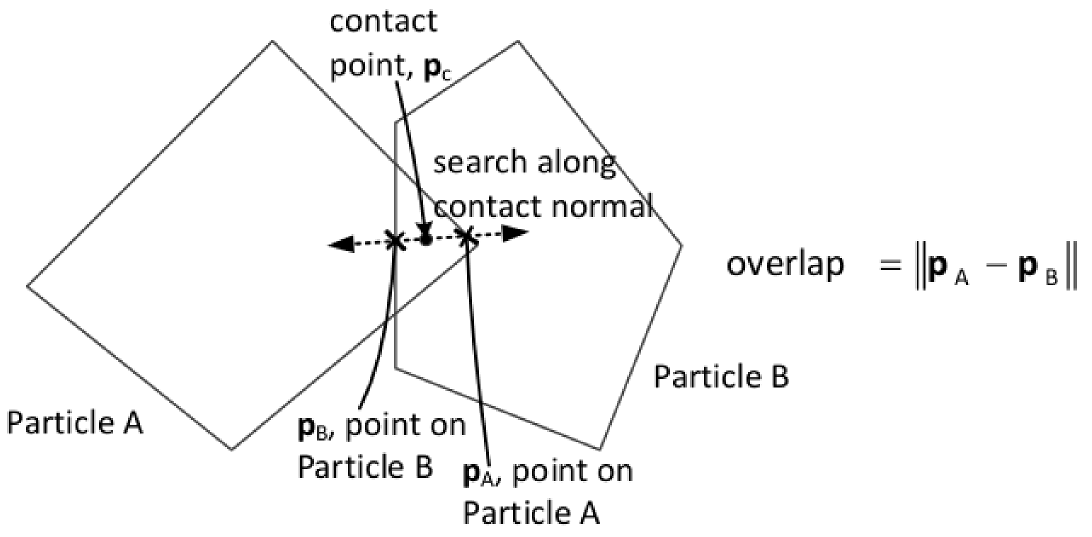

contact point, $\mathbf{p}_c$
search along contact normal
overlap $= \|\mathbf{p}_A - \mathbf{p}_B\|$
Particle B
Particle A
$\mathbf{p}_B$, point on Particle B
$\mathbf{p}_A$, point on Particle A

Figure 5.5: Illustration of the calculation of relative normal displacement. The overlap is exaggerated for clarity (after Boon, 2013).

area of a 2D slice of the 3D overlap region along the shear direction, passing from the contact point. For the `Potential Blocks`, this was developed in Boon (2013), as shown in Figure 5.6, where the contact area was traced on the boundaries of the overlap region, along the contact plane, until a closed loop was found.

For the `Potential Particles`, where the particle faces, edges and corners are rounded, this tracing technique is not applicable, as it only works for regions with planar boundaries. Instead, an algorithm was developed in-house as part of this thesis, to trace the boundaries of any convex region which does not rely on its boundaries being planar. This algorithm was developed to offer the choice of a non-linear contact law, with increasing stiffness for increasing stress levels, in simulations using the potential particles, as this is the experimental behaviour seen in the literature for the interactions of granular materials (e.g. Nadimi and Fonseca, 2017a). This algorithm detects points of the overlap region along the contact plane and starting from the contact point, by considering trace vectors which pass from the contact point and are pointing to direction around 360° for equal angle steps. Connecting the contact point with pairs of sequential points of the contact area boundary forms triangles, which add up to a convex polygon, the area of which (equal to the cumulative surface area of all associated triangles) is equal to the contact area for the current time-step. This computational approach to calculate the contact area for convex contact regions with any shape of boundaries, including curved boundaries,

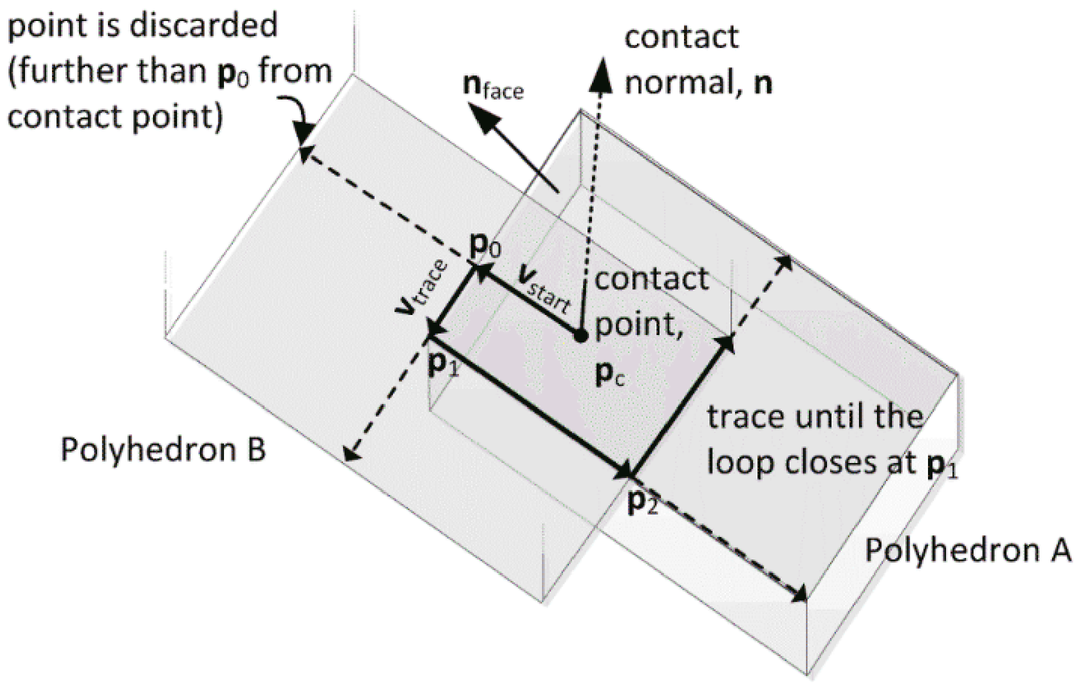
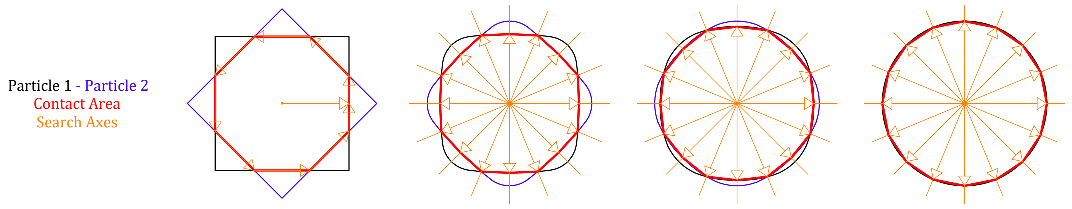

point is discarded (further than $\mathbf{p}_0$ from contact point)
contact normal, $\mathbf{n}$
$\mathbf{n}_{face}$
$\mathbf{p}_0$
$\mathbf{v}_{trace}$
$\mathbf{v}_{start}$
contact point, $\mathbf{p}_c$
$\mathbf{p}_1$
$\mathbf{p}_2$
trace until the loop closes at $\mathbf{p}_1$
Polyhedron B
Polyhedron A

Figure 5.6: Illustration of the calculation of contact area between two polyhedron in 3-D (after Boon, 2013).

is illustrated in Figure 5.7 along with an application of the method proposed by Boon (2013). This approach is subject to discretisation error (unline the method of Boon (2013) for regions with planar boundaries), while the user is in control of how many angle steps to consider, with a finer discretisation of the contact area leading to a more accurate but computationally more expensive calculation.

Figure 5.7 demonstrates four simple case studies calculating the area of contact regions with planar boundaries, using the computation method of Boon (2013), and rounded boundaries, including one for a perfectly spherical contact region, using the method proposed in this section. The first case demonstrates a contact area made of two rectangular sections of two contacting `Potential Blocks`, where the exact contact area is calculated using the algorithm of Boon (2013). The second and third example demonstrate two rounded contact regions, resulting from two contacting `Potential Particles`, the area of which is calculated using the proposed method. Last, the calculation of a perfectly spherical contact area is demonstrated, as the `Potential Particles` can be used to simulate perfect spheres for a value $k = 1$. It becomes evident that the more rounded the cross section of the contact region, the larger the induced discretisation error of the proposed method, although it is kept at a maximum of $\approx 2.55\%$ when considering 16 angle steps, to approximate the boundaries of the contact region, which is a small underestimation of its actual values, considering the amount of other ambiguities present in a discrete element simulation of non-spherical particles.



Particle Shape	Potential Blocks	Potential Particles	Potential Particles	Potential Particles
Exact Contact Area	82.843 mm^2	88.027 mm^2	106.034 mm^2	118.838 mm^2
Calculated Contact Area	82.843 mm^2	87.463 mm^2	103.991 mm^2	115.807 mm^2
Error: (Calc-Exact)/Exact	0	-0.64%	-1.92%	-2.55%
Angle step (for PPs only)		22.5°	22.5°	22.5°
No of Points on Boundary	10	16	16	16

Figure 5.7: Illustration of the calculation of contact area for contact regions with boundaries of various shapes; (left) planar; (middle two) rounded; (right) spherical.

It should be noted, that up till now, and during the work of Boon (2013), the `Potential Blocks` only supported the non-linear contact law discussed in this section and the `Potential Particles` only supported a linear contact law, considering only relative normal displacement (and not contact area). During this project, the contact law of the `Potential Blocks` was expanded as part of this thesis to also support a linear contact model, by setting the contact area $A_c = 1.0$ in Equation 5.7, while the contact law of the `Potential Particles` was modified to take into account the contact area, leading to non-linear stiffness, with the algorithmic implementations discussed in this section. This is controlled in a simulation via a boolean parameter in the YADE scripts calculating characteristics of the contact geometry, for these codes, entitled `calContactArea`.

5.4.2 Normal viscous force

When viscous damping is employed, the normal viscous force is calculated as:

$$\mathbf{F_{n,viscous}} = c_n \cdot \mathbf{v_n} \tag{5.8}$$

where:

$\mathbf{F_{n,viscous}}$	= the normal viscous force;
$\mathbf{v_n}$	= the normal component of the relative velocity of the particles;
c_n	$= 2 \cdot \beta_n \cdot \sqrt{m \cdot k_n}$ the critical damping coefficient;
β_n	= the normal viscous damping ratio;
m	$= \frac{m_1 \cdot m_2}{m_1 + m_2}$ the equivalent mass of the two particles in contact;
k_n	$= k_{n,vol} \cdot A_c$ the normal stiffness (units: force/distance);

An upper limit has been set for the viscous normal force, so that its magnitude does not exceed the one of the actual contact force.

The total normal force is calculated as $F_n = F_{n,el} + F_{n,viscous}$. The normal viscous damping force can exceed the elastic contact force, resulting in a negative total normal force. This would create an unphysical contact situation, where the particles are attracting each other during an otherwise compressive collision (as reported in Modenese, 2013). This check was implemented in the source code of the contact laws of the `Potential Particles` and `Potential Blocks` codes in YADE as part of this thesis, by introducing a boolean variable `allowViscousAttraction`, in order to avoid unnatural attraction between the particles, when $F_{n,viscous} > F_{n,elastic}$. It has been observed that unphysical attraction can be an unduly byproduct of using viscous damping (Antypov and Elliott, 2011) usually near the end stages of a contact, when the relative normal displacement and thus the elastic contact force values are relatively small. Because of this, Schwager and Pöschel (2007) concluded that attractive forces should be excluded by cutting off the interaction force, and considered the end of a contact not as the moment of zero relative normal displacement, but as the moment of zero acceleration. At the moment, this boolean variable leaves this modelling choice, of whether to set an upper limit for the viscous normal force, for the user/modeller to decide.

5.4.3 Shear contact force

The shear contact force during a collision of two `Potential Blocks` is assumed elastic-perfectly plastic, following the DEM mainstream incremental implementation of Hart et al. (1988). In Hart et al. (1988), to account for the incremental rotation of the contact plane, the existing shear force vector F_s is corrected as:

$$\mathbf{F^s_{i\_rotated}} = \mathbf{F^s_i} - \mathbf{F^s_i} \times \mathbf{n_{old}} \times \mathbf{n} \tag{5.9}$$

where:

$\mathbf{n_{old}}$ = the previous normal vector;
$\mathbf{n}$ = the current normal vector;

The increment of the shear force vector $\varDelta F_s$ before yield is calculated as:

$$\mathbf{\Delta F^s} = -k_{s,vol} \cdot A_c \cdot \varDelta \mathbf{u_s} \tag{5.10}$$

so that the total shear force vector F^s can be updated as:

$$\mathbf{F^s_{i+1}} = \mathbf{F^s_{i\_rotated}} + \mathbf{\Delta F^s} \tag{5.11}$$

where:

$\varDelta \mathbf{u_s}$ = the incremental relative shear displacement (units: length);
k_n = $k_{n,vol} \cdot A_c$ the normal stiffness (units: force/distance).

The magnitude of shear force is given by:

$$F^s = \sqrt{\mathbf{F^s} \cdot \mathbf{F^s}} \tag{5.12}$$

The maximum absolute value of shear force $F_{s,max}$ is defined employing the Mohr-Coulomb strength criterion, so that:

$$F_{s,max} = F_n \cdot \tan(\phi) \tag{5.13}$$

If $F^s > F^s_{max}$, the shear force is reduced to its plastic limiting value:

$$\mathbf{F^s_{max}} = \mathbf{F^s} \cdot \frac{F^s_{max}}{F^s} \tag{5.14}$$

At the moment, these codes do not consider a shear viscous damping force, even if viscous damping is applied in the normal contact direction, to avoid interference of damping with the sliding of contacts, as in Boon (2013). This is a delicate point, worth revisiting, as Thornton et al. (2013) have reported that *the inclusion of a dashpot in the normal force calculation but not in the tangential force calculation is not a valid method of dissipating energy in order to obtain values of en < 1*, a conclusion they reached though for spherical particles.

5.5 Compliance with periodic boundaries

In the original works of Boon et al. (2012, 2013, 2015a,b); Boon (2013), there was no algorithmic cooperation between the `Potential Particles` and `Potential Blocks` and the periodic boundaries of YADE. So, this feature was developed as part of this thesis. To achieve the aims of this thesis, the contact detection routines and the contact laws of the `Potential Particles` and the `Potential Blocks` were modified to comply with the periodic boundary conditions, which were already implemented inside the source code of YADE.

In particular, to achieve contact detection of particles in a periodic domain, a *shift* parameter is employed, which monitors the position of each particle and counts how many cells it has moved away from the initial, reference periodic cell. According to the value of this parameter, preliminary contact detection is performed between candidate particles that are in close proximity, across cells, using the sweep and prune of algorithm already existing in YADE. The accurate contact detection routines of Boon et al. (2012, 2013) were also modified to account for this $shift$ in the position of the contacting particles. This shifting parameter is used instead of modifying the actual position of the particle, in order to avoid abrupt changes in position and in order to avoid missing contacts, when particles are exactly crossing the boundary, and thus they appear in two opposite sides of the periodic cell.

In addition to achieving contact detection, the contact laws of the `Potential Particles` and the `Potential Blocks` were updated to handle velocity gradient fields imposed on the periodic box. Deformations are introduced in a periodic box as a uniform velocity gradient field ∇v, applied on all particles. As a result, a *shifting velocity* is considered, accounting for the gradient velocity field and the current size of the box, which is added to the velocity of all particles.

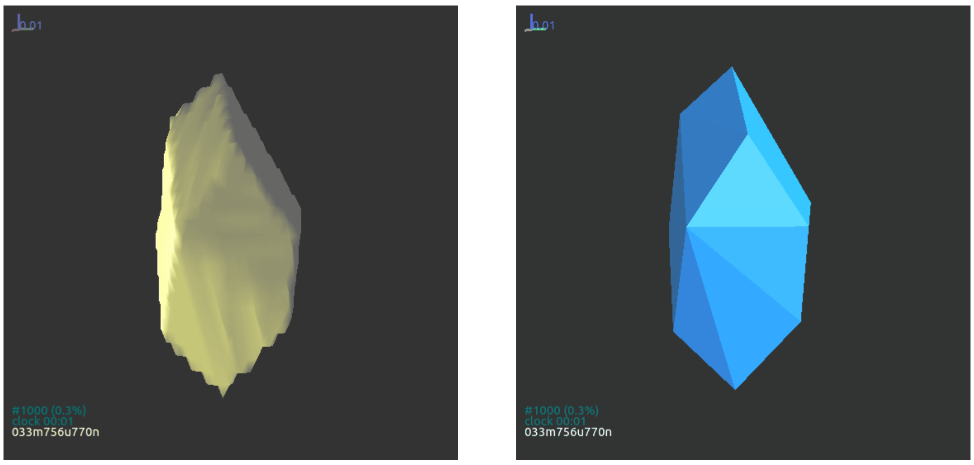
Developing these algorithmic features was crucial to this project, since periodic boundaries can be used to investigate material behaviour using an ideal elementary test, which focuses on the material alone, without any influence of boundaries. This is achieved through the repetition of a single reference cell, rid of boundary effects, which introduce unnecessary and unphysical anisotropy in elementary tests, such as triaxial compression tests Thornton (2000). By avoiding to induce such boundary effects in the DEM simulations, the size of a Representative Element Volume can be reduced significantly (while remaining representative), reducing thus also the total number of contacts in each time-step of the simulation. This downsizing of the numerical models, while keeping them representative and free of boundary-induced anisotropy can be transformative towards achieving efficient and trustworthy DEM simulations, especially when using non-spherical particles, in which case the computational cost of performing contact detection is much larger than the alterative of using spheres.

5.6 Visualisation and output of particles

The source code of YADE is written in its majority in C++, while its many modules have been wrapped in Python, offering modelling versatility, a friendly user interface and exposure of most simulation parameters to the user level. This allows for the user to inquire most model parameters during and after a simulation, enhancing transparency of the results.

YADE offers 3D rendering based on QGLViewer. Although this is not meant to be a full post-processing unit, it is an extremely helpful feature while building a model, as the user has the ability to view the model as it is being built, without needing to run simulation steps and rely to external, third-party output programs, e.g. in VTK format, utilising tools such as Paraview. Visualisation of the geometry of a particle is implemented in YADE through a series of OpenGL functors, one for each existing particle shape class. Until this doctoral project, the `Potential Blocks` code did not have a proper OpenGL functor. A draft functor existed, which mimicked the functor of the `Potential Particles`, which used the Marching Cubes algorithm, to reconstruct the particle surface. This solution resulted in a faulty visualisation of the `Potential Blocks`, as it could not provide efficient and geometrically accurate rendering of its planar faces and sharp edges and corners. In a nutshell, the Marching Cubes algorithm considers a 3D grid, the points of which are evaluated based on an implicit potential function. Although this is an appropriate reconstruction solution for the `Potential Particles`, which are mathematically described by such an implicit function, the same cannot be said for the piece-wise, sharp, polyhedral geometry of the `Potential Blocks`. In addition, the consideration of such a drawing grid introduces a step-like artificial roughness on particle faces, which is only minimised by creating a finer grid. Also, 3D grids are extremely memory demanding, and when some thousand particles are present in a simulation, their computational cost becomes unaffordable.

As a result, a code developed by Boon et al. (2015a) to calculate the particle vertices from the given equations of the particle faces, and to triangulate the particle surface was utilised and optimised, as part of this thesis. This triangulation was saved in the shape class of the `Potential Blocks` code, and used to define a new OpenGL functor, which offers computationally light and robust visualisation of these particles. The old and new visualisation of an irregular Potential Block is shown in Figure 5.8



(a) (b)

Figure 5.8: An irregular potential block (a) existing visualisation algorithm; (b) new visualisation algorithm developed as part of this thesis.

A similar issue of high-memory demand and non-accurate geometric representation existed with the function to export the geometry of `Potential Blocks` in VTK format, where a custom-made function was developed by Boon (2013), based on 3D grids, like above. VTK is a popular format for the post-processing of results from numerical simulations of this nature. A new lightweight function was developed as part of this thesis to export the triangulation of `Potential Blocks`, within the `export` Python module of YADE, entitled `exportPotentialBlocks`, which is less memory intensive, generates output files of the minimal possible size and exports the exact geometry of the particles of interest.

5.7 Practical examples

This section attempts to introduce the reader to practical aspects of building models using the `Potential Particles` and the `Potential Blocks` in YADE.

5.7.1 Shape definition of a PP and a PB

A strong merit of the Potential Particles and the Potential Blocks codes lies in the fact that the geometric definition of the particle shape and the contact detection problem are resolved using only the equations of the faces of the particles. In this way, using a single data structure, there is no need to store information about the vertices or their connectivity to establish contact, a feature that makes them computationally affordable, while all contacts are handled in the same way (there is no need to distinguish among face-face, face-edge, face-vertex, edge-edge, edge-vertex or vertex-vertex contacts). Due to this, the geometry of a particle is defined in the shape class using the values of the normal vectors of the faces and the distances of the faces from the local origin.

As a result of this, there is no need to triangulate the particle surface, as each face can have any convex polygonal shape. For example, to define a cuboid (6 faces) with rounded edges, an edge length of D, centred to its local centroid and aligned to its principal axes using the Potential Particles code, the following lines of code suffice to define the geometry of the particle:

```
r=D/5.  # radius of edges
k=0.3   # degree of spherical term
R=D/2.  # radius of spherical term
b=Body()    # a,b,c,d: plane coefficients of faces
b.shape=PotentialParticle( r=r, k=k, R=R,
      a=[   1.0,    -1.0,     0.0,     0.0,     0.0,     0.0],
      b=[   0.0,     0.0,     1.0,    -1.0,     0.0,     0.0],
      c=[   0.0,     0.0,     0.0,     0.0,     1.0,    -1.0],
      d=[D/2.-r,  D/2.-r,  D/2.-r,  D/2.-r,  D/2.-r,  D/2.-r], ...)
```

Using the Potential Particles code, this is not a perfect cube, since the particle geometry is defined by a potential function as in Equation 5.2. Note that within this potential function, these planes are summed quadratically, the particle edges are rounded by a radius of curvature r and then the particle faces are curved by the addition of a `shadow` spherical particle with a radius R, to a percentage defined by the parameter k. A value r is deducted from each element of the vector parameter

d, to compensate for expanding the potential particle by r in Equation 5.2.

The parameters a_i, b_i, c_i, d_i stated above correspond to the planes used in Equation 5.5:

$$
\begin{aligned}
1.0x + 0.0y + 0.0z = D/2 &\Rightarrow +x = D/2 \\
-1.0x + 0.0y + 0.0z = D/2 &\Rightarrow -x = D/2 \\
0.0x + 1.0y + 0.0z = D/2 &\Rightarrow +y = D/2 \\
0.0x - 1.0y + 0.0z = D/2 &\Rightarrow -y = D/2 \\
0.0x + 0.0y + 1.0z = D/2 &\Rightarrow +z = D/2 \\
0.0x + 0.0y - 1.0z = D/2 &\Rightarrow -z = D/2
\end{aligned}
\tag{5.15}
$$

The following lines of code are used to model a cube with an edge of D using the Potential Blocks code:

```
r=D/5.  # radius of inner potential particle
b=Body()    # a,b,c,d: plane coefficients of faces
b.shape=PotentialBlock( r=r,
        a=[   1.0,    -1.0,     0.0,     0.0,     0.0,     0.0],
        b=[   0.0,     0.0,     1.0,    -1.0,     0.0,     0.0],
        c=[   0.0,     0.0,     0.0,     0.0,     1.0,    -1.0],
        d=[D/2.-r,  D/2.-r,  D/2.-r,  D/2.-r,  D/2.-r,  D/2.-r], ...)
```

Using the Potential Blocks code, this particle will have sharp edges and flat faces in what regards its geometry (i.e. the space it occupies), defined by the given planes, while for the calculation of the contact normal, an inner potential particle with rounded edges is used, formulated as in Equation 5.6, located fully inside the actual particle. The distances of the planes from the local origin, stored in the vector parameter d, are reduced by r to achieve an exact edge length of D, using Equation 5.5. The value of r must be sufficiently small, so that $d_{min} - r > 0$, while it should be sufficiently large, to allow for a proper calculation of the gradient of the inner Potential Particle at the contact point. A recommended value is $r \approx 0.5 * d_{min}$, as it ensures that $d_{min} - r > 0$ is positive in all cases, and thus the value of this parameter can be decided in an automated manner; this automation is especially valuable if multiple particles are imported in YADE.

To ensure that the d_i plane coefficients represent a distance in the local particle coordinate system, it is advised to normalize the normal vector of each plane, so that ${a_i}^2 + {b_i}^2 + {c_i}^2 = 1$. There is no limit to the number of the particle faces that can be used, a feature that allows the modelling of a variety of convex particle shapes.

In practice, it is usual for the geometry of a particle to be given in terms of vertices & their connectivity (e.g. in the form of a surface mesh, like in .stl files). In such cases, the user can calculate the normal vector of each face, which will give the coefficients a_i, b_i, c_i and using a vertex of each face, then calculate the coefficients d_i. This functionality has been developed within `SHAPE` (Angelidakis et al., 2021b).

5.7.2 Body definition of a PP and a PB

To define a body (term interchangeable with "particle" in YADE) using the `PotentialParticle` or `PotentialBlock` shape classes, it has to be assembled using the `_commonBodySetup` function, which can be found in the file `py/utils.py` of YADE. For example, to define a `PotentialParticle` body using YADE's Python interface:

```python
O.materials.append(FrictMat(frictionAngle=radians(30), density=2650,
↪  label='frictMat'))
b=Body()
b.shape=PotentialParticle(...)
b.aspherical=True # To be used in conjunction with
↪  exactAsphericalRot=True in the NewtonIntegrator
# V: Volume
# I11, I22, I33: Principal inertias
utils._commonBodySetup(b,V,Vector3(I11,I22,I33), material='frictMat',
↪  pos=(0,0,0), fixed=False)
b.state.pos=Vector3(xPos,yPos,zPos)
b.state.ori=Quaternion((random.random(), random.random(),
↪  random.random()), random.random())
b.shape.volume=V;
O.bodies.append(b)
```

The `PotentialParticle` must be initially defined, so that the local axes coincide with its principal axes, for which the inertia tensor is diagonal. More specifically, the plane coefficients (a_i, b_i, c_i) defining the plane normals must be rotated, so that when the orientation of the particle is zero, the `PotentialParticle` is oriented to its principal axes. This is a common convention in YADE, to ensure that the reference orientations correspond to the principal inertial axes, a delicate yet significant point of interest, which guarantees that the integration of motions for aspherical particles happens in a local coordinate system where the inertia tensor is diagonal. This transformation to the principal directions is important for the correct calculation of particle rotations. It should be noted that the principal inertia values I_{11}, I_{22}, I_{33} mentioned here are divided by the density of the considered material, since they are multiplied with the density inside the `_commonBodySetup` function. The mass of the particle is calculated within the same function as well, so it does not need to be set manually `b.mass=V · density`.

For the Potential Particles code, there is currently no algorithm for the automatic calculation of volume and inertia, and so they have to be calculated manually by the user. For the Potential Blocks, an automatic calculation has been implemented during this project, for the volume and inertia tensor, so the user does not have to define the particle to its principal axes, since this is handled automatically within the source code, by an automatic routine developed as part of this thesis.

For example, to define a `PotentialBlock` body, the following lines can be defined using the Python interface of YADE:

```python
O.materials.append(FrictMat(frictionAngle=radians(30), density=2650,
 ↪  label='frictMat'))
b=Body()
b.shape=PotentialBlock(...)
b.aspherical=True # To be used in conjunction with
 ↪  exactAsphericalRot=True in the NewtonIntegrator
utils._commonBodySetup(b,b.shape.volume,b.shape.inertia,
 ↪  material='frictMat', pos=Vector3(xPos,yPos,zPos), fixed=False)
b.state.ori=Quaternion((random.random(), random.random(),
 ↪  random.random()), random.random())
O.bodies.append(b)
```

More examples of both codes can be found in the folders /examples/Potential-Particles/ and /examples/PotentialBlocks/, in the source code repository of YADE, where the syntax of the codes is further demonstrated.

5.8 Energy calculations

This section discusses some energy calculations that have implemented in `YADE` as part of this thesis, using the `Potential Blocks` (Boon et al., 2012) and the `Potential Particles` (Boon et al., 2013) codes. These calculations follow conceptually similar ones found in the source code of `YADE` for spheres, with small adaptations. Energy calculations are important to check the correctness of a DEM code and ensure the employed contact law does not violate energy conservation. This is particularly important for the simulation of dynamic processes.

5.8.1 Energy types present in a simulation

This section lays out the different types of energy that develop during a typical discrete element simulation. During the contact of two particles, this entails: elastic potential (strain) energy, plastic dissipation due to friction and dissipation due to viscous damping. These energy calculations were developed for the Potential Particles and Potential Blocks codes as part of this thesis, following existing calculations in `YADE` of other classes of particle shape. In addition, the gravitational potential energy, the kinetic energy, the kinetic energy due to the velocity field applied by moving the periodic boundaries to impose a prescribed strain rate and energy dissipation due to local damping appear in these simulations, were already implemented in `YADE`. What follows is a brief description of these energies and their way of calculation.

5.8.1.1 Gravitational potential energy

Already implemented in `YADE`. The formulation can be found in this link.

5.8.1.2 Kinetic energy

Already implemented in `YADE`. The formulation can be found in this link.

5.8.1.3 Kinetic energy from moving the periodic boundaries

Already implemented in `YADE`. The formulation can be found in this link.

5.8.1.4 Dissipation due to local damping

Already implemented in `YADE`. The formulation can be found in this link.

5.8.1.5 Elastic stored strain energy

The elastic strain energy, stored in each contact, is calculated jointly for the normal and shear contact forces. This calculation takes place after applying the Mohr-Coulomb criterion, in order to use the updated shear contact forces.

$$elastPotential = \frac{1}{2}k_n {u_n}^2 + \frac{1}{2}k_s {u_{s,el}}^2 = \frac{1}{2}\left(\frac{{F_n}^2}{k_n} + \frac{{F_s}^2}{k_s}\right) \tag{5.16}$$

where:

u_n	$=$ the relative displacement along the normal direction;
$u_{s,el}$	$=$ the shear increment distance;
k_n	$= k_{n,vol} \cdot A_c$ the normal stiffness (units: force/distance);
k_s	$= k_{s,vol} \cdot A_c$ the shear stiffness (units: force/distance);
F_n	$=$ the current normal force;
F_s	$=$ the current shear force (after applying the M-C criterion).

5.8.1.6 Plastic dissipation due to friction

The plastic dissipation due to friction is calculated for a sliding contact as:

$$plastDissip = \frac{1}{k_s} \cdot (\mathbf{F_{s,trial}} - \mathbf{F_{s,max}}) \cdot \mathbf{F_{s,max}} \tag{5.17}$$

where:

k_s	$= k_{s,vol} \cdot A_c$ the shear stiffness (units: force/distance);
$F_{s,trial}$	$= k_{s,vol} \cdot A_c \cdot u_s$
$F_{s,max}$	$=$ the frictional limit from Mohr-Coulomb.

In particular, $\mathbf{F_{s,trial}}$ (shown in Figure 5.9) is the shear force before applying the Mohr-Coulomb criterion for the sliding contact at hand (i.e. $F_{s,trial} > F_{s,max}$). The plastic dissipation is monitored inside the code in a cumulative manner, by adding each new calculated value to the ones of the previous time-steps.

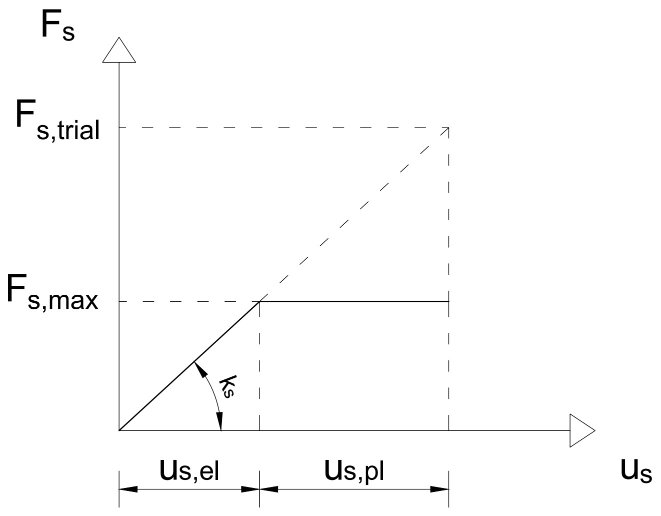


Figure 5.9: Shear forces and displacements before and after applying the Mohr-Coulomb criterion.

5.8.1.7 Ratio of sliding contacts and mobilised shear

Although this is not an energy calculation, it is included here, since it is monitored along with the plastic dissipation due to friction. This ratio can be used to assess the kinematic state of the particles during a simulation, by identifying what percentage of contacts is sliding. On each time-step, the ratio of the sliding contacts over the total number of real contacts can be monitored and assessed at a sample scale. Along with monitoring the percentage of sliding contacts, this index provides information of how close each contact is to sliding in a quantified manner as well, by considering the ratio:

$$mobilizedShear = \frac{F_{s,current}}{F_{s,max}} \tag{5.18}$$

where:

$F_{s,current}$ = the current shear force;
$F_{s,max}$ = the frictional limit force from Mohr-Coulomb.

5.8.1.8 Dissipation due to normal viscous damping force

If viscous damping is employed, the plastic dissipation due to the normal viscous force is calculated as:

$$viscousDissip = \mathbf{F_{n,viscous}} \cdot \mathbf{v_n} \cdot dt \tag{5.19}$$

where:

$F_{n,viscous}$	= the normal viscous damping force;
v_n	= the normal component of the relative velocity;
dt	= the time-step.

5.8.1.9 Dissipation due to shear viscous damping force

Currently, wezero dissipation due to viscous damping in the contact shear direction is considered.

5.8.2 Case studies

This section demonstrates a set of discrete element simulations of different contact scenarios for the collision of a free cuboid on a flat surface, modelled using the Potential Blocks code, which performs free fall (only gravity is applied with at-rest initial conditions), such as face-to-face, vertex-to-face and edge-to-face, varying different parameters of the simulation that affect the contact physics. The material and inertial parameters are kept equal among all simulations. These case studies aim to investigate if energy conservation is achieved in all cases and discuss potential causes if not. No local damping (proportional to particle acceleration) was considered for these simulations, as it is unphysical and unduly affects simulations of dynamic phenomena, like the free falls demonstrated in these examples.

The legend entries in the following figures stand for:

- `gravWork`: Work of gravity (gravitational potential energy);
- `kinTrans`: Translational kinetic energy;
- `kinRot`: Rotational kinetic energy;
- `elastPotential`: Elastic strain energy stored in each contact;
- `total`: Overall energy of the system, through superposition of all individual energy components;

- `normDampDissip`: Dissipation due to viscous damping of the contact normal damping;
- `plastDissip`: Dissipation due to plastic shear deformation (slip) of sliding contacts;

Figure 5.10 demonstrates a series of consequent face-to-face elastic contacts (no dissipation due to friction or viscous damping is considered), during which the energy of the system is conserved.

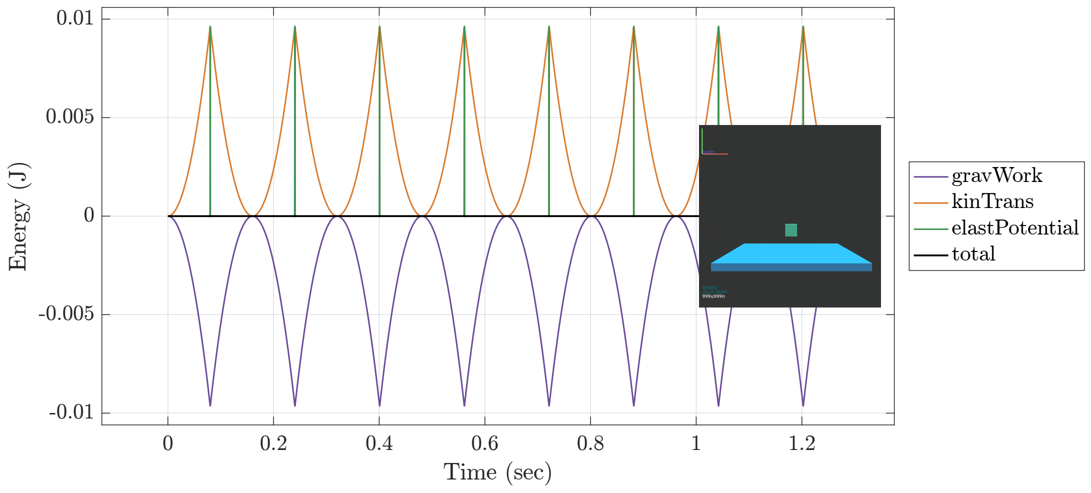


Figure 5.10: Face-face contact - No friction - No viscous damping (inset: initial orientation of moving particle).

Figure 5.11 demonstrates what happens to the same system of face-to-face collisions if a degree of viscous damping is introduced. The elastic energy of the system is gradually spent by the viscous dissipation, while the overall energy is again conserved.

Figure 5.12 demonstrates a series of vertex-to-face collisions of an elastic system, during which the free particle rotates excessively. After the first collision, the total energy fluctuates around its initial zero value but is not kept constant at all times.

Figure 5.13 demonstrates what happens to the same vertex-to-face collision system if plastic dissipation due to friction occurs. It becomes evident that the energy balance is not kept constant, as the gravity work equalises after 0.5sec, but the plastic dissipation keeps increasing due to sliding. This does not reflect natural behaviour for this dynamic setting.

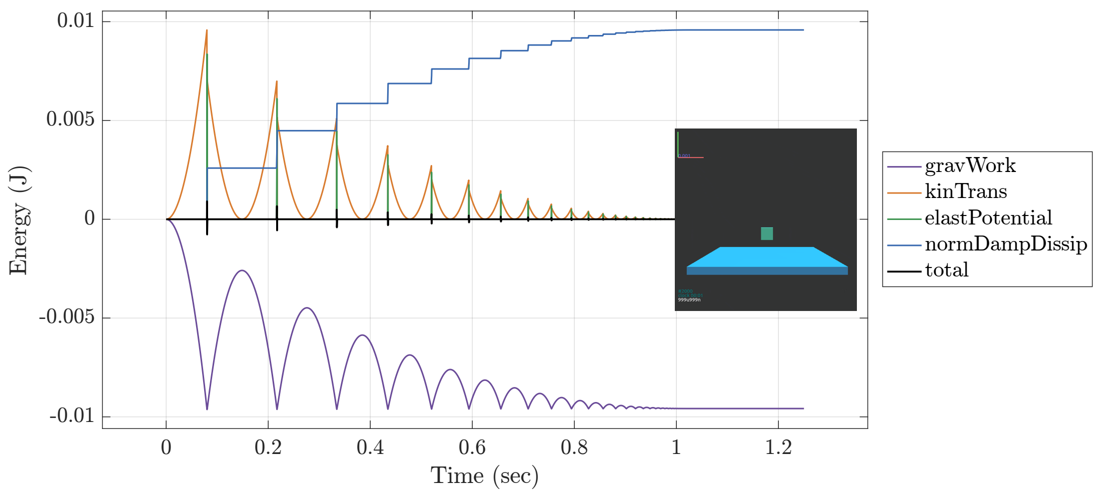


Figure 5.11: Face-face contact - No friction - Viscous damping ratio = 0.05 (inset: initial orientation of moving particle).

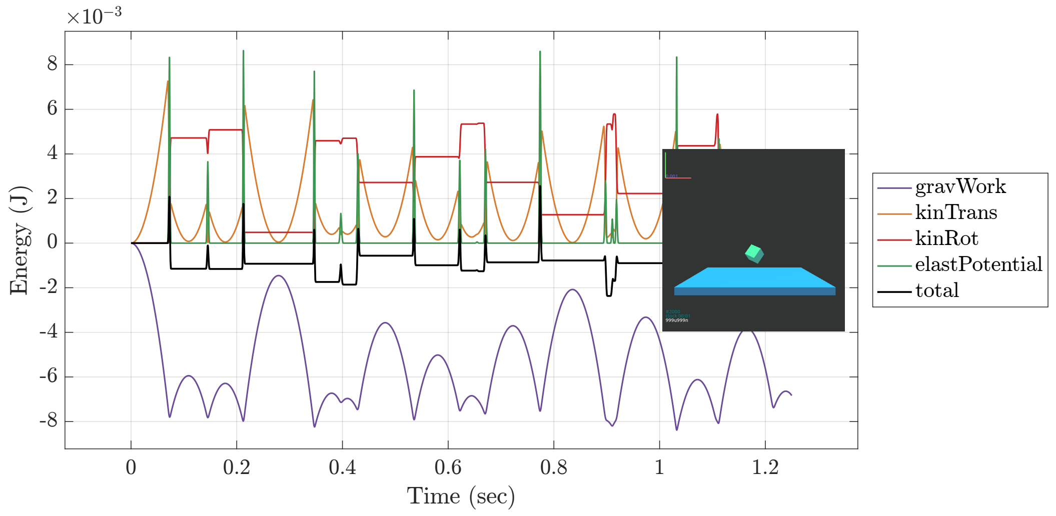


Figure 5.12: Vertex-face contact - No friction - No viscous damping.

Observing how irregular the motion of the vertex-to-face system was, it was attempted to simulate a simpler situation, of a perfect edge-to-face system, with a 45 ° rotation around the global Y axes, as visualised in the inset of Figure 5.14. After collision the free particle does not rotate, but keeps translating vertically, as the system is perfectly symmetric and the calculation of contact normals is robust enough to not allow rounding errors make the particle rotate. Though, Figure 5.14 makes it apparent that the energy is conserved before and after the contact, but

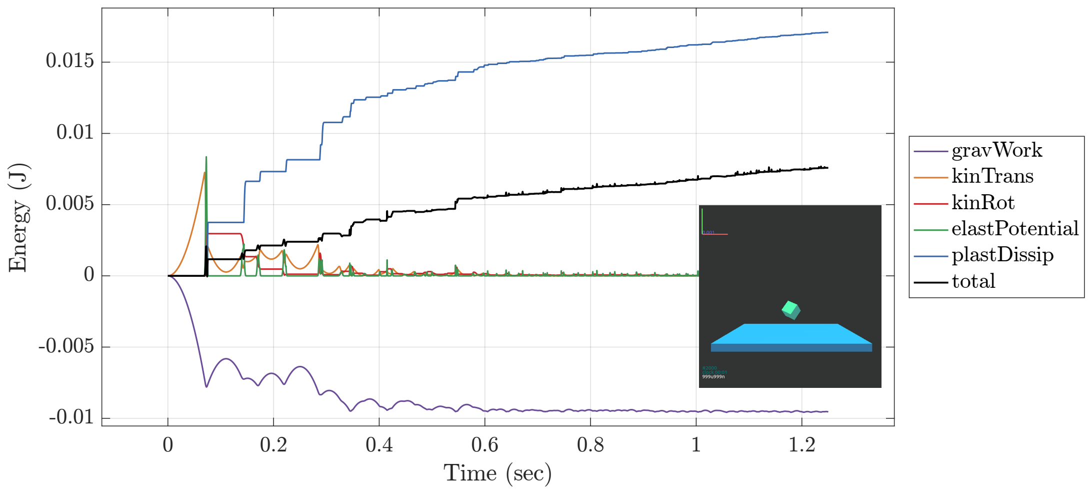


Figure 5.13: Vertex-face contact - Interparticle friction angle = 30° - No viscous damping.

not during the contact. This is attributed to the formulation of the current non-linear, overlap distance-based contact law used in the Potential Blocks code, as it provides an estimation of the volume of the contact region, rather than an accurate calculation of the overlap volume enclosed by the contacting polyhedra.

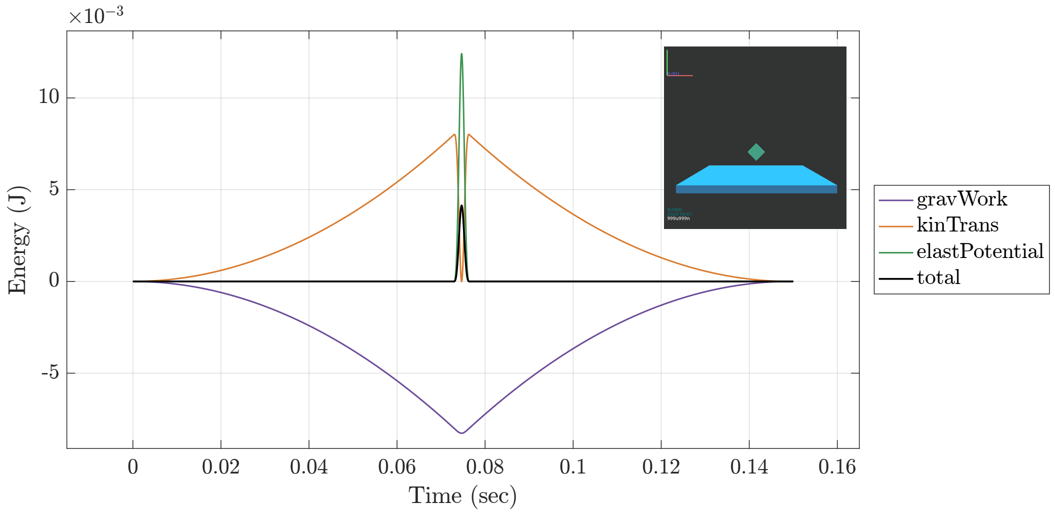


Figure 5.14: Edge-face contact - 45° initial rotation - No friction - No viscous damping.

Last, another edge-to-face scenario was tested for a 30 ° rotation of the particle around the global Y axis, aiming to make the free particle rotate post-contact. Figure 5.15 demonstrates that energy is no longer conserved after contact, when the particle starts to rotate. These case studies were tested for various time-step values, to ensure that this issue was not relating to the integration of motion.

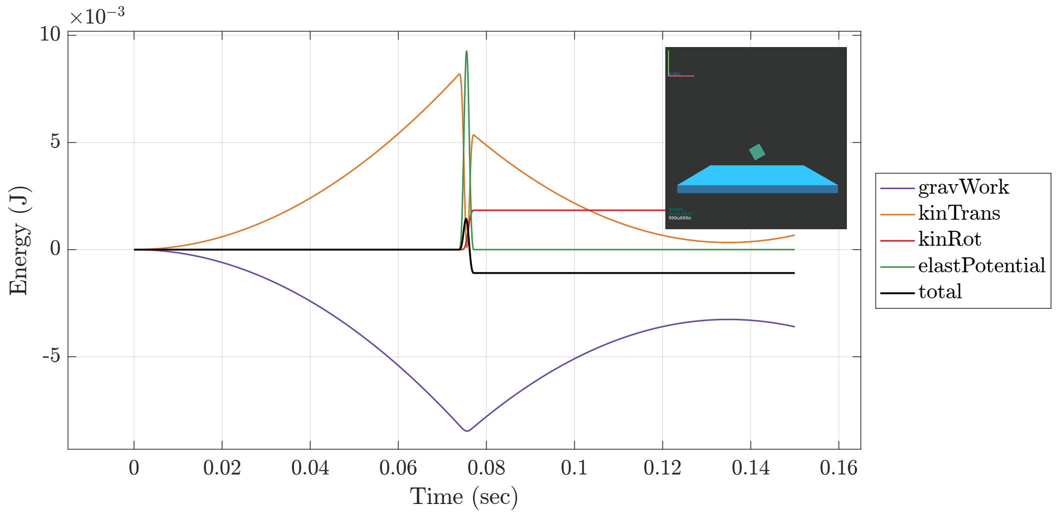


Figure 5.15: Edge-face contact - 30° initial rotation - No friction - No viscous damping.

Achieving energy conservation is not trivial for simulations involving contacts among non-spherical particles. Past literature had claimed that energy conservation cannot be achieved in discrete element simulations of non-spherical particles. In particular, Matuttis and Chen (2014, p.59) argue that "for many discrete element models with non-spherical particles, there is no energy conservation". They continue, by stating that "This often happens in particle simulations of collisions involving rotations of non-spherical particles: when the particles turn during the collision, the repulsive force on approach can be different from the force upon separation". This is the behaviour observed in these case studies, where the total energy is conserved only in the scenarios where no rotations occur. The severity of creating extra, unphysical, energy in a system with every new contact becomes more profound for simulations in a periodic cell, since there, small surges of energy generated by individual contacts propagate through the whole assembly of particles, in the absence of boundaries.

In recent years, theoretical formulations have been proposed in the literature, defining the characteristics of energy-conserving contact laws for non-spherical particles in DEM simulations (Feng et al., 2012), which consider the normal contact force to be a function of the overlap volume. Earlier this year, (Feng and Tan, 2021) demonstrated that energy conservation is possible also with overlap-distance based contact laws, and they used the Minskowski difference of simplices to achieve this. The Expanding Polytope Algorithm (EPA), a traditional pair to calculate contact forces when the Gilbert–Johnson–Keerthi (GJK) distance algorithm is used to perform contact detection, based on the Miskowski difference of polytopes also leads to energy calculation (as seen in Zhao and Zhao, 2021).

5.9 Simulating the Round Robin test of AOR with the Potential Particles

The Japanese domestic committee TC105 (Technical Committee 105: Geo-Mechanics from Micro to Macro) of the International Society for Soil Mechanics and Geotechnical Engineering (ISSMGE) organised in 2019 a round robin series of tests to assess the angle of repose (AOR) of an artificial, 3D-printed material (see Figure 5.16), through discrete element simulations [2]. This activity aimed to provide experimental benchmarks and to compare results from various research groups, using various Discrete Element codes. This section demonstrates results from participation in this Round Robin exercise using the Potential Particles, to model the particles of the material of interest. This was an opportunity to compare and validate results of the Potential Particles code, against the experimental benchmarks and against the results from the numerical simulations of the other participants.

The contribution of this particular modelling approach in this exercise, using a single Potential Particle to represent the target 3D-printed particle, aimed to offer an alternative approach in the simulation of non-spherical particles with rounded edges, like the ones used in the benchmark tests. With the Potential Particles, each physical particle can be simulated with one particle, instead of four spheres in a clump (an obvious choice because of its close resemblance to the physical particle shape), reducing the complexity of the model. To achieve this, some additional work was required at a pre-processing stage, compared to the clumped spheres approach, in order to identify the appropriate shape for the potential particle.

5.9.1 Experimental apparatuses

Several methods have been proposed in the literature to measure the angle of repose of granular materials (Geldart et al., 2006; Al-Hashemi and Al-Amoudi, 2018). In this Round Robin exercise, the angle of repose was measured using two experimental

[2] `https://www.issmge.org/news/tc105-dem-round-robin-test`

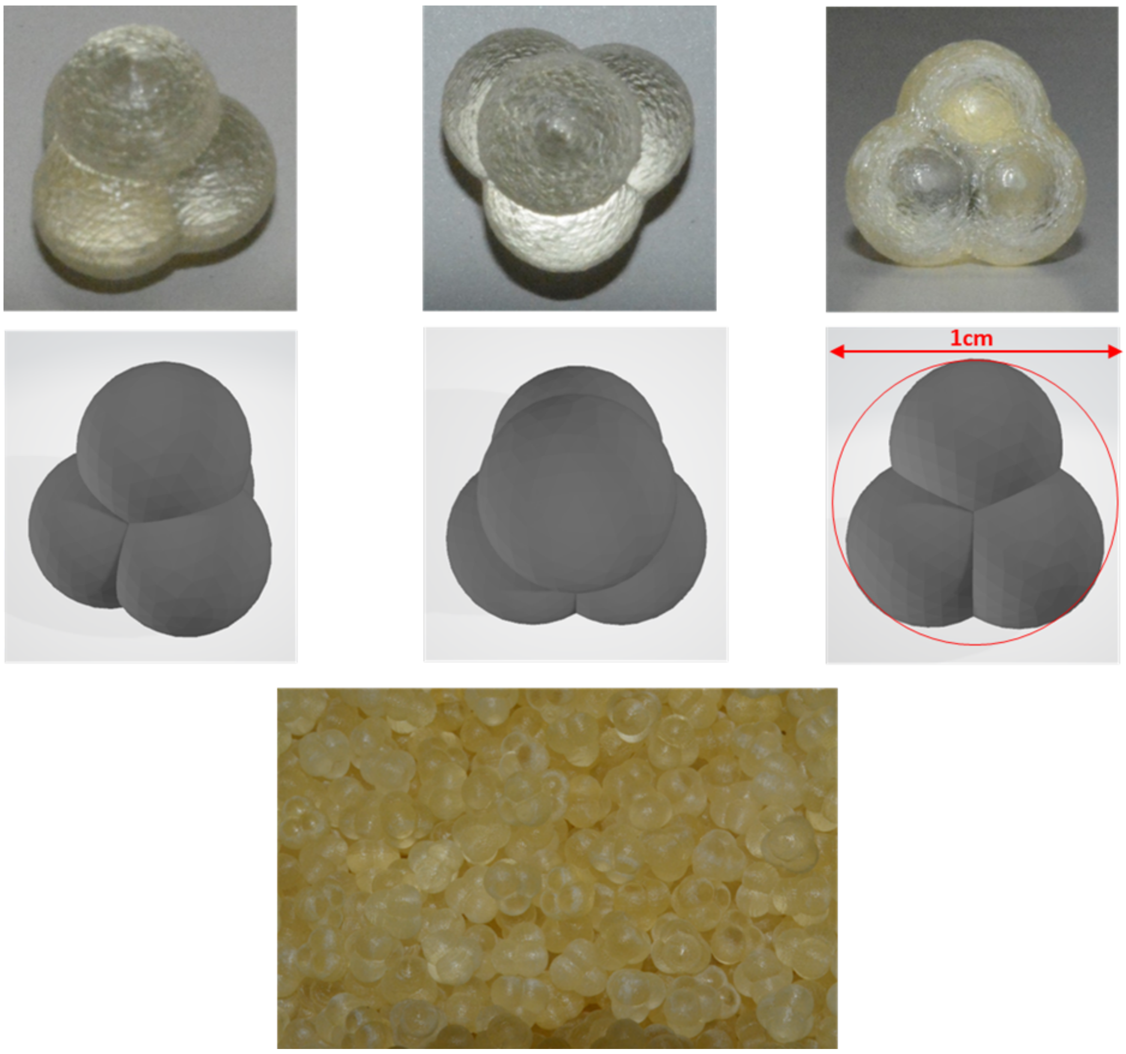
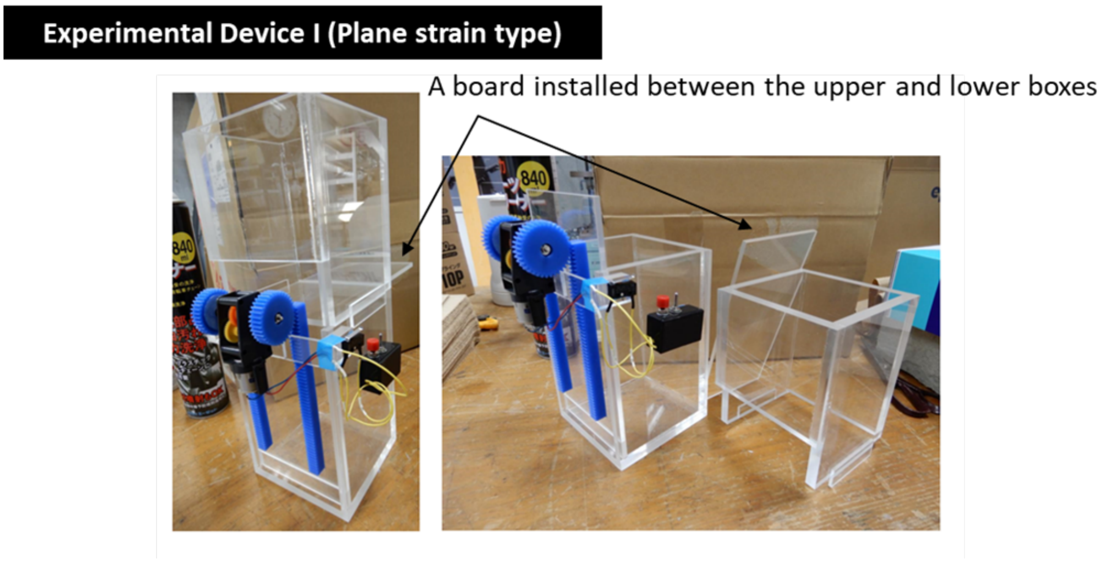
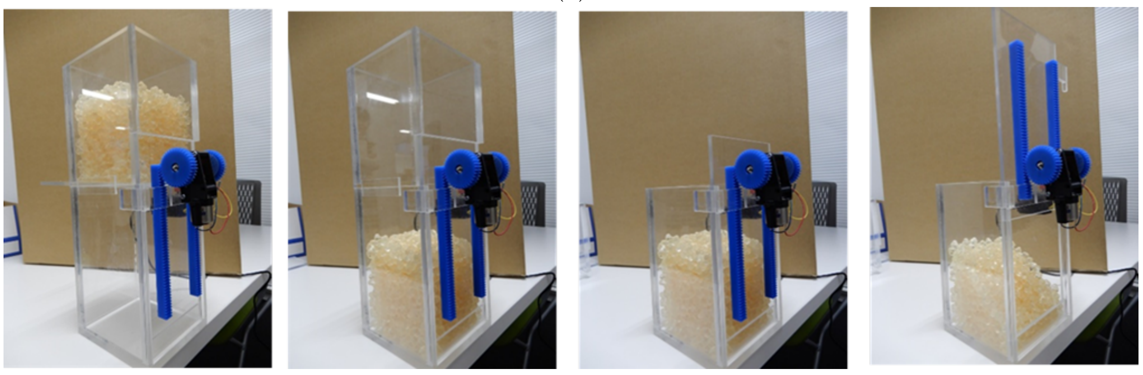


Figure 5.16: Experimental, 3D-printed material studied in this exercise (source: TC105, 2021).

devices, designed and manufactured by members of the Japanese domestic TC105. Repose was measured for two conditions: (i) plane strain and (ii) axial-symmetric, aiming to compare how the angle of repose varies for heaps of different shapes. Figure 5.17 demonstrates the "Experimental Device I" and the main steps to measure plane strain AOR formations. Figure 5.18 shows the setup of the "Experimental Device II" to measure axial-symmetric AOR, while Figure 5.19 shows the main steps of measurement.


Experimental Device I (Plane strain type)
A board installed between the upper and lower boxes

(a)



(b)

Figure 5.17: (a) Experimental setup and (b) test procedure of Device I, for plane-strain AOR (source: TC105, 2021).

5.9.2 Preliminary evaluation of this modelling approach

A brief discussion can be found below, listing the pros and cons of using the Potential Particles to simulate the particles used in this benchmark.

Pros:

- A single particle is used to simulate each physical particle, resulting to fewer collision checks during the simulation compared to using a clump of spheres (with four sphere-members for each physical particle).

- The drawbacks of using clumps will be avoided altogether, namely: overesti-

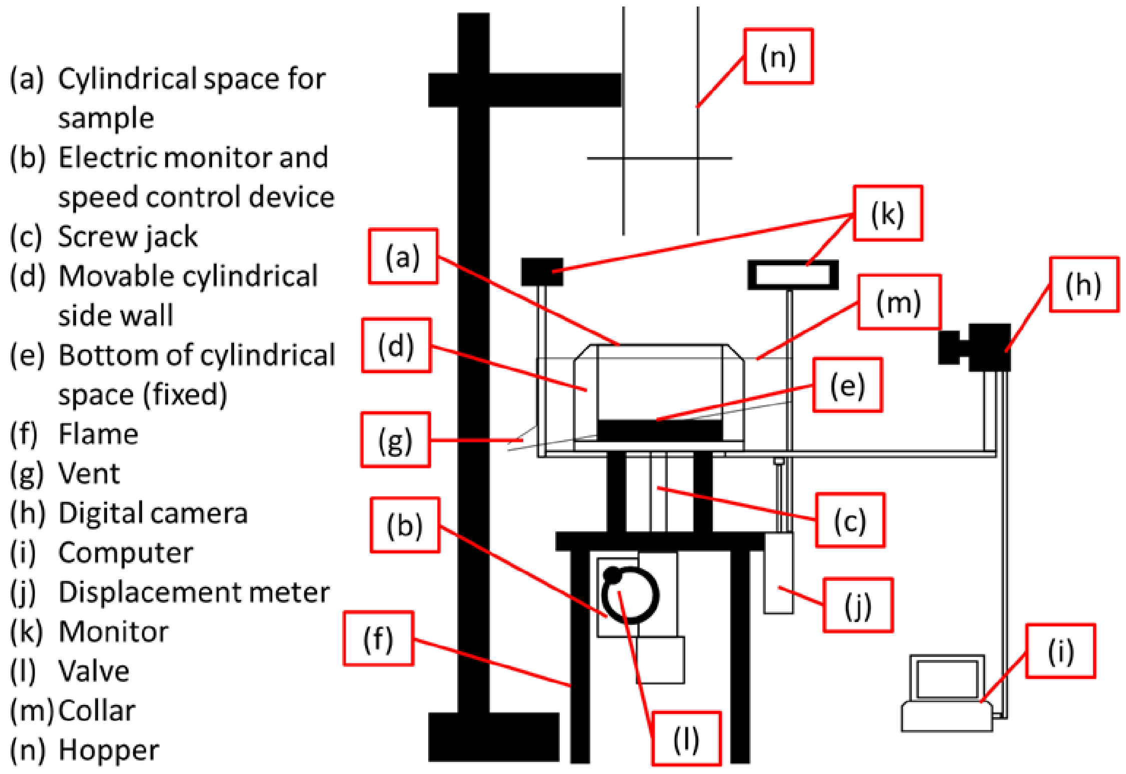


Figure 5.18: Experimental setup of Device II, for axial-symmetric AOR (source: TC105, 2021).

mated mass and inertia due to non-uniform density within each clump, because of the sphere-members overlapping.

Cons:

- The contact detection among Potential Particles is much more expensive from a computational standpoint, compared to contact detection among spheres.
- The concavity of the physical particle cannot be represented, underestimating their potential to interlock.
- The mass and inertia will still be overestimated in comparison to the physical particle, as the produced Potential Particle will enclose the physical particle (and thus has slightly larger volume). This can be mitigated with density adjustment.

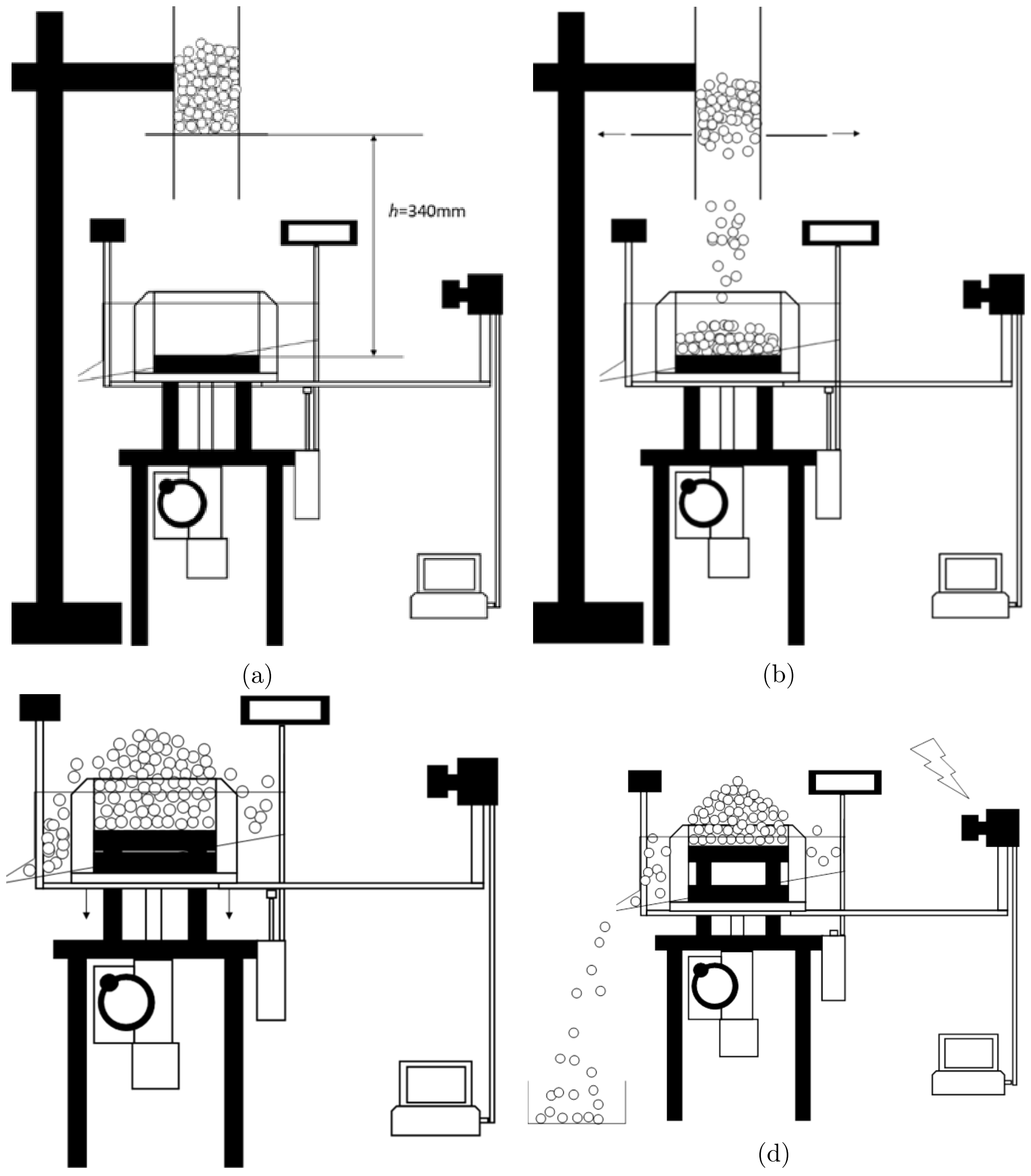


Figure 5.19: Measurement procedure for Device II: (a) deposition of material on upper cylinder; (b) air pluviation; (c) lowering of cylindrical sides; (d) reposed condition (source: TC105, 2021).

5.9.3 Approximating the given particle shape using Potential Particles

The given particle shape was approximated by a rounded tetrahedron, using the mathematical formulation of the Potential Particles. Following this approach, the particle shape is defined by an implicit function, which is assembled as a combination of quadratically summed planes and a “shadow” spherical particle, which controls the curvature of the faces.

To decide which planes to use in order to assemble the Potential Particle in this exercise, two criteria have to be satisfied: a physical one and a practical one, with the latter aiming to achieve post-processing convenience:

1. First and foremost, the main criterion is to capture the morphology of the physical particle as faithfully as possible. This means the generated Potential Particle has to adequately approximate the size, surface curvature, mass and inertia of the given physical particle. On a second level, shape descriptors such as the sphericity of the particle can be computed, to ensure that the selected particle shape is appropriate to describe the morphology of the physical particle.

2. Having post-processing in mind, where the participants of the exercise had to give as output the centres of the spheres (if clumped particles are employed), to evaluate the angle of repose, the potential particle must have a straightforward analogy to this format.

To achieve these criteria, the planes used to assemble the potential particle were chosen as the faces of the tetrahedron connecting the centroids of the spheres making the physical particle, as shown in Figure 5.20. In the Potential Particles formulation, these planes will be expanded and the edges will be rounded by a radius r, while the faces will be further curved to a degree specified by k and for a radius of the “shadow” particle R. This approach can be generalised to approximate any convex shape, given a tessellation of its surface, or a multi-sphere representation of a particle made of spheres with equal radii.

To match the local surface curvature of the physical particle, a radius $r = r_s$ was chosen, which controls the roundness of the edges and corners of the Potential Particle, where r_s the radius of each individual sphere making the clump-like rounded tetrahedral morphology of the physical particle. The radius of the shadow particle was assigned to $R = \sqrt{2} \cdot r_s$, to capture the curvature of faces of the given particle shape. The only parameter needed to be calibrated in order to match the given particle shape was the parameter k, which controls the curvature of the faces. For small values of k, infinitesimally approaching zero, the particle faces would be

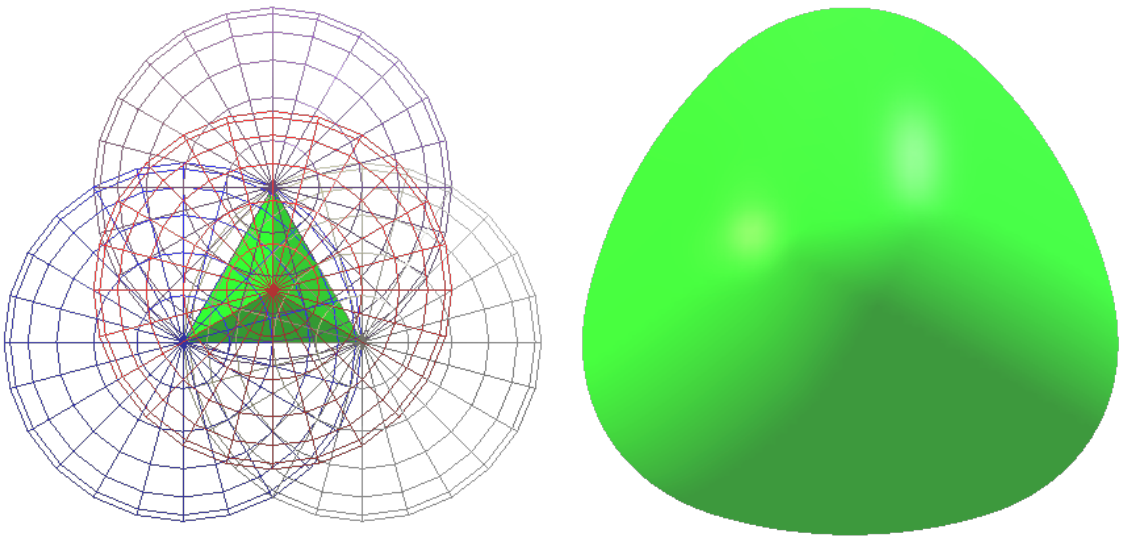
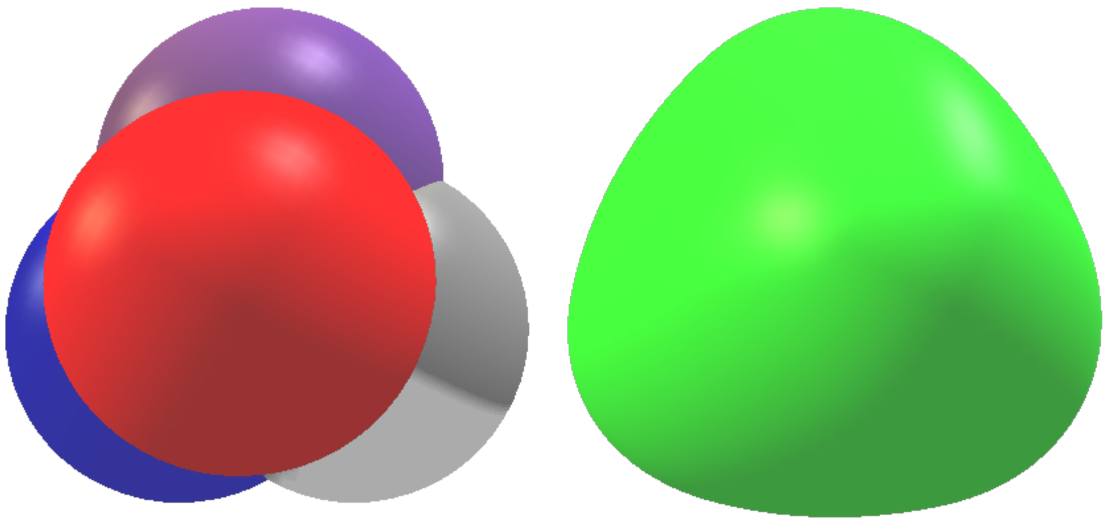


Figure 5.20: Deciding the planes used to construct the Potential Particle; (left) tetrahedron connecting the centres of the spheres of the clump; (right) Potential Particle assembled as a rounded, expanded version of the initially sharp tetrahedron.

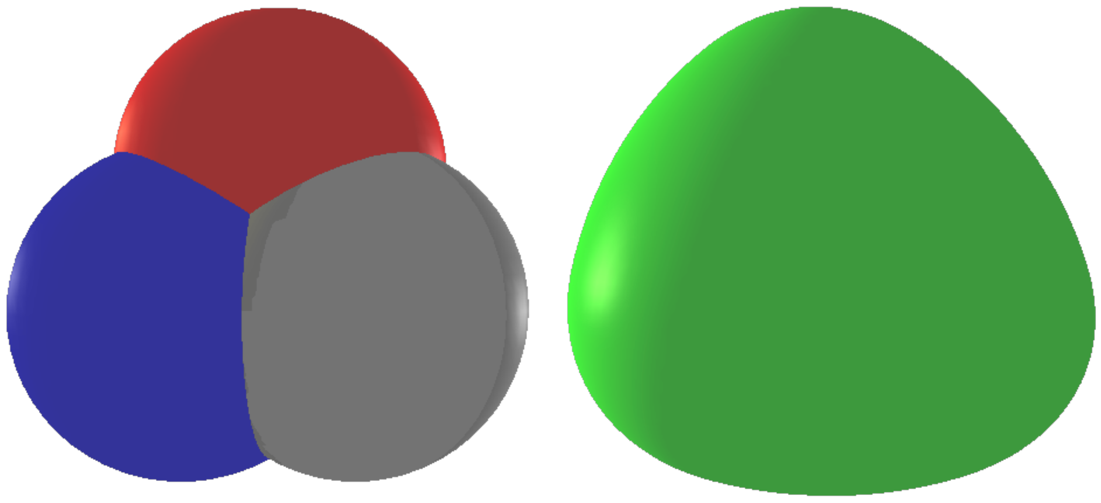
practically flat, while for values approximating unity, the potential particle would become perfectly spherical. A value of $k = 0.65$ led to a good match with the target geometry. The following Figure 5.21 and Figure 5.22 demonstrate visually the geometrical faithfulness of the generated Potential Particle to the shape of the real, physical particle.

5.9.4 Modelling versatility of this approach

In this exercise, the participants were given to simulate a granular material with non-spherical particles at two reposed states, using the devices for plane-strain and axial-symmetric boundary conditions, as in the experimental benchmarks. To simulate these problems, the participants were tasked to simulate the rounded, tetrahedral-like grains of the studied material, along with cuboidal elements of various sizes, making the moving and still parts of the two devices. It is reminded that Device I consists of an upper and lower box and a vertical moving plate, while Device II consists of a hollow-cylindrical hopper where the granular material is deposited initially, a horizontal moving plate and a lower, vertically movable hollow cylinder with boundaries of oblique ends. Using the Potential Particles in `YADE`, all the components of these models can be build using a single, unified approach, i.e. one



(a)



(b)

Figure 5.21: (a) Front and (b) back views of clump of spheres (left) and rounded tetrahedral particle using the Potential Particles (right).

code for all shapes, as shown in Figure 5.23.

5.9.5 Particle shape characterisation

As demonstrated in Figure 5.21 and Figure 5.22, the considered Potential Particle can approximate the morphology of the physical particle faithfully, as it can rep-

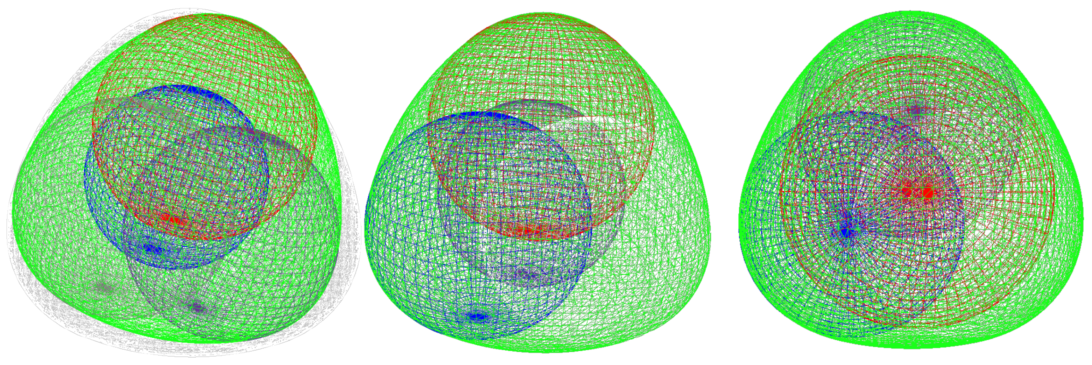


Figure 5.22: Overlapping clump of spheres and fitted tetrahedral Potential Particle, viewed from different perspectives.

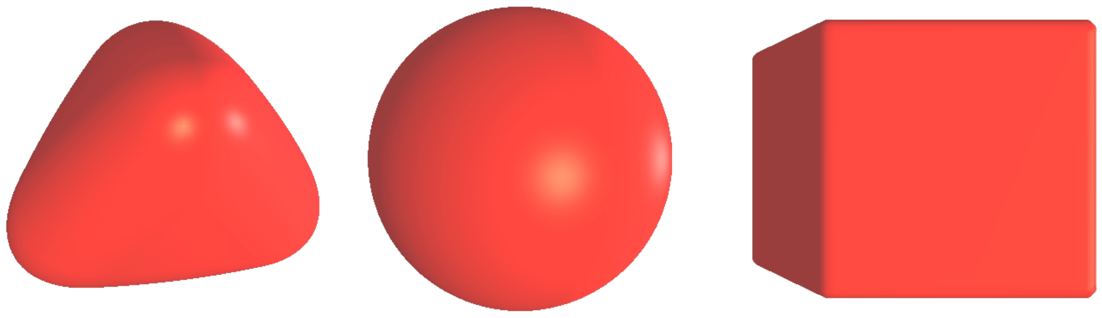


Figure 5.23: Definition of rounded tetrahedral, spherical and cuboidal particles using the Potential Particles.

resent the main dimensions of the particle, determining particle form, along and the curvatures of its edges/corners, relating to particle roundness. Albeit faithful enough, the produced Potential Particle cannot represent the concavity of the physical particle, and thus it is expected that the Potential Particles in the following simulations will interlock less than the physical material, potentially demonstrating a slightly smaller angle of repose. A quantitative characterisation of particle form was performed using SHAPE (Angelidakis et al., 2021b), to identify the fidelity between the Potential Particle used in the numerical simulations and the physical particle. The surface mesh of the physical particle was produced by defining it as a clump of 4 overlapping spheres and using the surface extraction module of CLUMP (Angelidakis et al., 2021a) to tessellate its surface.

A comparison of geometrical and morphological parameters is also offered with a clump of spheres, where the overlapping parts of the sphere-members are counted

Table 5.1: Parameters of particle form for physical particle, potential particle and clump of overlapping spheres, without correction for overlaps.

Shape characteristics	[1] Physical Particle	[2] Potential Particle	$\frac{[2]-[1]}{[1]}$	[3] Clump of Overlapping Spheres	$\frac{[3]-[1]}{[1]}$
Volume (m^3)	$3.3304 \cdot 10^{-7}$	$3.9248\ 10^{-7}$	17.85%	$4.9965 \cdot 10^{-7}$	50.03%
Surface area (m^2)	$2.491 \cdot 10^{-4}$	$2.632 \cdot 10^{-4}$	5.66%	$2.491 \cdot 10^{-4}$	0
Inertia tensor/ρ (m^5)	$\begin{bmatrix} 2.584 & 0 & 0 \\ 0 & 2.584 & 0 \\ 0 & 0 & 2.584 \end{bmatrix} \cdot 10^{-12}$	$\begin{bmatrix} 3.286 & 0 & 0 \\ 0 & 3.286 & 0 \\ 0 & 0 & 3.286 \end{bmatrix} \cdot 10^{-12}$	27.17%	$\begin{bmatrix} 3.123 & 0 & 0 \\ 0 & 3.123 & 0 \\ 0 & 0 & 3.123 \end{bmatrix} \cdot 10^{-12}$	20.86%
Convexity	0.954	1	4.82%	0.954	0
Sphericity	0.9328	0.9849	5.59%	0.9328	0

multiple times in the calculation of volume, centroid and inertia. This was carried out, as to date many DEM codes still calculate these overlapping parts multiple times when using clumps, causing an overestimation of the inertial characteristics (mass and inertia tensor) of clumps with overlapping sphere-members. In such cases, this can lead to particles with non-uniform density, as the density shows a spatial increase at areas where spheres overlap. To mitigate this, methods to adjust the density of each sphere-member have been proposed in the literature (such as Ferellec and McDowell, 2010) to correct mass and inertia. Modern discrete element codes, such as `YADE`, mitigate this in a more elegant way: A three-dimensional grid is considered enclosing the clump particle, and it is evaluated if each element (voxel) of the grid belongs to one of the sphere-members of the clump. These voxels are used to calculate particle mass and inertia, without the need for density correction. This approach is dependent on the resolution of the grid, i.e. a finer grid will produce more accurate results but will result in higher computational cost; however, this is a task performed once for each clump, upon initialisation, and has been implemented by the `YADE` developers to require minimal input from the user, who only needs to define the size of the grid, effectively minimising pre-processing efforts compared to applying density adjustment techniques.

Table 5.1 demonstrates the shape parameters of the physical particle, of the produced potential particle and of a clump where the geometrically overlapping parts of the particles are considered multiple times in the calculation of volume and geometric inertia (i.e. physical inertia divided by density).

As expected, the considered Potential Particle has larger values of volume and geometric inertia. To mitigate this, in some of the simulations, the density of the material was scaled down, multiplied with a scale factor equal to the ratio of the volume of the physical particle, over the volume of the Potential Particle, i.e.

$SF = 3.3304 \cdot 10^{-7}/3.9248 \cdot 10^{-7} = 0.849$. Translating this into the inertial properties of the Potential particles, this scaling brings the error of mass down to zero, while the error in inertia values drops from 27.17 % down to 7.96 % after scaling density. This is considered to be a better alternative than using clumps of overlapping spheres, as using the Potential Particle modelling approach guarantees uniform density distribution throughout the particle geometry. It is interesting to note that using overlapping spheres without some correction for uniform density leads to an error of 50.03 % for the volume and 20.86 % for the values of the principal inertia tensor.

5.9.6 Numerical simulations and results

The organisers of this round robin study performed a series of experiments using the material of interest, at the particle scale, to determine its mechanical properties. They provided the statistical distribution of each material parameter (TC105, 2021), and the average values of these distributions were considered in the numerical models described in this section. Table 5.2 collects the material properties considered in the Potential Particle simulations. Four cases of different material properties were considered, aiming to investigate their effect in the angle of repose for both devices.

The normal stiffness (k_n), the friction coefficients of the particles with other particles and with the walls (μ_{pp} and μ_{pw} respectively) and the normal viscous coefficient were considered constant across simulations, as shown in Table 5.2, based on the available experimental evidence. In particular, the normal viscous coefficient ($\beta_n = 0.071$) was calculated based on the average value of the experimentally-derived coefficient of restitution $en \approx 0.80$, via the relationship $\beta_n = -(\log e_n)/\sqrt{\pi^2 + (\log e_n)^2}$.

A small number of parameters was varied in a parametric manner, to investigate their effect on the measured angle of repose for both repose states, i.e. in Device I and Device II. One parameter was varied at a time, allowing for a direct comparison among cases with small differences. The shear-to-normal stiffness ratio (k_n/k_s) was not defined during the experimental mechanical characterisation of the 3D-printed material, and so three different cases were defined; first, using an objectively small value for the ratio ($k_s/k_n = 0.2$); second, a value equal to the Poisson ratio of the material ($k_s/k_n = \nu = 0.37$), a practice followed within the source code of `YADE` for the linear contact model (Smilauer et al., 2021); and third, a larger value based on the widely-accepted relationship ($k_s/k_n = (1-\nu)/(1-0.5\nu)$) derived analytically in Mindlin (1949), and also discussed in Cundall and Strack (1979) and Thornton et al. (2013). Following the discussion on particle shape in the previous section, two values were considered for the density of the material: first, the nominal density ($\rho = 1111$ kg/m^3) and second, the reduced density ($\rho = 0.849 \cdot 1111 = 943$ kg/m^3), which makes the mass of the potential particle match the mass of the actual 3D-printed particle. A change in mass triggers a change in the critical time-step of each simulation group. The time-step was calculated as 10% of the critical time-step, which was computed using the formula $\mathit{\Delta} t_{crit} = \sqrt{2} \cdot \sqrt{m/k_n}$, where m the mass of the particle, as in Smilauer et al. (2021).

Group A describes a set of simulations where a small ratio of shear to normal stiffness coefficients ($k_s/k_n = 0.2$) is employed, considering the nominal density of the material. Group B describes a set of simulations with same small stiffness ratio but reduced density and thus time-step. Group C considers a stiffness ratio equal to the Poisson ratio ($k_s/k_n = \nu = 0.37$) and reduced density. Last, Group D considers the analytically-derived high stiffness ratio ($k_s/k_n = \nu = 0.773$) and reduced material density. The simulations in the parameter groups A, B and C consider the same initial packing arrangement, aiming to focus only on the differences induced by the different material properties, while the simulations of group D consider different packing arrangements, aiming to investigate the effect of initial packing on the angle of repose. For Device I, 2 simulations are conducted for each of the parameter groups A, B and C, and 18 simulations with the parameters of group D, resulting to a total of 24 DEM simulations. For Device II, 2 simulations are conducted for each of the parameter groups A, B and C, and 19 simulations with the parameters of group D, resulting to a total of 25 DEM simulations. The parameter group D was employed for the majority of simulations which investigate the effect of packing arrangement, as the method used to calculate the shear-to-normal stiffness ratio, there, is considered to be more realistic considering its analytical derivation by Mindlin (1949).

Figure 5.24 illustrates the initial state of the models created in `YADE`, using the Potential Particles code, to simulate both devices, before air pluviation initiates. The front plates of Device I and the cylindrical hopper in Device II (containing initially all particles) are made transparent, to make visualisation of the initial arrangement of particles clearer.

For Device I, the angle of repose is measured as the angle between the horizontal plane and a line connecting the center of the apex sphere and the upper end of

Table 5.2: Material properties used in DEM simulations.

Group of Parameters	A	B	C	D
k_n (N/m)	1200	1200	1200	1200
k_s/k_n	0.2	0.2	0.37	0.773
ρ (kg/m^3)	1111	943	943	943
Δt (sec)	$8.52 \cdot 10^{-5}$	$7.86 \cdot 10^{-5}$	$7.86 \cdot 10^{-5}$	$7.86 \cdot 10^{-5}$
μ_{pp}	0.713	0.713	0.713	0.713
μ_{pw}	0.514	0.514	0.514	0.514
β_n	0.071	0.071	0.071	0.071

the fixed wall (i.e. the wall directly below the moving plate). The apex sphere corresponds to the sphere of maximum height among the four spheres corresponding to the physical particle at the highest position, as shown in Equation 5.20.

$$\theta = \arctan\left(\frac{z}{L}\right) \tag{5.20}$$

where:

z = is the vertical height difference between the apex sphere and the fixed wall;
L = is the horizontal distance between the apex sphere and the fixed wall.

A similar logic is employed to calculate the angle of repose for Device II. At the axisymmetric repose state, the apex sphere is not guaranteed to be at the centre of the the heap and so 360 points are considered on the top of the cylindrical wall, resulting to 360 measurements from different directions. Two angles are defined using these measurements, the average angle of repose ${\theta_i}^{ave}$ shown in Equation 5.21 and the average of the maximum and minimum recorded values ${\theta_{mm}}^{ave}$ shown in Equation 5.22. These averages aim to quantify the mean statistical angle of repose and the middle of the range of values, as each of the 360 recorded values have different horizontal distances to the cylindrical wall.

$$\theta_i^{ave} = \frac{\sum_1^{360} \theta_i}{360} \tag{5.21}$$

where:

θ_i = is the measured angle of repose along a direction i.

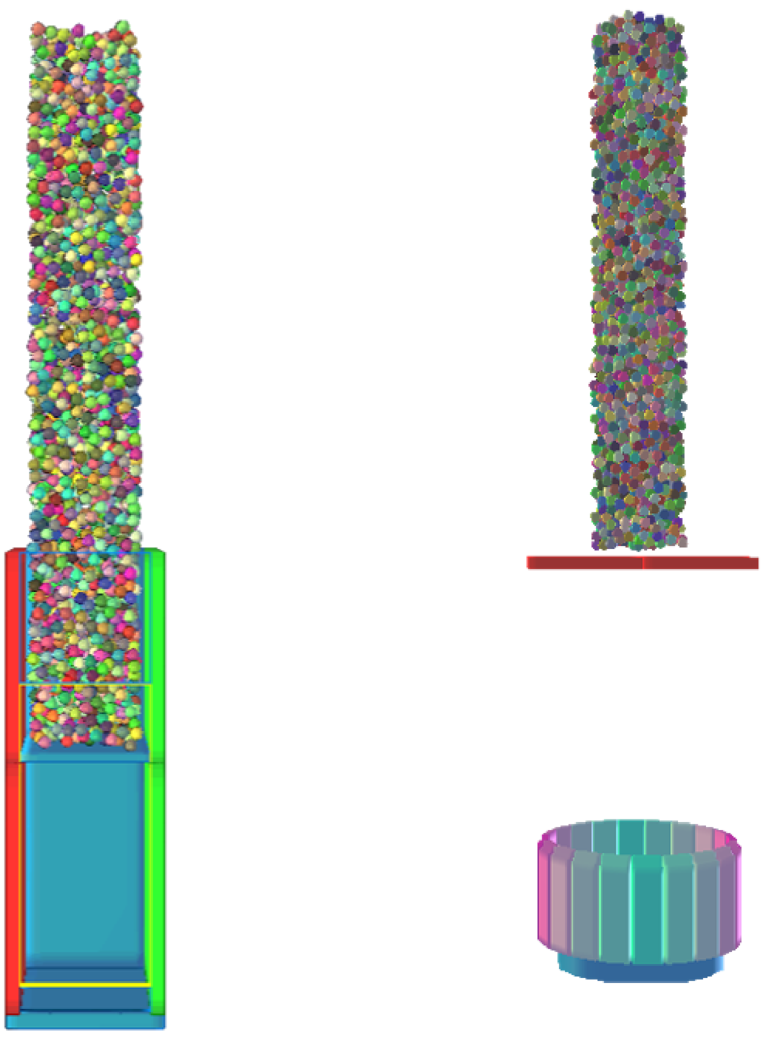


Figure 5.24: Models of devices to measure the angle of repose using the Potential Particles (a) plain-strain device; (b) axial-symmetric device.

$$\theta_{mm}^{ave} = \frac{\theta_{max} + \theta min}{2} \tag{5.22}$$

where:

θ_{max} = is the maximum θ_i value;
θ_{min} = is the minimum θ_i value.

Table 5.3 collects the average, standard deviation, maximum and minimum values of angle of repose for Device I. The results of Groups A, B and C demonstrate

Table 5.3: Values of AOR (deg) from the DEM simulations of Device I.

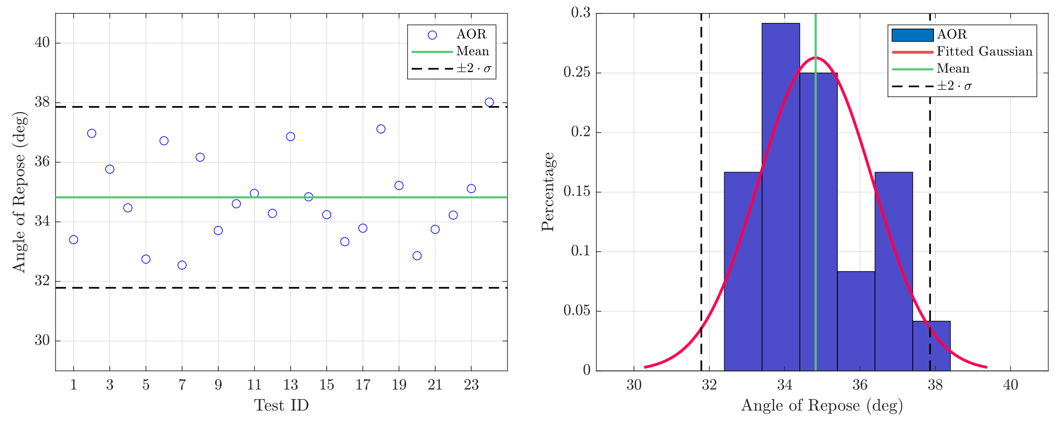
Group of Parameters	Number of Simulations	Average AOR	Standard Deviation of AOR	Minimum AOR	Maximum AOR
A	2	33.6	*	33.4	33.7
B	2	35.6	*	34.2	37.0
C	2	35.4	*	35.1	35.8
D	18	34.8	1.61	32.5	38.0
All simulations	**24**	**34.8**	**1.52**	**32.5**	**38.0**

* Standard deviation is not provided, as two values are not an adequate statistical sample.

that employing different shear-to-normal stiffness ratio values did not lead to significant variations in the measured angle of repose across groups compared to Group D or had a consistent trend with the change of shear stiffness. Instead, the large range of results among the simulations of Group D shows that the initial packing arrangement of the particles leads to more significant values of angle of repose, for the given devices and sample size. Considering the small effect of varying the shear stiffness in comparison to the effect of initial particle arrangement, which is considered to be the defining source of variation among simulations, the results of all simulations are presented together.

Figure 5.25 demonstrates the results for the angle of repose from the DEM simulations using the Potential Particles for Device I. Figure 5.25a shows the results for all four material groups (A,B,C,D), while Figure 5.25b attempts a statistical processing of these results, fitting a normal distribution curve to them. The similarity of results from all groups is further demonstrated by the fact that the entirety of results falls within two standard deviations ($\pm 2 \cdot \sigma$) of the fitted normal distribution to the histogram in Figure 5.25b. This similarity of results justifies that the Groups A, B and C do not demonstrate a high degree of variance compared to Group D and thus packing arrangement seems to play a more significant role than the stiffness ratio. Consequently, handling the results differently depending on their stiffness ratio is not necessary.

Table 5.4 and Table 5.5 show these values for ${\theta_i}^{ave}$ and ${\theta_{mm}}^{ave}$, respectively, from the DEM simulations of Device II. Figure 5.26 and Figure 5.27 demonstrate the results for the ${\theta_i}^{ave}$ and ${\theta_{mm}}^{ave}$ angles of repose, respectively, from the DEM simulations using the Potential Particles for Device II. As for Device I, the results from the simulations with different stiffness ratios in Groups A, B and C did not



(a) (b)

Figure 5.25: Measured AOR using the Potential Particles for Device I (a) results of all tests; (b) statistical distribution of results.

Table 5.4: Values of $\theta_i{}^{ave}$ AOR (deg) from the DEM simulations of Device II.

Group of Parameters	Number of Simulations	Average $\theta_i{}^{ave}$	Standard Deviation of $\theta_i{}^{ave}$	Minimum $\theta_i{}^{ave}$	Maximum $\theta_i{}^{ave}$
A	2	29.7	*	29.5	29.9
B	2	28.3	*	27.1	29.4
C	2	29.5	*	29.4	29.6
D	19	29.7	0.78	28.5	31.2
All simulations	**25**	**29.6**	**0.85**	**27.1**	**31.2**

* Standard deviation is not provided, as two values are not an adequate statistical sample.

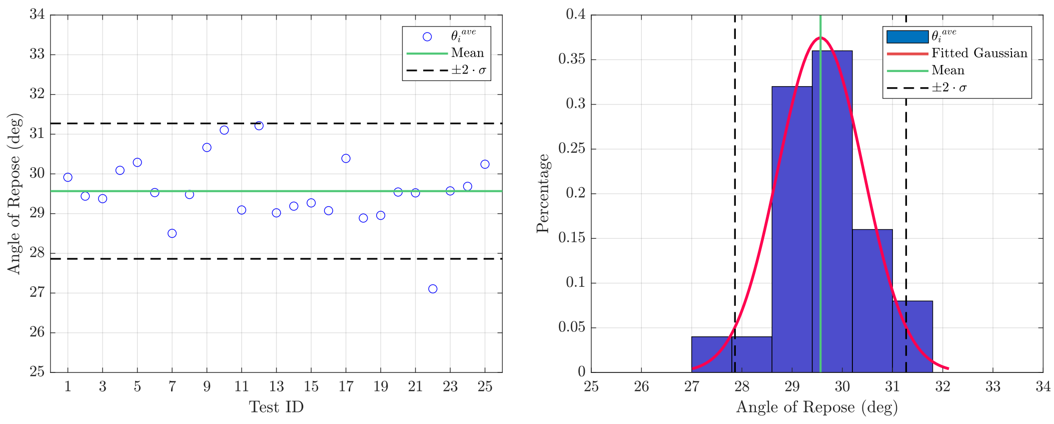
present significant variations compared to Group D; thereby all results are plotted together, considering the packing arrangement as the only significant differentiating factor. Figure 5.26a and Figure 5.27a show the results for all four material groups (A,B,C,D), while Figure 5.26b and Figure 5.27b show a statistical processing of these results, fitting to them normal distribution curves.

Table 5.6 reports the experimental values of angle of repose for Devices I and II. Comparing the average values of AOR measured in the numerical simulations with the ones measured during the physical experiments, it is found: for Device I (34.8° DEM - 41.4° experimental), while for Device II - $\theta_i{}^{ave}$ (29.6° DEM - 35.3° experimental) and Device II - $\theta_{mm}{}^{ave}$ (29.8° DEM - 35.6° experimental). These differences of 6.6° for Device I and 5.7° and 5.8° for Device II arise from the convex morphology of the potential particles used in the simulations. The concavities of the physical

Table 5.5: Values of ${\theta_{mm}}^{ave}$ AOR (deg) from the DEM simulations of Device II.

Group of Parameters	Number of Simulations	Average ${\theta_{mm}}^{ave}$	Standard Deviation of ${\theta_{mm}}^{ave}$	Minimum ${\theta_{mm}}^{ave}$	Maximum ${\theta_{mm}}^{ave}$
A	2	30.0	*	29.6	30.4
B	2	29.1	*	28.3	30.0
C	2	29.6	*	29.4	29.7
D	19	29.9	0.79	28.5	31.4
All simulations	**25**	**29.8**	**0.77**	**28.3**	**31.4**

* Standard deviation is not provided, as two values are not an adequate statistical sample.



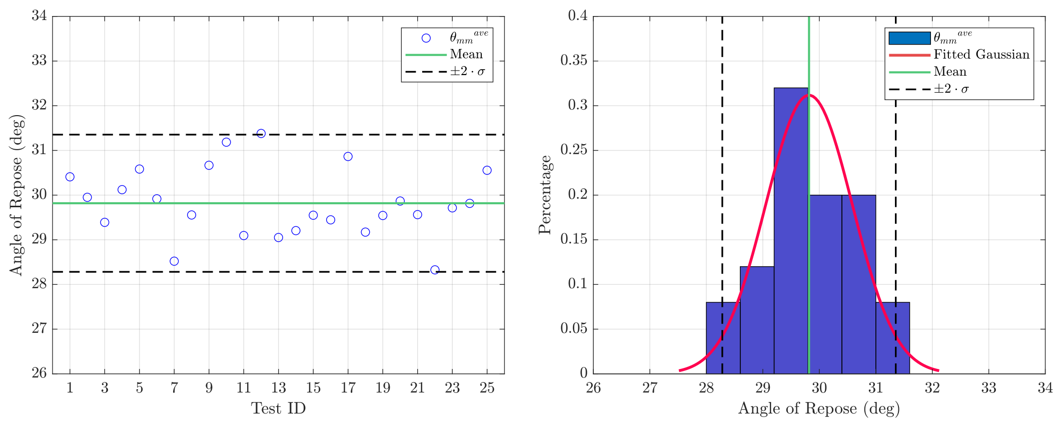
(a) (b)

Figure 5.26: Measured ${\theta_i}^{ave}$ using the Potential Particles for Device II (a) results of all tests; (b) statistical distribution of results.

particles allow them to interlock better, something not represented by the convex potential particles in the numerical simulations. This is an interesting finding, highlighting the importance of particle concavity in the formation of a reposed state. Even for these physical particles of high convexity (equal to 0.954), using a convex potential particle led to a notable underestimation of the AOR values, around 15%, for both devices.

5.10 Conclusions

This chapter discussed features of algorithmic development of two existing codes to simulate generalised non-spherical particles and sharp polyhedra, namely `Potential Particles` and `Potential Blocks`, implemented as part of this thesis.



(a) (b)

Figure 5.27: Measured $\theta_{mm}{}^{ave}$ using the Potential Particles for Device II (a) results of all tests; (b) statistical distribution of results.

Table 5.6: Measurement data of AOR (deg) obtained from experiments (source TC105, 2021).

	Num*	Average	Standard Deviation	Minimum	Maximum
Device I					
AOR	400	41.4	1.28	38.3	46.3
Device II					
$\theta_i{}^{ave}$	50	35.3	0.9	33.3	37.3
$\theta_{mm}{}^{ave}$	50	35.6	0.9	33.3	37.6

* Num stands for the number of repetitions of each experiment.

The contact laws of both codes were further developed, aiming to widen their range of applicability. In particular, the contact law of the `Potential Blocks` code was extended to also support constant normal and shear stiffness, exhibiting a linear contact behaviour with viscous damping, while the contact law of the `Potential Particles` code was extended to support non-linear stiffness, utilising a novel heuristic algorithm to calculate the contact area of the overlap region.

The contact laws of both codes were further developed to eliminate situations of non-physical attractive contact forces in the presence of viscous damping, by limiting the magnitude of the viscous contact force, so that the resultant contact force exhibits only compressive behaviour.

The contact detection algorithms of both the `Potential Particles` and the `Potential Blocks` were modified to work with periodic boundary conditions, which were already implemented in YADE. The development of this feature allows for the consideration of smaller sample sizes (i.e. numbers of particles), analysed in periodic space, which are not subjected to boundary effects, offering thus a speedup of computations. Reducing the sample size while retaining the representative nature of the sample is especially important for non-spherical particles, where the computational cost of performing contact detection is significant.

Energy calculations were developed for both codes, allowing for the monitoring of elastic potential (strain) energy, dissipation due to frictional sliding and dissipation due to viscous damping. These calculations complement existing calculations in YADE for the calculation of gravitational potential energy, kinetic energy (both translational and rotational), kinetic energy from moving the periodic boundaries and dissipation due to local (acceleration-dependent) damping. Two-particle DEM simulations of non-spherical particles were presented, using the `Potential Blocks`, to identify cases where energy was conserved, and opposite cases where the contact laws and the energy calculations of these codes need to be revisited.

The *Potential Particles* were employed to simulate two benchmark problems proposed by the Japanese domestic Technical Committe 105 (TC105) of the ISSMGE, to establish the angle of repose of a 3D-printed material, for plane-strain and axisymmetric repose states. The results indicate that under-representing the concavity of the physical material using convex potential particles led to an under-estimation of the angle of repose. Quantified morphological differences between the physical material and the generated potential particle were provided.

All the features developed in this chapter have been shared with the global community of YADE users and developers in an open-source manner.

Chapter 6

Mechanical characterisation of railway ballast

6.1 Introduction

The study of particulate materials at the mesoscopic scale is conducted using a Representative Element Volume (REV)[1]. This is a subset of the bulk material, which is considered to be representative of its main characteristics. The nature of these characteristics, as well as the size of the REV depend on the application at hand.

This chapter lays down a methodology to generate REVs of non-spherical particles, which reflect the statistical distribution of both particle size and shape. This methodology applies to any modelling approach for granular materials, be it multi-sphere particles, polyhedra, potential particles and µFE, inter alia. This methodology does not mean to replicate an existing sample, as e.g. in Kawamoto et al. (2018), but it rather attempts to reconstruct a representative sample, given the available morphological information of the particles at their original fidelity level, before any simplification or approximation is applied.

The methodology is applied in this chapter to generate representative samples of railway ballast using multi-sphere particles. Shape characterisation is performed for particles of the material at their original fidelity level. Then, simplified particles are generated using the two aforementioned approaches and their morphology is characterised and compared to that of the original particles. Then, simplified particles are chosen aiming to approximate the size and shape polydispersity of the original

[1]Often referred to as Representative Elementary Volume or Representative Volume Element (RVE).

particles, as closely as possible. Discrete element tests of triaxial compression are performed for the simplified particles and compared to available experimental data.

6.2 Background

Several studies in the literature have employed the Discrete Element Method to characterise the mechanical behaviour of granular materials, in terms of critical state shear strength (Harkness and Zervos, 2019; Angelidakis et al., 2021a), packing (Nadimi and Fonseca, 2016; Soltanbeigi et al., 2021) and fabric (Orosz et al., 2021). When spherical particles are employed, size polydispersity plays a crucial role in the stiffness and compressibility (Minh and Cheng, 2013), the critical state shear strength (Jiang et al., 2018) and stress distribution (Liu et al., 2021) of granular assemblies. For non-spherical particles, particle morphology and its polydispersity play a significant role, additionally to particle size effects.

In recent years, studies using non-spherical particles in discrete element simulations have become more common. Many of these studies employ idealised particle shapes, such as ellipsoids (Zheng et al., 2013), superellipsoids and superquadrics (Zhao et al., 2018; Podlozhnyuk et al., 2017) or poly-superellipsoids (Zhao and Zhao, 2019). Regarding realistic particle shapes, multi-sphere particles (Garcia et al., 2009), polyhedra (Boon et al., 2015b) and potential particles (Ahmed et al., 2016) have been used to simulate irregular particles. In many occasions, the shape of these irregular particles is selected randomly; e.g. Eliáš (2014) used Voronoi tessellation to generate randomly convex polyhedra, which he scaled to generate particles of varying aspect ratios. Ahmed et al. (2016) used a small number of ballast-particle images to generate potential particles approximating the realistic ones in a qualitative manner.

When polyhedra are used, it is a common requirement for the particles to be convex, to achieve efficient contact detection. Multiple convex polyhedra connected rigidly in a clump have been used to generate concave particles (Govender et al., 2016; Boon et al., 2015b), a process which increases the number of particles in the simulation and thus computational cost. Recent advances on the field, aided by new methods of Computational Geometry to perform Approximate Convex Decomposition (Lien and Amato, 2008) allow for a systematic decomposition of concave particles into a small set of non-overlapping, convex particles.

Kawamoto et al. (2018) used the Level-Set DEM (LS-DEM) to simulate the mechanical behaviour of concave particles, without needing to decompose them into subsets of convex ones, demonstrating that *"All you need is shape"*. Using a voxelated representation of the particle morphology, the LS-DEM was used to replicate the exact morphology of particles and their arrangements in a granular assembly imaged using X-ray Computed Tomography. This allowed for a faithful replica of the real particle geometries, which were referred to as *avatars* or *digital twins*, rather than *models*, by Kawamoto et al. (2018) and Jostad et al. (2021), respectively, aiming to emphasize their increased level of fidelity. In the LS-DEM, particle shape is represented by a collection of voxels, a data format similar to that of reconstructed images derived using Computed Tomography. This practical similarity regarding the voxelated data representation of each particle minimises the extent of pre and post-processing efforts when image-informed modelling is of interest, as it bridges the distance between digital imaging and numerical modelling.

Using a voxelated representation of particles for numerical modelling purposes is not a recent advancement in the field. The DigiPac algorithm, with both 2D and 3D flavours (Jia and Williams, 2001; Jia et al., 2007), pioneered the modelling of non-spherical particles, in situations of complex particle morphologies, where spheres or convex polyhedra could not offer a realistic modelling of packing conditions or voidage size and evolution. Digipac facilitated the modelling of particles with concavities and intragranular voids, with little computational cost. This was the case, as initially the software offered geometrical, digitally-based packing prediction, where the particles could move randomly, without a consideration of the contact forces, a fact that contributed positively to the computational efficiency of the method, at the cost of estimation accuracy.

To improve this, Caulkin et al. (2009) presented two new versions of DigiPac, where the full contact characteristics were taken into consideration, i.e. the interaction forces between particles. Although they reported increased computational cost associated with these new versions, they observed more realistic simulations of the packing structures of non-spherical particles under various dynamic loading conditions, including rotations, vibrations and tapping. These two versions introduced DigiDEM and DigiCGP; DigiDEM is a fully physics-based approach, comprising an implementation of the Discrete Element method, while DigiCGP is a collision-guided digital packing model, designed as an intermediate solution between DigiPac and DigiDEM, and is partially guided by collisions, leading to reduced computational cost but also accuracy. Caulkin et al. (2015) demonstrated the merits and limitations of voxel-based modelling approaches, against the multi-sphere and polyhedral approaches facilitating a discussion from both morphological and computational standpoints. It should be noted that although both the LS-DEM and DigiDEM use voxelated particle representations, their similarities do not extend past that, as each method employs different logics to facilitate contact detection and to estimate interaction characteristics, such as contact point, contact normal and contact law.

The Finite Element Method (FEM) and the Finite-Discrete Element Method (FDEM) have been used to simulate non-spherical particles. Both methods merit from their inherent ability to simulate interactions of concave particles (Nadimi et al., 2020; Farsi et al., 2021), while Nadimi and Fonseca (2017b); Nadimi et al. (2017) demonstrated that the "General contact algorithm" of Abaqus (Abaqus, 2014) has the capacity to replicate the behaviour of analytical contact laws for interacting spheres and different loading conditions, i.e. Hertz (1882) for normal contact interactions, Mindlin and Deresiewicz (1953) for shear contact interaction, Lubkin (1951) for torsional loading conditions and Johnson (1987) for rotation. However, these methods result in an increase of computational cost, compared to the classic DEM, as they are hindered by the need for a fine discretisation of the interior of the particle (as shown in Nadimi and Fonseca, 2018), in order to achieve a proper integration of stresses and calculation of contact characteristics (i.e. contact force magnitude and normal direction). The FEM and FDEM allow for the direct and accurate modelling of particle deformability, something that is only approximated in the classic DEM (Cundall and Strack, 1979) considering soft-particle approach.

Having discussed the various available approaches to simulate non-spherical particles, the question arises of what shapes these particles should have. A recent study by Wang et al. (2021) proposed a methodology to generate random packings of granular materials with complex particle shapes, with controlled form, roundness and convexity features. Such algorithms are useful to facilitate the generation of packings with particles of polydisperse sizes and shapes. A delicate point in such a method is to decide which formulae are useful to quantify the different aspects of particle shape. As demonstrated in Chapter 2 and Chapter 3, multiple formulae exist to represent particle form and roundness and thus the results of this packing generation algorithm will depend on the chosen sets of indices.

When realistic particles are of interest, it is important to ensure that the particles used in the numerical simulations maintain some association in terms of size and shape to the particles of the original fidelity level. To this end, it is essential to define a methodology based on quantitative shape characterisation, which will ensure the particles considered in the numerical simulations follow size and shape distributions with some association to the ones of the physical particles.

6.3 A methodology to generate representative element volumes in the DEM using realistic particle shapes

This section formulates the steps of a proposed methodology, to generate REVs of non-spherical particles, which follow a target size and shape distribution and can mimic its shear strength characteristics (in terms of mobilised friction angle and dilatancy). Although the particle size distribution (PSD) is a well defined term, the notion of a "shape distribution" has been used loosely in the literature (e.g. see different definitions in Itoh and Wanibe, 1991; Rorato et al., 2018), to represent the overall morphological "profile" of the particles of a granular material. This study focuses on particle form, which is quantified in terms of elongation and flatness, using the new formulae proposed in Chapter 2. Particle form is here isolated for the sake of simplicity, and because form affects significantly the packing characteristics of irregular particles at the REV scale. This does not mean to take away from the importance of roundness and roughness, as angular particles have been demonstrated to lead to further interlocking (Nadimi et al., 2020) and stress concentrations, while rough particle surfaces affect the normal and shear stiffness when they form contacts with their surrounding particles (Otsubo et al., 2017; Nadimi et al., 2020; Sandeep et al., 2019).

Reflecting on the structure of this thesis, the methodology proposed in this chapter brings together the formulae for flatness and elongation proposed in Chapter 2 to characterise particle form, the `SHAPE` and `CLUMP` codes demonstrated in Chapter 3 and Chapter 4, respectively, for particle shape characterisation, simplification and for generation of numerical models, as demonstrated in Figure 6.1.

The main steps of the proposed methodology to generate REVs which follow a size and shape distribution are detailed below:

- The form of the scanned particles is characterised in terms of flatness and elongation, for the original fidelity level.

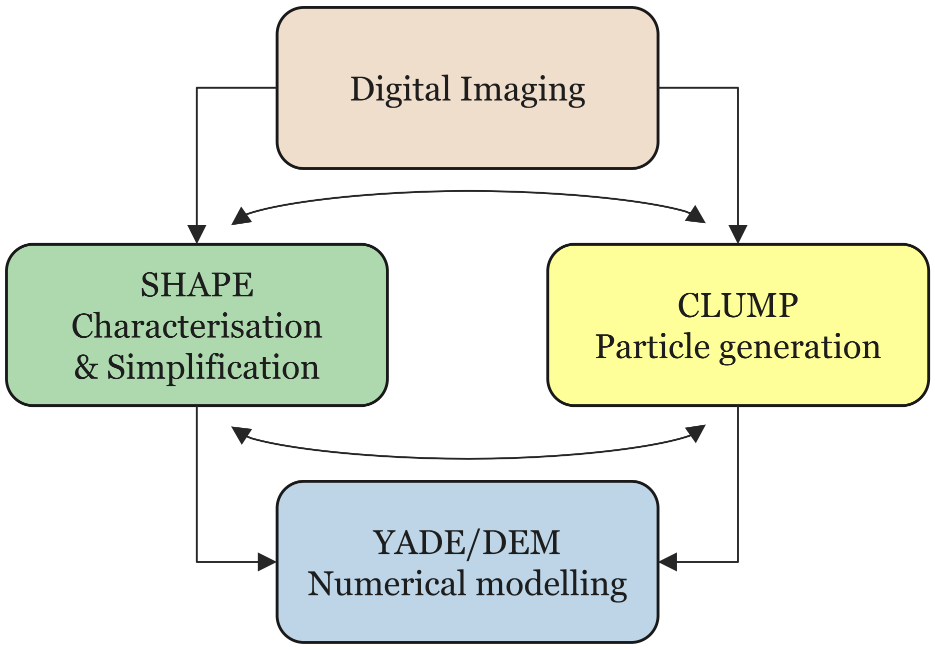


Figure 6.1: Interaction of components to characterise the particle shapes of real particulate materials based on imaging data, create simplified particle representations, characterise their morphology and generate them inside numerical codes.

- Simplified particles are generated using multi-sphere particles.
- The morphology of the simplified particles is characterised and compared to that of the original particles.
- Simplified particles are selected based on their flatness and elongation values, aiming to approximate the distribution of flatness and elongation of the particles of the original fidelity level at the REV level. This is performed separately for each sieve cut.
- Numerical simulations are performed for different fidelity levels.

It should be noted that the simplified particles will typically have a different shape distribution, in terms of flatness and elongation, compared to the original fidelity level. This inescapable loss of fidelity is an artifact of the simplification process. Following this proposed methodology, the simplified particles are used in such proportions so that they approximate the shape distribution of the original fidelity level. As a result, the particles used in the DEM simulation will have altered morphological characteristics if compared to their original counterpart, but the overall shape distribution at the sample scale will be close to that of the original fidelity level.

6.4 Experimental characterisation

The methodology developed in the previous section is applied to study the shear strength of railway ballast via discrete element simulations. This section discusses results of experimental characterisation of the material, reported in Xiao et al. (2017). The material of interest is granite ballast, with regular densities ($\rho = 2650 kg/m^3$) and particle sizes. Figure 6.2 shows the particle size distribution of the ballast material, with sieve sizes varying from 16 mm to 63 mm, as detailed in Xiao et al. (2017).

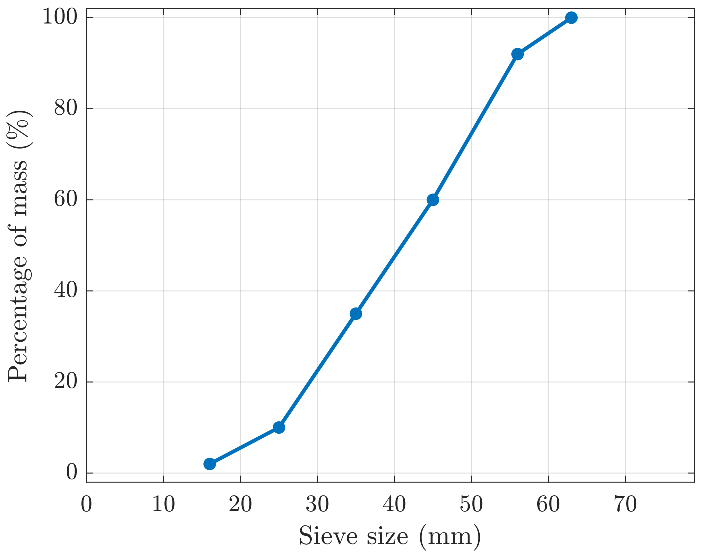


Figure 6.2: Particle size distribution (PSD) of the ballast material.

Xiao et al. (2017) scanned the morphology of several hundred ballast grains using a `Creaform Go!SCAN3D` hand-held imaging laser scanner and they provided a subset of 100 of their scans in aid of this thesis. The point cloud corresponding to the surface of each particle was reconstructed in this thesis using the "Ball-Pivoting" method in `Meshlab` (Cignoni et al., 2008). The large size of the ballast particles along with the high resolution of the laser scanner resulted in particle surfaces of approximately 60,000 to 600,000 faces per particle, which provided detailed morphological information of each particle. Figure 6.3 shows the scanning of a single particle as presented in Xiao et al. (2017), while Figure 6.4 shows all the 100 analysed ballast particles corresponding to each sieve cut.

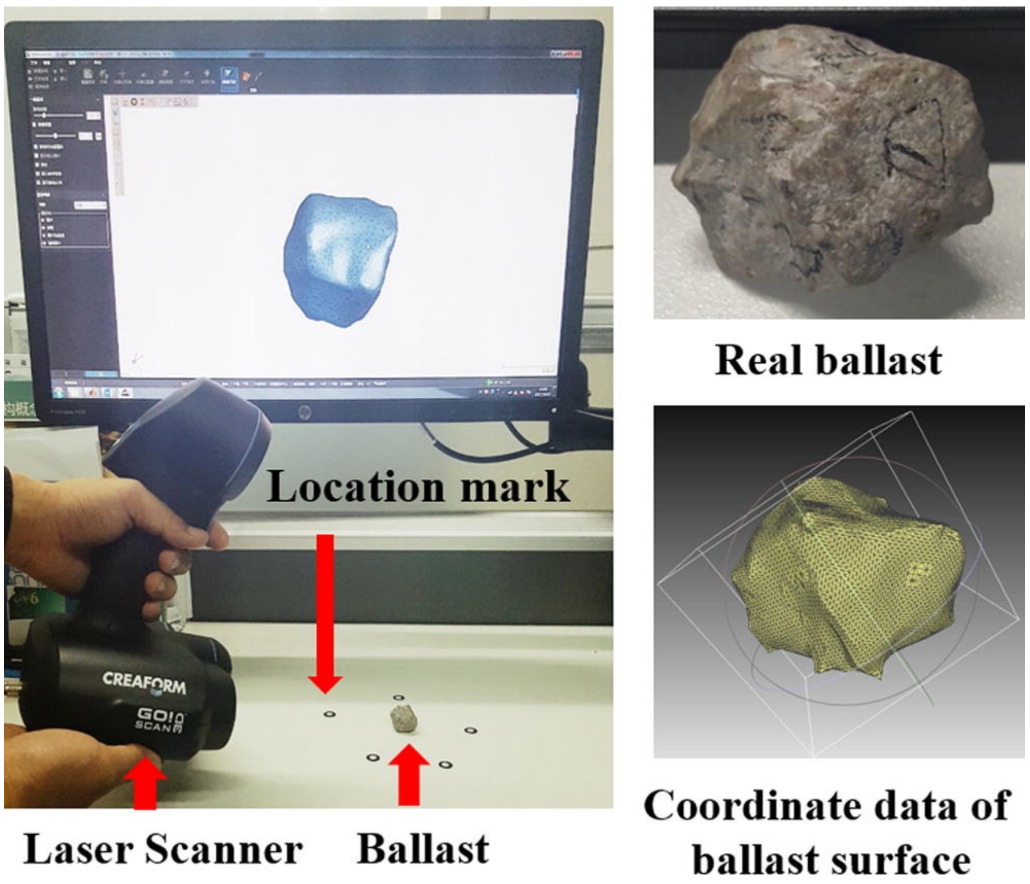


Figure 6.3: Laser scanning of a single ballast grain (reprinted from Xiao et al. (2017) with permission from Elsevier).

Xiao et al. (2017) tested cylindrical ballast samples with a diameter of 300mm and a height of 600mm using a large-scale triaxial apparatus. Initially, the samples were compacted by artificial vibration, compressed isotropically, and then sheared under static and cyclic deviatoric loading. The static loading was imposed via strain control with uniform loading velocity of 3 mm/min (i.e. 0.5% of the sample height per minute) and for confining pressures of 10 kPa, 30 kPa and 60 kPa. Figure 6.5 shows the experimental results reported by Xiao et al. (2017) for the static shear tests of the material, along with results from two-dimensional discrete element simulations using clumps. It becomes evident that the material exhibits dilative behaviour for all levels of confining stress. The static triaxial shear tests were conducted for axial strains up to 10%, which is an average level of imposed deformation, not enough for the material to reach critical state conditions. More information on the numerical

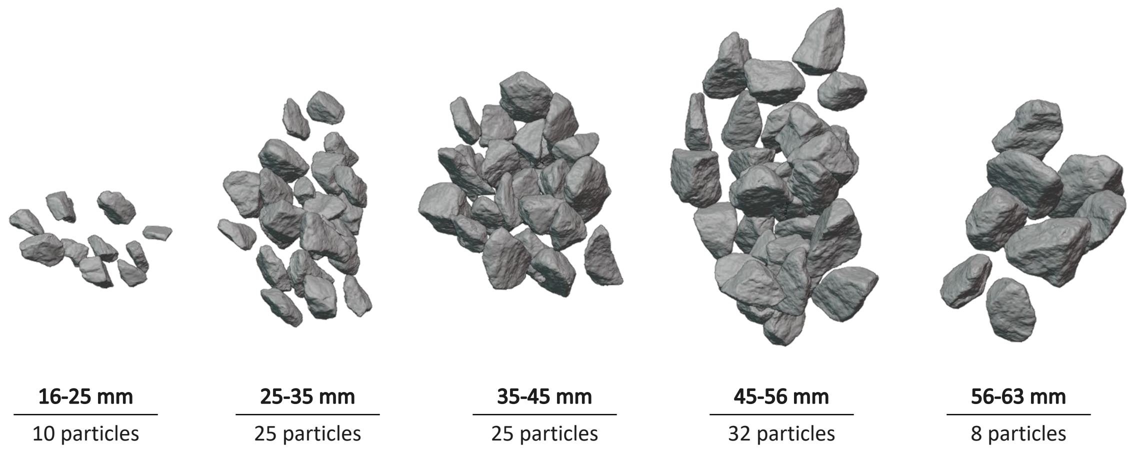


Figure 6.4: Scans of the 100 analysed ballast particles for all sieve cuts.

simulations of this figure can be found in Xiao et al. (2017).

6.5 Morphological characterisation

Quantitative characterisation of particle morphology is at the center of the proposed methodology. The SHAPE and CLUMP codes played a catalytic role in performing shape characterisation and generating simplified multi-sphere particles, in an automated manner. This section reports results of shape characterisation for 100 analysed ballast particles.

6.5.1 Original fidelity level

First, the morphology of the scanned particles is characterised in SHAPE for the original fidelity level, i.e. before any simplification takes place. The bubble charts of Figure 6.6 show values of the degree of true sphericity, intercept sphericity and convexity, plotted on top of a Zingg plot. This style of visualisation allows the direct comparison of different shape indices, e.g. by demonstrating that compact ballast particles take higher values of sphericity, using both the interpretations of Wadell (1932) and Krumbein (1941), an observation that was made for cuboids and ellipsoids in Chapter 2. On the other hand, convexity does not show any correlation with the particle aspect ratios, as it could be expected, since concavities can appear on the surface any particle, be it elongated, flat or compact.

The bubble charts of Figure 6.7 illustrate values of elongation, flatness and com-

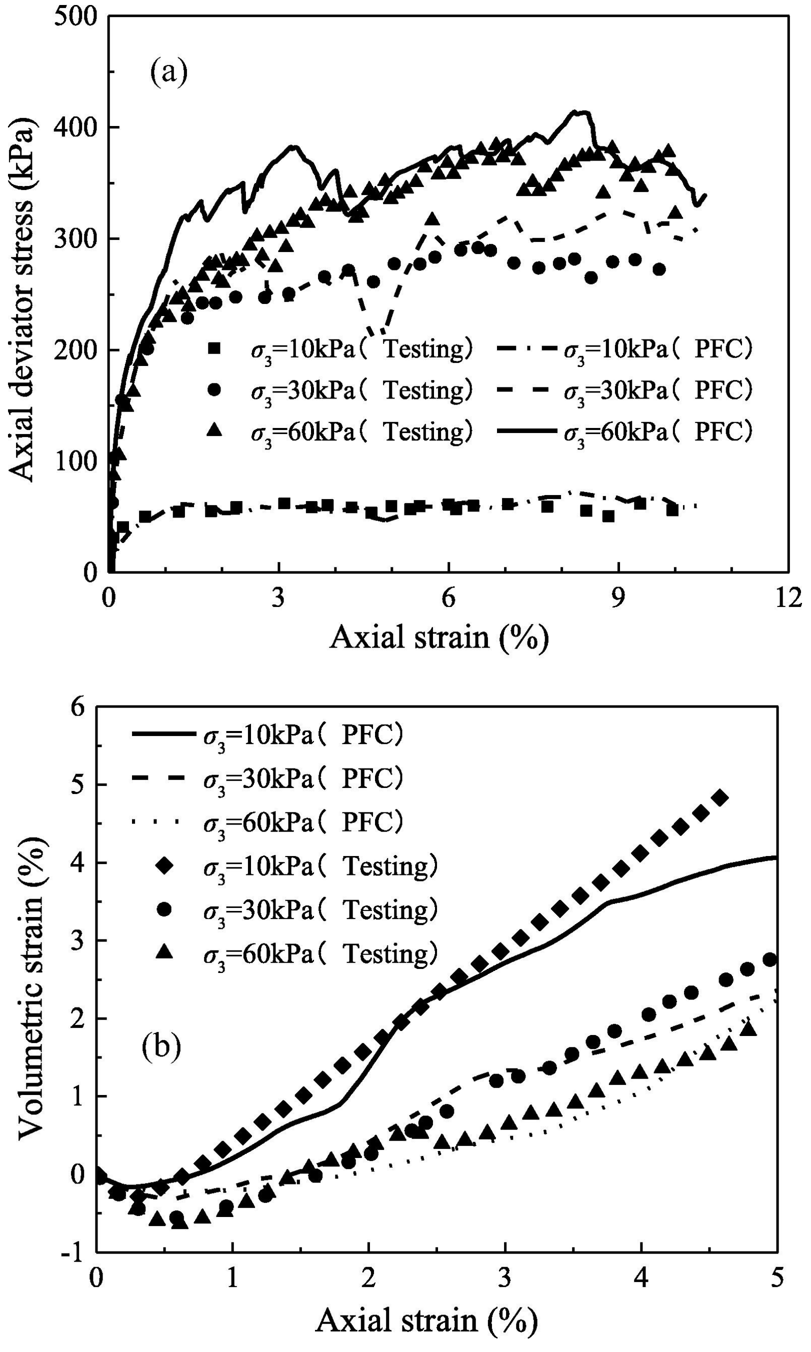


Figure 6.5: Experimental and numerical results under static loading, a) axial deviatoric stress versus axial strain and b) volumetric strain versus axial strain. (reprinted from Xiao et al. (2017) with permission from Elsevier).

pactness, expressed with the formulae proposed in Chapter 2. The main particle dimensions used in the calculation of these indices were considered as the sizes of the Oriented Bounding Box of minimum volume of each particle. Looking into the

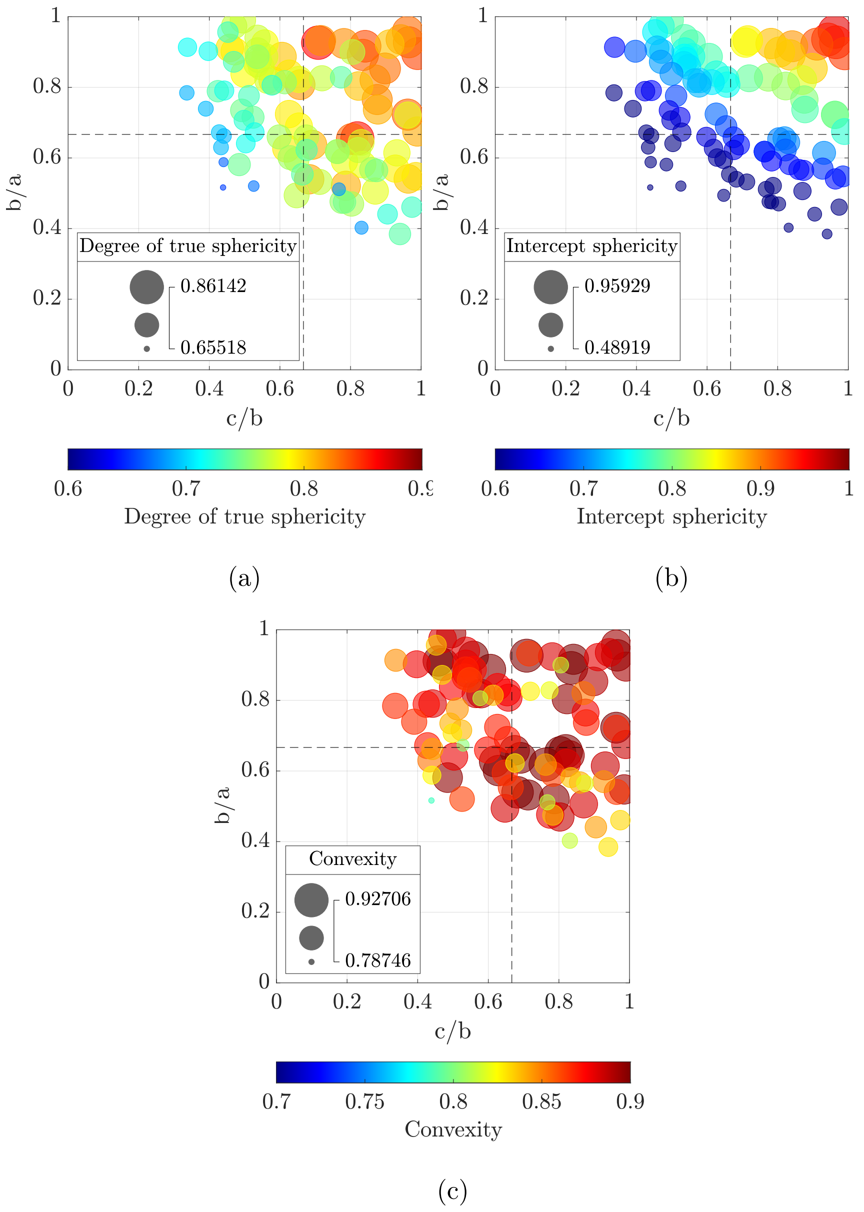


Figure 6.6: Shape characteristics of 100 ballast particles for the original fidelity level (a) Degree of true sphericity (Wadell, 1932); (b) Intercept sphericity (Krumbein, 1941);(c) Convexity.

values of each shape index (illustrated by the size and color of the bubbles), it becomes evident that the particles in the region `IV` of the Zingg plot for elongated particles take the highest elongation values using the proposed indices, the particles in the region `I` of the Zingg plot for flat particles take the highest values of flatness, while the particles in the region `II` of the Zingg plot for compact particles take the highest compactness values. This good agreement between the proposed indices and the Zingg system was expected, considering that all particles fall on the upper-right region of the plot and this agreement was demonstrated for the regions I, II and IV in Chapter 2. Figure 6.7d shows a classification of particle form for the 100 ballast particles at their original fidelity level, based on their flatness and elongation values, using the proposed classification system introduced in Chapter 2. Overall, 37% of all particles are flat, 23% are compact, 9% are bladed and 31% are elongated.

Figure 6.8 shows the morphology of some of the ballast particles in more detail, for each of the four morphological classes for flat, compact, bladed and elongated particles. Figure 6.8a shows the particles with maximum fl, el and co values, corresponding to extreme particle morphologies of flat, elongated and compact ballast particles, respectively, along with the bladed particle closest to the origin of the Zingg plot. Figure 6.8b shows four more ballast particles, each corresponding to a representative particle of each class. These are particles of average size, in respect to the particle size distribution shown in Figure 6.2, i.e. their size approximates the D50 particle diameter of the material.

6.5.2 Multi-sphere particles

Having demonstrated the shape characteristics for the original fidelity level, shape characterisation follows for the multi-sphere particles which will be used in DEM simulations. The quantitative characterisation of multi-sphere particles is not trivial. The surface extraction routine of the `CLUMP` code, developed as part of this thesis, is used to tessellate the surface of each multi-sphere particle, in the form of a triangulated mesh, which is then processed using the `SHAPE` code. This was carried out in an automated manner, utilising the good cooperation between these two open-source tools, minimising the processing time to mere minutes and also minimising human error. Several fidelity levels were considered, for 10, 20, 30, 40 and 50 spheres per particle.

The analysis in this section focuses on the elongation, flatness and compactness indices, which are used to quantify the shape distribution, following the methodology to generate REVs with shape distribution faithful to that of the original material. As discussed in Chapter 2, these indices add up to unity (i.e. $el + fl + co = 1$) and so their illustration in a ternary plot is considered intuitive, as it allows for a monitoring of all three of them on the same chart.

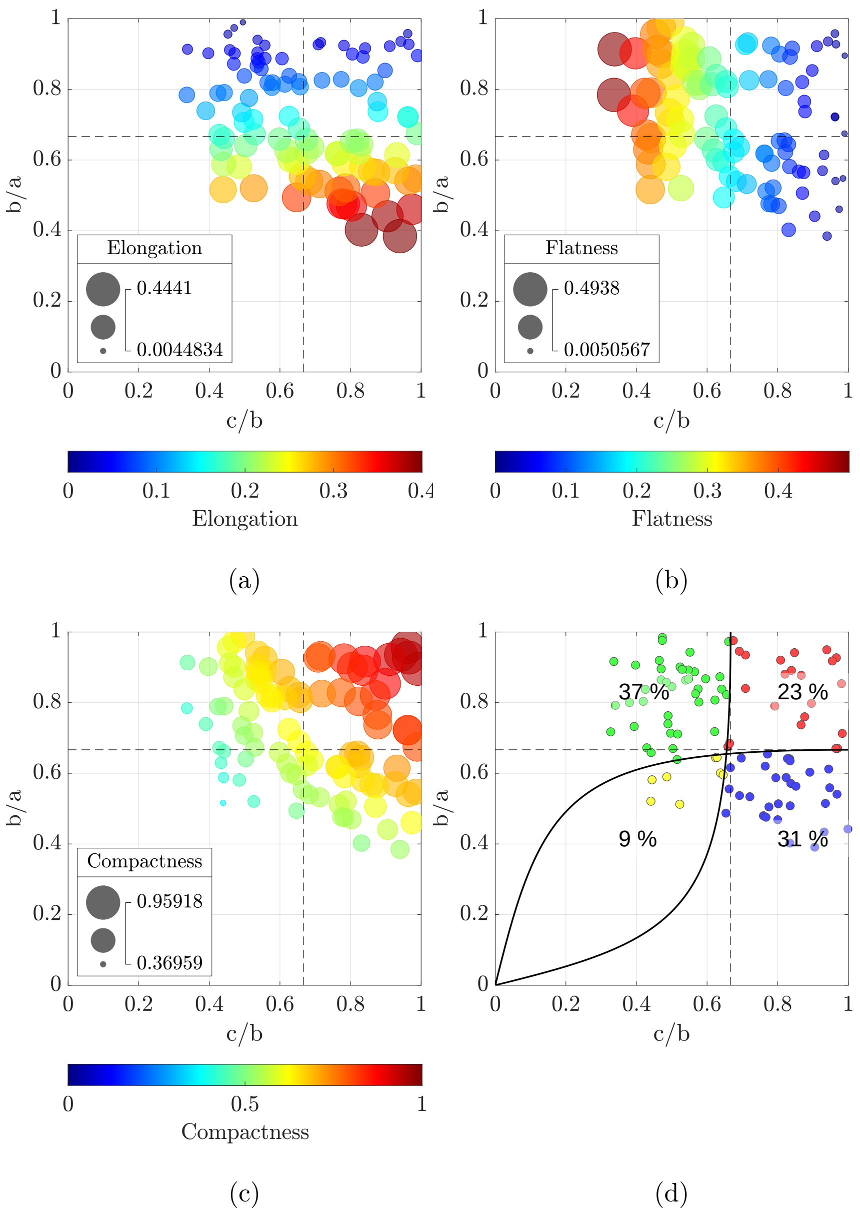
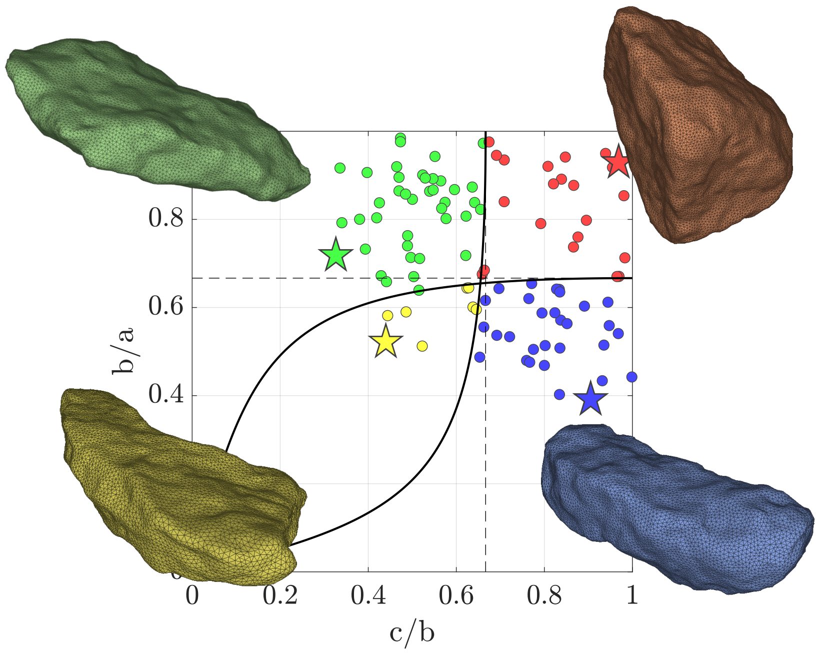
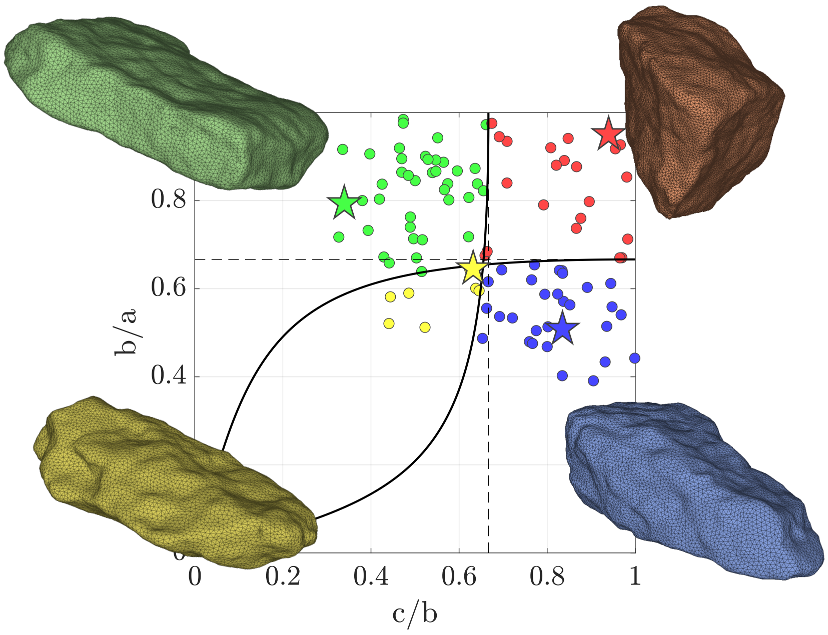


Figure 6.7: Shape characteristics of 100 ballast particles for the original fidelity levels (a) Elongation; (b) Flatness; (c) Compactness; (d) Classification of particle form, using the indices proposed in Chapter 2.

Figure 6.9 illustrates the shape distribution in terms of elongation, flatness and compactness, for the original 100 particles and for clumps of decreasing fidelity, with 50, 40, 30, 20 and 10 spheres per particle. These ternary plots are drawn on top of a density map, a ternary histogram in other words, which counts the percentage of particles belonging to each bin of the chart. This provides an accurate



(a)



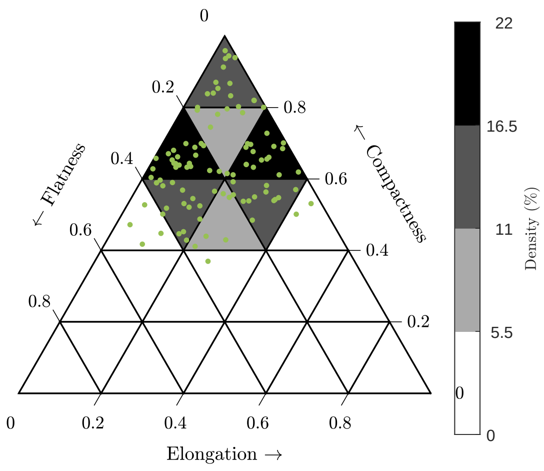
(b)

Figure 6.8: Railway ballast particles for each morphological class; (a) particles with extreme shape parameter values; (b) representative particles of average size. Each star marks the morphology of the visualised particle with same colour, for each class.

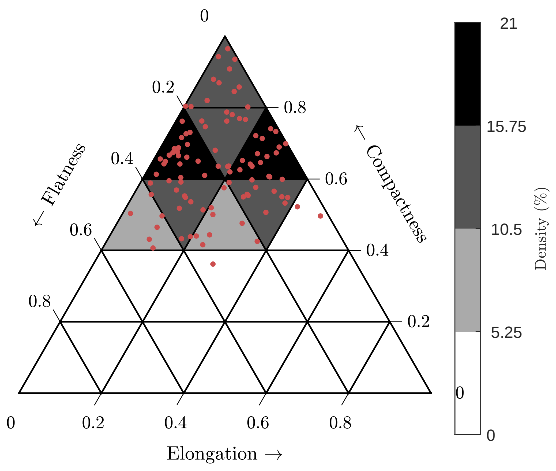
quantification of the alteration of particle morphology, for decreasing particle fidelity. Following this method of binning the particles for each fidelity level, the changes among bins are monitored, as the particles can migrate to neighbouring bins during the simplification of each particle (i.e. model it with less spheres).

Having that information, the problem of creating an REV for each fidelity level, which follows the shape distribution of the original fidelity level, is simplified to a problem where the simplified particles are selected to participate in the REV with a probability equal to the percentage of appearance of particles of the same bin, but for the original fidelity level. If a bin remains empty after particle simplification, but was not empty for the original fidelity level, particles from the closest bin are selected to fill the necessary percentage of similarly-shaped particles, as neighbouring bins have similar form features. The number of bins depends on the number of available particles, the dispersion of form parameters within this sample of particles and the degree of accuracy, i.e. how closely the particle shape distribution of a simplified fidelity level is sought be kept, in comparison to the original one. In this case study, where 100 particles were analysed, 5 bins provided an adequate discretisation of the elongation - flatness - compactness domain.

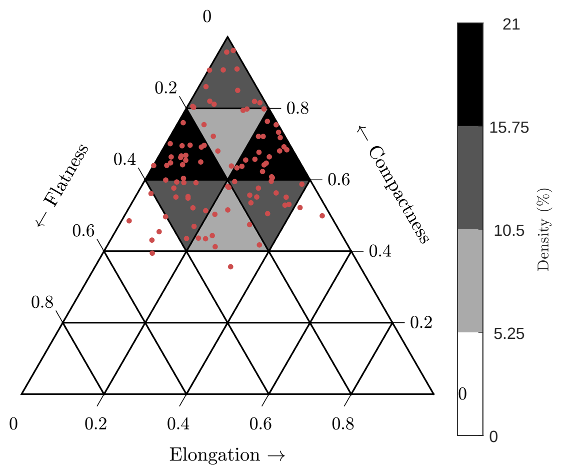
It should be emphasised that the simplified clump particles are not used in proportions to their altered shape distribution (marked with red dots in Figures 6.9b to 6.9f) during sample preparation. Instead, the simplified particles are first categorised using their altered shape distribution, and are then used to approximate as closely as possible the shape distribution of the original fidelity level (marked with green dots in Figure 6.9a).



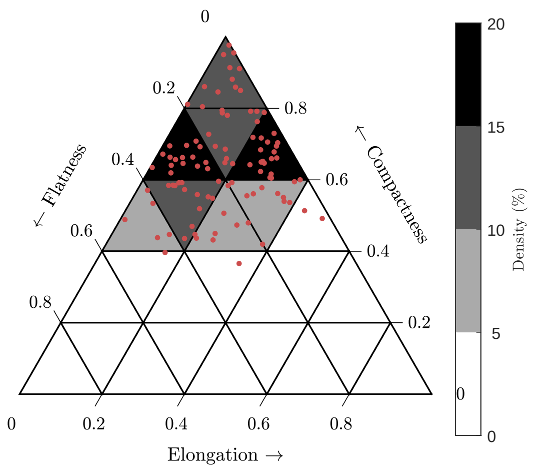
(a)



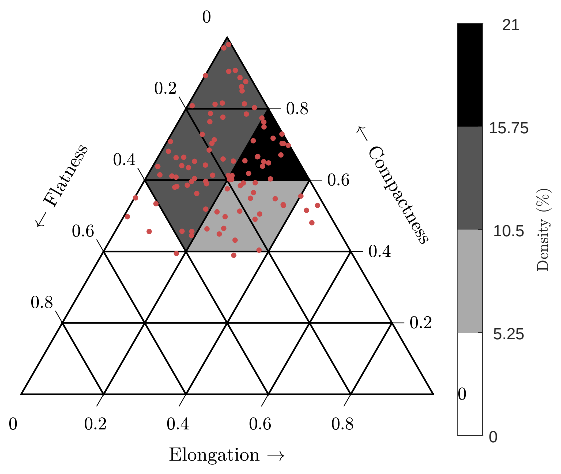
(b)



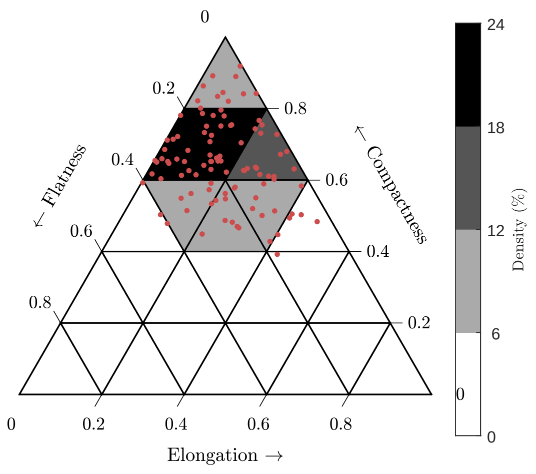
(c)



(d)



(e)



(f)

Figure 6.9: Ternary plots of elongation, flatness and compactness of the 100 analysed ballast particles for (a) the original fidelity level (green) and clumps (red) with (b) 50 spheres (c) 40 spheres (d) 30 spheres (e) 20 spheres (f) 10 spheres.

6.6 Mechanical characterisation

This section discusses the numerical simulations of triaxial tests on railway ballast using multi-sphere particles. The Euclidean transform method, proposed in Chapter 4 was used to generate multi-sphere non-spherical particles, having as target particle geometry the ballast particles scanned by Xiao et al. (2017).

A parametric study was conducted, varying different modelling parameters, which aimed to investigate the effect of each parameter in the shear strength of the material at critical state conditions. For each variation, only one aspect of the simulation is altered. Table 6.1 shows the modelling parameters for each group of simulations. The parameters of interest entail: the size of the REV, the initial packing arrangement of the particles, the number of sphere-members of each clump, the inter-particle friction and the particle shape distribution.

In the first group of simulations (Group S in Table 6.1), the size of the REV was investigated, in search for a number of particles per simulation that is adequately small, to minimise computational cost, but adequately large, to keep the samples representative. In the second group (Group P in Table 6.1), REVs of the same size but different initial packings were examined, to ensure that these simulations demonstrate representative and reproducible mechanical behaviour. Having established an adequate size of the REV, in the third group of simulations (Group N in Table 6.1), the number of sphere-members of each clump is varied to different fidelity levels, to investigate its effect on the shear strength and dilatancy of the material of interest. At this stage, the size and initial packing arrangement of the REV have been established, and the effect of particle fidelity on the observed behaviour is investigated. To explore the effect of mechanical parameters on the shear strength, in the fourth group (Group F in Table 6.1), the inter-particle friction angle of the particles is varied to explore the dispersion of results for cases of low friction coefficients to cases of high friction coefficients. Extensive literature exists on the effects of inter-particle friction on the meso-scale shear strength of granular materials (e.g. Angus et al., 2020; Huang, 2014) and it was deemed interesting to study these effects of the analysed ballast material and using simulations of image-informed non-spherical particles. Last, in the fifth group (Group D in Table 6.1), different size and shape distributions were employed to study their effect on the mechanical response of the material during shearing. The shear strength of the real material, i.e. considering the real size and shape distribution, was compared to that of four monodisperse samples with particles belonging to different shape classes (i.e. flat, compact, bladed and elongated), along with simplified shape distributions constructed using these four different particles in different proportions.

In the first four groups of simulations, 100 different particle shapes were populated into samples of 5000-1000 particles, representing the real size and shape distributions of the material, as recorded for the original fidelity level. In the last group, monodisperse and simplified polydisperse packings composed of only four particles with different form were analysed, to quantify the effect of size and shape polydispersity in the shear behaviour of the material.

6.6.1 Sample preparation

The same sample preparation procedure was followed to generate the initial packings of all analysed samples. First, each REV was generated at a loose state, where no particles were in contact, using the `makeClumpCloud(minCorner, maxCorner, clumps, num, seed, periodic)` routine of `YADE`. The `minCorner, maxCorner` parameters define the size of a cuboidal domain where the initial sample will be generated. The `clumps` parameter is a list of all the different clump particles that will be used with equal probability to generate the sample. To achieve a target particle size and shape distribution, each clump must be replicated in this list several times, proportionally to its probability of appearance. The `num` parameter defines the total number of clump particles that will be generated. A `seed` parameter was used to generate packings with the same initial position and orientation of particles, utilising a random number generation routine in `YADE`. Last, the parameter *periodic* defines whether the packing should be generated in periodic space, which was the preferred modelling choice in the triaxial tests discussed in this chapter. Using `makeClumpCloud`, each clump is generated in its bounding sphere, and positioned within the domain defined by `[minCorner, maxCorner]` if it does not intersect with the bounding sphere of another existing clump. This generation procedure is repeated until the total number of clumps (`num`) is generated or until a maximum number of 200 attempts to generate a single clump is reached.

The mass and inertia tensor of each clump particle was calculated using the routine `updateClumpProperties(discretization=25)` of `YADE`, which calculates the inertial characteristics of clumps with overlapping sphere-members assuming homogeneous density, overcoming the problems of clumps with non-homogeneous density reported by Ferellec and McDowell (2010). This routine creates a three-dimensional voxelated grid enclosing the clump, aligned with the global axes of the model, with resolution controlled by the `discretization` parameter; if a voxel falls inside one of the clump sphere-members, it contributes to the calculation of mass and inertia, thus solving the issue of overestimating the inertial characteristics for clumps with overlapping members. A grid of size 25 was found to provide an adequate estimation of mass and inertia for the studied particles, through a trial-and-error search, as finer grids led to only small improvements in these calculations, while they required more available memory.

The sample was isotropically compressed under a low inter-particle friction angle of 5° and high local damping (70%), a typical process for the efficient generation of dense packings. The Young's modulus of the material was considered $E = 1GPa$ and its Poisson ratio $\nu = 0.23$, which are typical values for granite ballast, as found in the literature for DEM simulations. In particular, the actual Young's modulus of granite is higher ($\approx 50 - 70GPa$), and down-scaling it is a common practice to enhance computational efficiency. Also, density scaling was applied, multiplying the density of the material by 100 (i.e. $\rho = 2650 \cdot 100kg/m^3$ was considered), again aiming to enhance computational efficiency. Employing these scaling approaches is typical for the modelling of quasi-static processes, such as the triaxial compression tests performed in this chapter, which are not affected by inertia effects, as the applied strain-rate is low, as in Thornton (2000). For dynamic processes which involve body (i.e. inertial) forces acting on the particles, such as gravity, scaling the mass and stiffness of the system should be carried out in a way that does not alter the eigenvalue and frictional characteristics of the system, e.g. via application of advanced stiffness scaling laws, like in He et al. (2021), where the effect of scaling stiffness on the tangential behaviour of granular materials was demonstrated.

The packings were compressed to achieve an isotropic stress state at $\sigma_3 = 60kPa$, and at a porosity of $n = 0.45$, as in the experiments of Xiao et al. (2017). Achieving the target porosity at the target isotropic stress was a product of calibration of the reduced friction angle. The unbalanced force ratio was monitored to ensure that the sample was at an equilibrated state during isotropic compression, where $uf = \sqrt{\left(\sum_1^{n_b} (\text{unbalanced forces})^2/n_b\right) / \left(\sum_1^{n_c} (\text{contact forces})^2/n_c\right)}$ (as in Ng, 2006), where n_b, n_c the number of particles and contact forces, respectively. The critical time step was calculated as $\Delta t = 0.6 \cdot \Delta t_{crit}$, where the critical time step was found based on the sonic speed E/ρ (Young's modulus over density), so that an elastic wave does not propagate farther than the minimum distance of integration points l_{min} during one step, as detailed in the documentation of `YADE` (Smilauer et al., 2021). This distance is considered to be $l_{min} = \min R_i$, i.e. the radius of the smallest sphere in the simulation, resulting in $\Delta t_{crit} = minR_i \cdot \sqrt{\rho/E}$. The Hertz-Mindlin contact model was considered (no-slip solution for shear), with viscosity in the normal contact direction.

After the desired isotropic state was achieved, the inter-particle friction value was reinstated to its actual value for each simulation, varied from 20° to 45°, as shown in Table 6.1. At the very last step of the isotropic compression, and before deviatoric loading was imposed, all interactions between particles were cleared, using the routine `O.interactions.clear()` of `YADE`, in order to avoid locking stresses, which would make the sample "prestressed" (as also reported in Ahmed et al., 2016). Clearing all interactions before shearing ensures that all formed contacts will be new, without inheriting the shearing history from the preparation stage, which was somewhat artificial, under high local damping and low friction. Still, this preparation procedure of isotropic compression merits from not imposing any form of anisotropy during sample preparation; e.g. if air-pluviation was to be employed as a preparation technique, it would induce orientation fabric anisotropy and stress-induced anisotropy, as these particles are irregular, with unequal principal inertia values (and thus unequal preferences to rotate among their three different principal axes), while gravity acts along one vertical axis; thus, choosing an isotropic compression procedure for sample preparation aimed to induce the least amount of anisotropy possible, before shearing.

No local damping was considered during the deviatoric loading stage, while low viscous damping was considered in the contact normal and shear directions, using viscous damping ratios of $\beta_n = \beta_s = \beta = 0.071$, which correspond to a coefficient of restitution of $e_n = e_s = e = 0.8$, considering the formula $\beta = -(\log e)/\sqrt{\pi^2 + (\log e)^2}$, as detailed in Smilauer et al. (2021).

Shearing was carried out for strains up to 35%, aiming to allow the material reach critical state, i.e. a state of excessive deformation, where the sample keeps deforming with no further increase of deviatoric stress.

In the following sections, the mechanical behaviour of the ballast samples is quantified under triaxial shearing loading conditions, using typical measures of the critical state theory of soil mechanics (Schofield and Wroth, 1968) and the sign convention of classical soil mechanics, where compressive stress and strain components are considered positive. The deviatoric stress was calculated as $q = \sigma_1 - \sigma_3$, where σ_1 and σ_3 the major and minor principal stresses, respectively, while the mean effective pressure was calculated as $p = (\sigma_1 + \sigma_2 + \sigma_3)/3$. Volumetric strain was calculated as $\epsilon_v = \epsilon_1 + \epsilon_2 + \epsilon_3$, where ϵ_1, ϵ_2, ϵ_3 are the axial strains along the principal stress directions. The shear strength is visualised in the following graphs using the stress ratio q/p and in terms of a so called "mobilised shear strength" (or "mobilised fric-

Table 6.1: Modelling parameters of parametric triaxial tests, using multi-sphere particles.

ID	Number of particles	Friction angle°	Initial packing	Size and shape distribution	Spheres per clump
S - Size of REV					
S1	10000	30	Different initial packings	Real size and shape distribution	10
S2	7000	30			
S3	5000	30			
P - Packing arrangement					
P1=S3	5000	30	Different initial packings	Real size and shape distribution	10
P2	5000	30			
P3	5000	30			
N - Number of clump sphere-members					
N1=S3	5000	30	Same initial packing	Real size and shape distribution	10
N2	5000	30			20
N3	5000	30			30
N4	5000	30			40
N5	5000	30			50
F - Inter-particle friction angle					
F1	5000	20	Same initial packing	Real size and shape distribution	10
F2	5000	25			
F3=S3	5000	30			
F4	5000	35			
F5	5000	45			
D - Size and shape distribution					
D1=S3	5000	30	Same initial positions	Real size and shape distribution	10
D2	5000	30		Monodisperse - Compact	
D3	5000	30		Monodisperse - Flat	
D4	5000	30		Monodisperse - Elongated	
D5	5000	30		Monodisperse - Bladed	
D6	5000	30		Polydisperse - Equal	
D7	5000	30		Polydisperse - Proportional	

tion angle") ϕ, expressing the maximum ratio of shear stress to normal stress at any plane, where $\phi = \arcsin\left((\sigma_1 - \sigma_3)/(\sigma_1 + \sigma_3)\right)$, as in Ahmed et al. (2016). All stresses are considered to be effective stresses, since these tests were carried out for drained conditions.

As shown in Table 6.1, each simulation group considers a case with same simulation parameters, equal to those of case S3, allowing for comparisons as different parameters are varied. The results of this simulation are marked with black lines in all of the following Figures 6.10 to 6.15.

6.6.2 Size of REV

First, an investigation of the size of the REV was conducted, to ensure the number of clump particles used in the DEM simulations is adequate. Figure 6.10 makes it apparent that 5000 multi-sphere particles particles were adequate to simulate the critical-state shear strength of the material of interest. From a deformations point of view, looking into Figure 6.10c, all studied samples exhibit the same degree of dilative behaviour. The unbalanced force ratio in Figure 6.10d demonstrates low values, indicating that the samples remained in an equilibrated state during triaxial shearing (Ng, 2006). These REVs remain small in size because of the periodic boundary conditions (Thornton, 2000) imposed in the simulation, which prevent the consideration of boundaries, which can lead to unrealistic kinematic conditions within a granular assembly, as particles tend to align with them. If rigid boundaries were to be considered, a much larger model (in terms of number of particles) would need to be considered, in order to minimise boundary effects. This is in fact one of the merits of simulating granular materials inside a periodic cell.

6.6.3 Packing arrangement

Having established the size of the REV, a second check is carried out, aiming to test if considering a different initial arrangement of the same 5000 ballast particles will lead to the same critical-state shear strength. Three different initial packing arrangements are tested and compared, using the sample generation procedure detailed in subsection 6.6.1. As shown in Table 6.1, the particles in all three packings have the same friction angle and number of spheres per clump. The only varying factor among these three simulations is the initial position and orientation during generation of the initial packing, which is controlled by the `seed` parameter of the `makeClumpCloud` function of `YADE`, as explained in subsection 6.6.1. Using three different `seed` values result in the generation of three different initial packings of the same clump particles. After generation at a loose state, these packings are isotropically compressed up to $\sigma_3 = 60kPa$, achieving a porosity of $n = 0.45$, as in the experiments of Xiao et al. (2017), before being sheared up to axial strains of 35%. Figure 6.11 demonstrates that rearrangement of the particles leads to exactly the same shear strength, validating the representative nature of the considered REVs. Based on the principles of the Critical State Soil Mechanics theory (Schofield and Wroth, 1968), the critical state characteristics of a soil are not influenced by the ini-

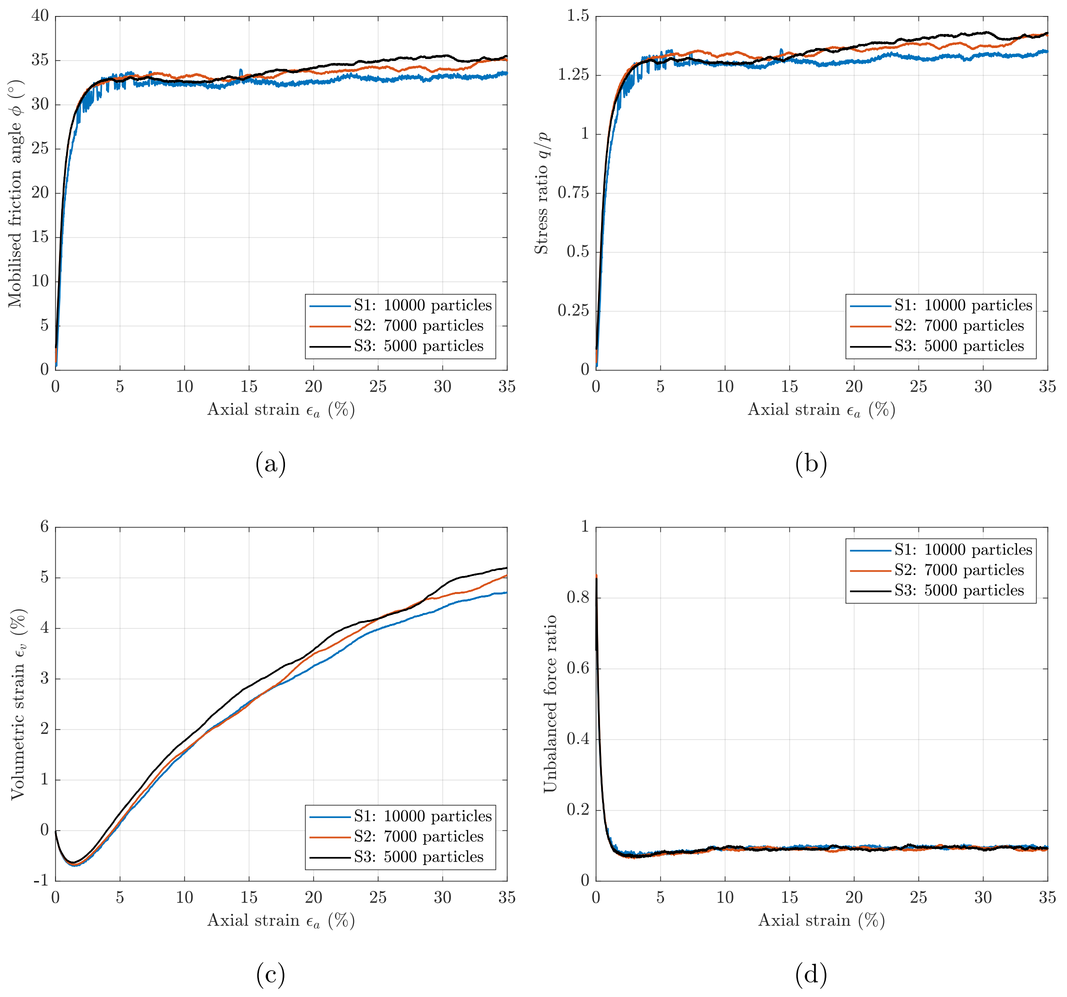
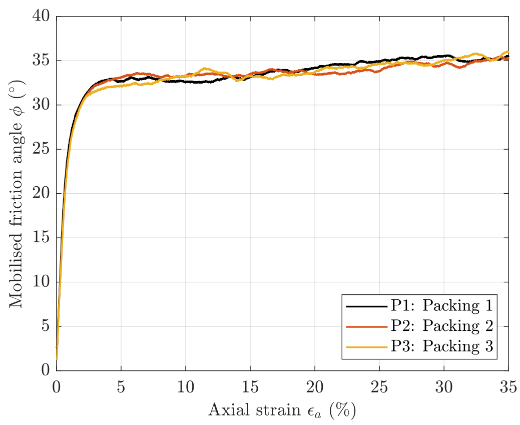


Figure 6.10: Results of triaxial tests varying the size of the REV (a) Mobilised friction angle vs axial strain; (b) Stress ratio vs axial strain; (c) Volumetric strain vs axial strain; (d) Unbalanced force ratio vs axial strain.

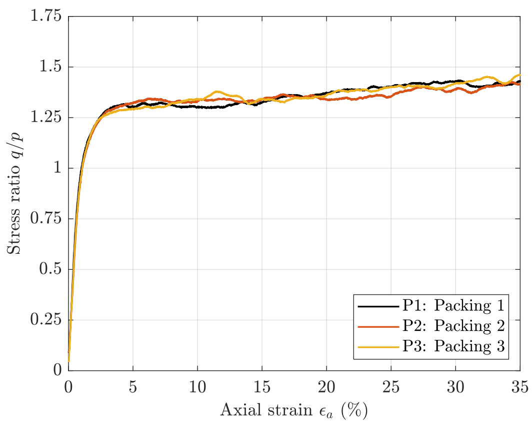
tial state of the soil, e.g. loose or dense, including the initial packing arrangement. As a result, the size of the REV consisted of 5000 clumps is deemed adequate to simulate representative behaviour of the analysed ballast material.

6.6.4 Number of clump sphere-members

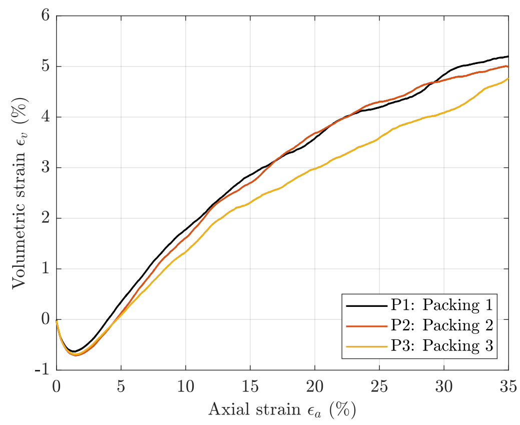
When multi-spheres are used to approximate the morphology of real particles, the number of spheres per particle is a significant parameter, as it relates (a) to how closely morphological features are captured by the multi-sphere approach, and (b) to the computational time, as each added sphere increases the consumption of memory



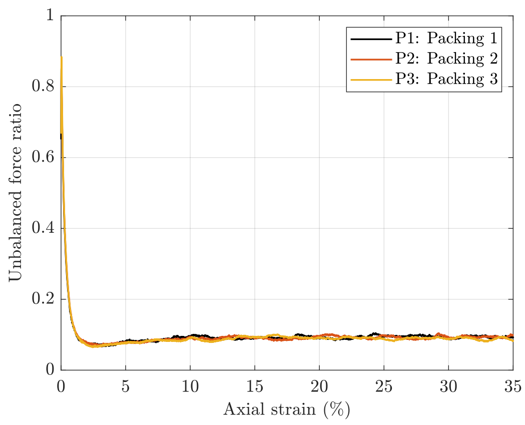
(a)



(b)



(c)



(d)

Figure 6.11: Results of triaxial tests varying the initial packing arrangement (a) Mobilised friction angle vs axial strain; (b) Stress ratio vs axial strain; (c) Volumetric strain vs axial strain; (d) Unbalanced force ratio vs axial strain.

used to track the state of the particle (e.g. its position), used to facilitate contact detection with other particles. Figure 6.12 demonstrates that for the particles studied in this case study, 10 spheres sufficed to represent their shear strength, as the addition of more spheres per particle did not lead to any differentiation in the results in terms of mobilised friction angle. Interesting to note the effect of the number of spheres on the dilatancy in terms of volumetric strain shown in Figure 6.12c, where considering more spheres per particle led to an increase of volumetric strain. Specifically, comparing the results of triaxial behaviour with clumps of 10 spheres and 50 spheres, the volumetric strain shows a discrepancy from 5.1% to 6.4%, respectively.

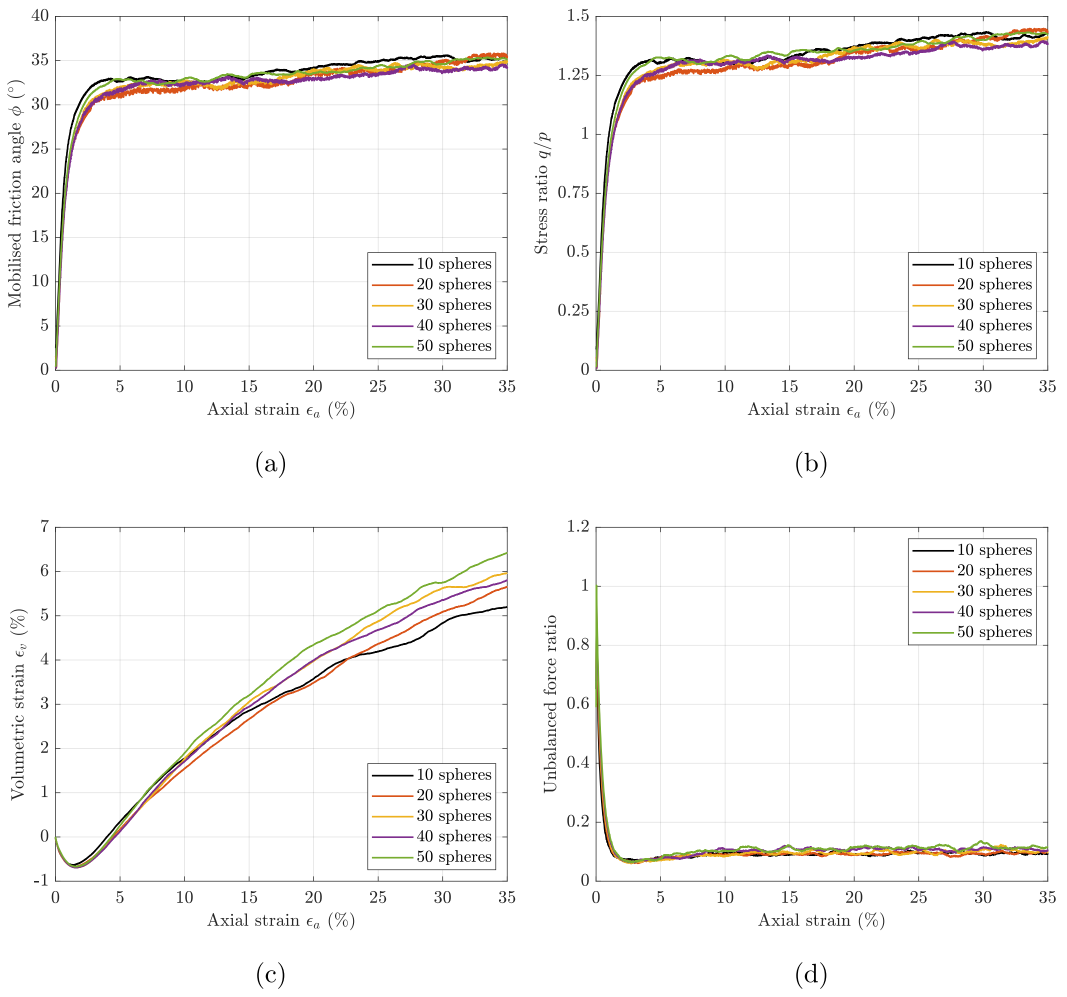


Figure 6.12: Results of triaxial tests varying the number of spheres of each clump (a) Mobilised friction angle vs axial strain; (b) Stress ratio vs axial strain; (c) Volumetric strain vs axial strain; (d) Unbalanced force ratio vs axial strain.

6.6.5 Inter-particle friction

Inter-particle friction plays a significant role in the shear strength of granular materials. Figure 6.13 illustrates the effect of the inter-particle friction angle (at a micro level) on the mobilised friction angle (at a meso level). The trend of the results is in agreement with expected values from similar studies (such as Ng, 2006; Modenese, 2013). The higher inter-particle friction values led to a higher peak shear strength, which then decreases when the material reaches critical state. A potential justification for this is that high values of inter-particle friction provide a more stable structure, resulting in nearly simultaneous buckling of the strong force chains, while low inter-particle friction values lead to some of the strong force chains yielding while others have not yet reached the sliding limit, and yielding happens more gradually (Barreto and O'Sullivan, 2012).

Looking into Figure 6.11c, higher inter-particle friction values led to higher degrees of dilatancy, as expected. As discussed in Huang (2014), the shear strength of a sample drops more rapidly post-peak for higher µ values, a fact that is evident for the sample with inter-particle friction of 45° in Figure 6.13a. Though, the strength of this sample at large strains falls below the samples with lower inter-particle friction values, which is counter-intuitive. This is in agreement with the findings of Huang et al. (2014), where high friction coefficients in DEM simulations were reported to lead to unrealistic behaviour.

Another reason for the rapid drop of shear strength in the case of high friction is that due to the high dilative behaviour, the sample is mobilised more and the material is not equilibrated at large strains compared to the simulations for lower inter-particle friction values. This can be evidenced by the high unbalanced force ratio in Figure 6.13d, compared to the simulations for smaller angles of inter-particle friction. A mitigation strategy for this would be to repeat the simulation with a lower strain-rate, which is not carried out here, as the high inter-particle friction coefficient in this case is deemed unrealistic.

6.6.6 Particle shape distribution

The statistical distribution of particle size and shape affects the stiffness and interlocking of granular materials. In this group of tests, the shear strength of the real material was compared with simplified distributions of particle shape. To take particle size out of the equation, the mean diameters D50 of the sample was considered as an average of the sizes present in the sample. Then, a compact, a flat, a bladed and an elongated particle were selected, all with sizes close to the average particle size of the material, which are the same particles shown in Figure 6.8b.

Monodisperse samples made of these particles were sheared up to 35% strain, aiming to quantify the effect of using particles with different shapes on the shear strength. Regarding the effect of particle shape on the shear strength, Figure 6.14 makes it apparent that the monodisperse sample considering the compact particle led to the lowest critical state shear strength, while the monodisperse sample with the flat and elongated particle led to the highest recorded shear strengths at critical

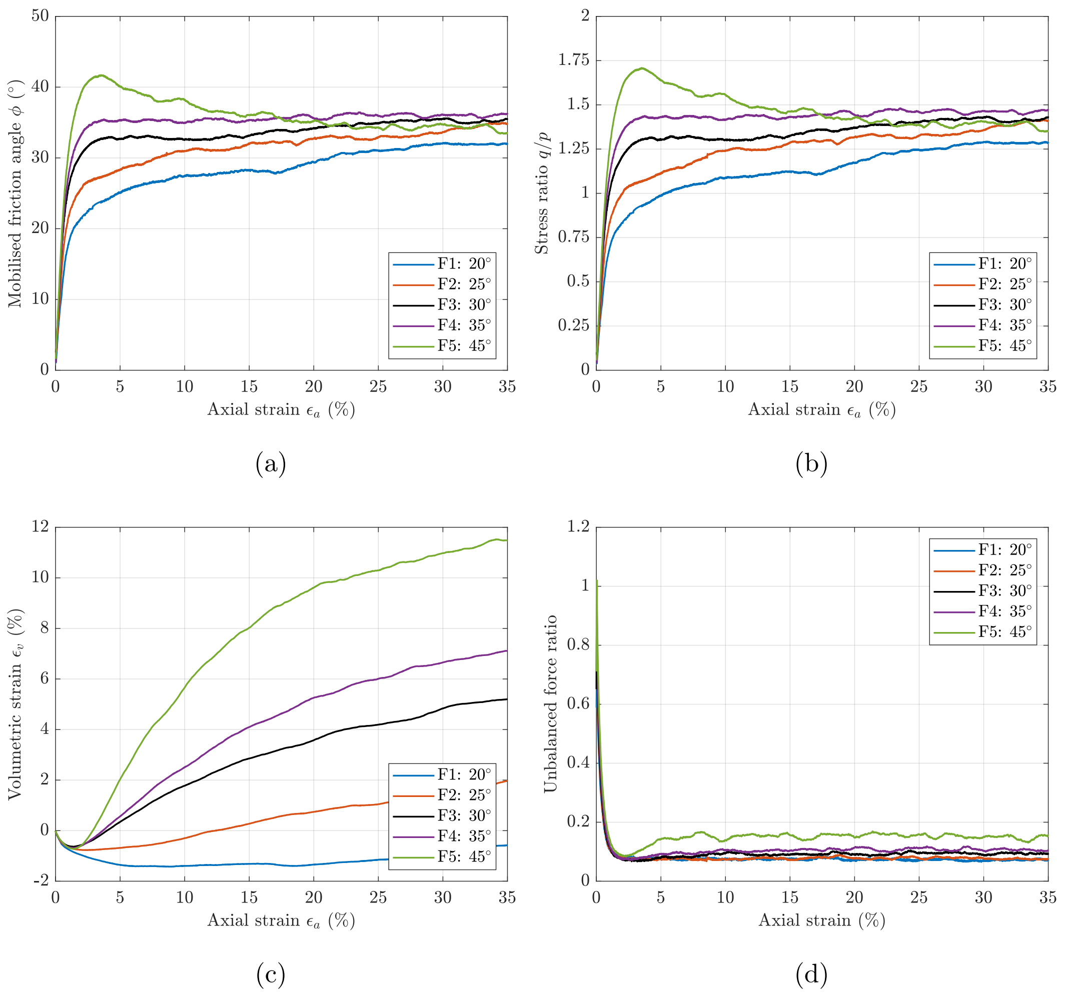


Figure 6.13: Results of triaxial tests varying the inter-particle friction (a) Mobilised friction angle vs axial strain; (b) Stress ratio vs axial strain; (c) Volumetric strain vs axial strain; (d) Unbalanced force ratio vs axial strain.

state. This is in conceptual agreement with the discussion regarding the "effective form" index proposed by Harkness and Zervos (2019) (see relevant discussion in Chapter 2), where compact particles take low values of effective form (and shear strength), while the elongated and flat ones take higher values.

Then, two simplified polydisperse packings were generated, using the four selected particles of different form, aiming to quantify the effect of shape polydispersity on the shear strength. In case D6 (see Table 6.1), the packing was generated using the four different particles with equal percentages, i.e. 25% of the sample was made of the compact, 25% of the flat, 25% of the elongated and 25% of the bladed particle. On the other hand, in case D7 these particles were used with the same

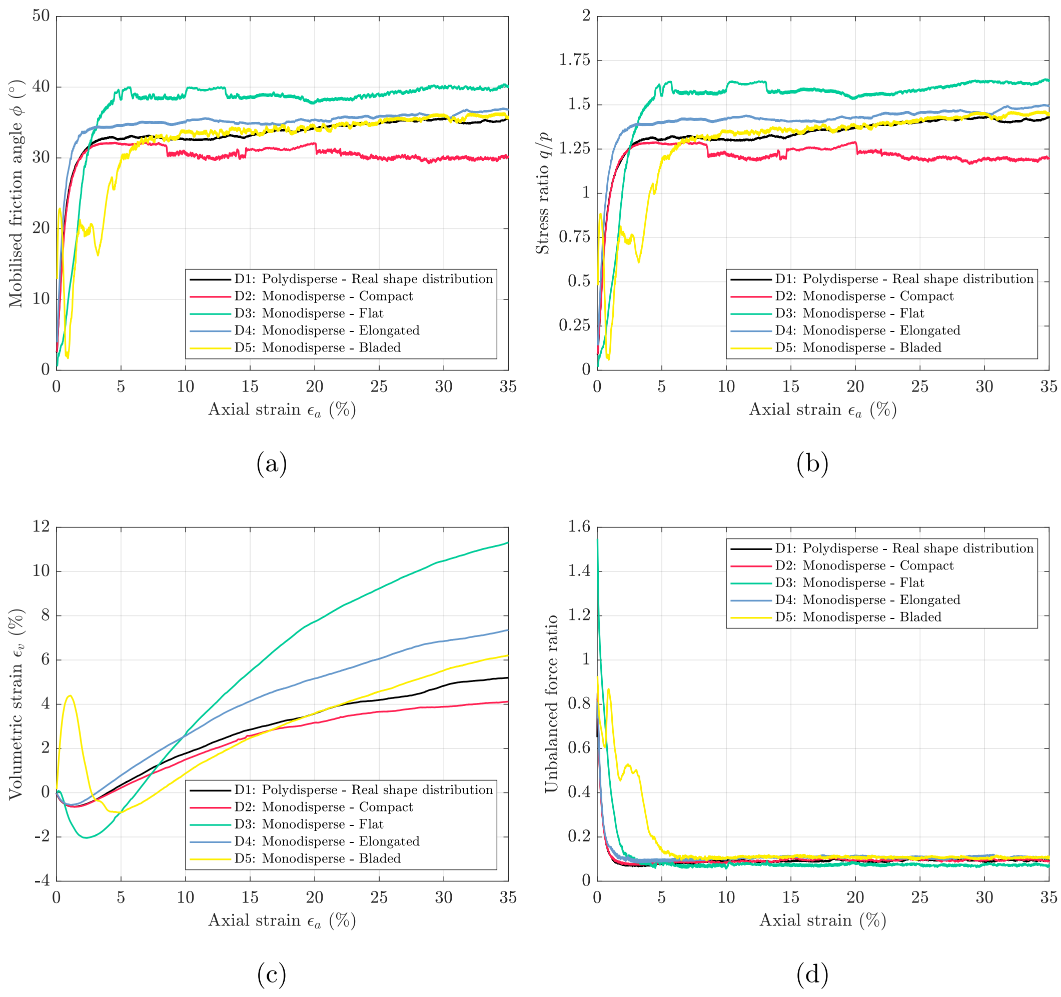


Figure 6.14: Results of triaxial tests for monodisperse samples (a) Mobilised friction angle vs axial strain; (b) Stress ratio vs axial strain; (c) Volumetric strain vs axial strain; (d) Unbalanced force ratio vs axial strain. The results of the monodisperse samples are coloured in accordance to the four shape classes as in Figure 6.8b.

percentages that compact, flat, bladed and elongated particles appeared in the material of interest (here the 100 analysed ballast particles), for the original fidelity level. In particular, using the classification system proposed in Chapter 2, the studied ballast particles were classified in the four classes of particle form, demonstrating percentages of: 37% flat, 23% compact, 9% bladed and 31% elongated particles, as illustrated in Figure 6.7d. These percentages were used to assemble the packing of simulation D7.

The simplified polydisperse packings consisted of only four different particles with average particle size demonstrate interesting behaviour patterns, shown in Figure 6.15. Packing D6, where the four particles were assigned with equal percentages overestimated shear strength, in comparison to the average baseline case D1 (which considers the real shape distribution) so it is not a good candidate to represent the shear strength of the real material. On the other hand, packing D7, where the simplified particles were assigned with percentages proportional to the percentage of similarly-shaped particles in the real material demonstrates a very close match when compared with the baseline case D1, both in terms of mobilised (macromechanical) friction angle, and dilatancy. Regarding the latter, the simplified polydisperse sample D7 exhibits slightly higher dilative behaviour than sample D1. Interesting to note how the monodisperse packing of flat particles D3 and the polydisperse sample with equal percentages of the four particles of different form D6 overestimate the dilative behaviour of the material, unlike D7, which managed to represent adequately the shear behaviour of the material, both in terms of shear strength and dilatancy. This indicates that simplified shape distributions have the capacity to represent the shear behaviour of granular materials in numerical simulations with irregular particles, as long as the shape distribution of the original material is somehow approximated.

6.7 Conclusions

This chapter presented a methodology to generate Representative Element Volumes (REVs) in the DEM using realistic particle shapes, along with an application of the methodology to railway ballast particles.

Results of morphological and mechanical characterisation of railway ballast were presented, at various fidelity levels, including the original fidelity level of scanning and simplified levels of multi-sphere particles generated using the Euclidean-distance transform approach introduced in Chapter 4.

Parametric triaxial tests were carried out for the ballast particles, at the scale of Representative Element Volumes, varying several modelling parameters and documenting their effect on the shear strength of the material. The varied parameters included (i) the size of the REV, (ii) the initial packing arrangement, (iii) the number of spheres making each clump, (iv) the inter-particle friction and (v) the particle shape distribution.

Regarding the latter, four monodisperse samples were sheared, made of only flat, compact, bladed and elongated particles, which were not able to represent the shear strength of material, compared to a sample following the real size and shape

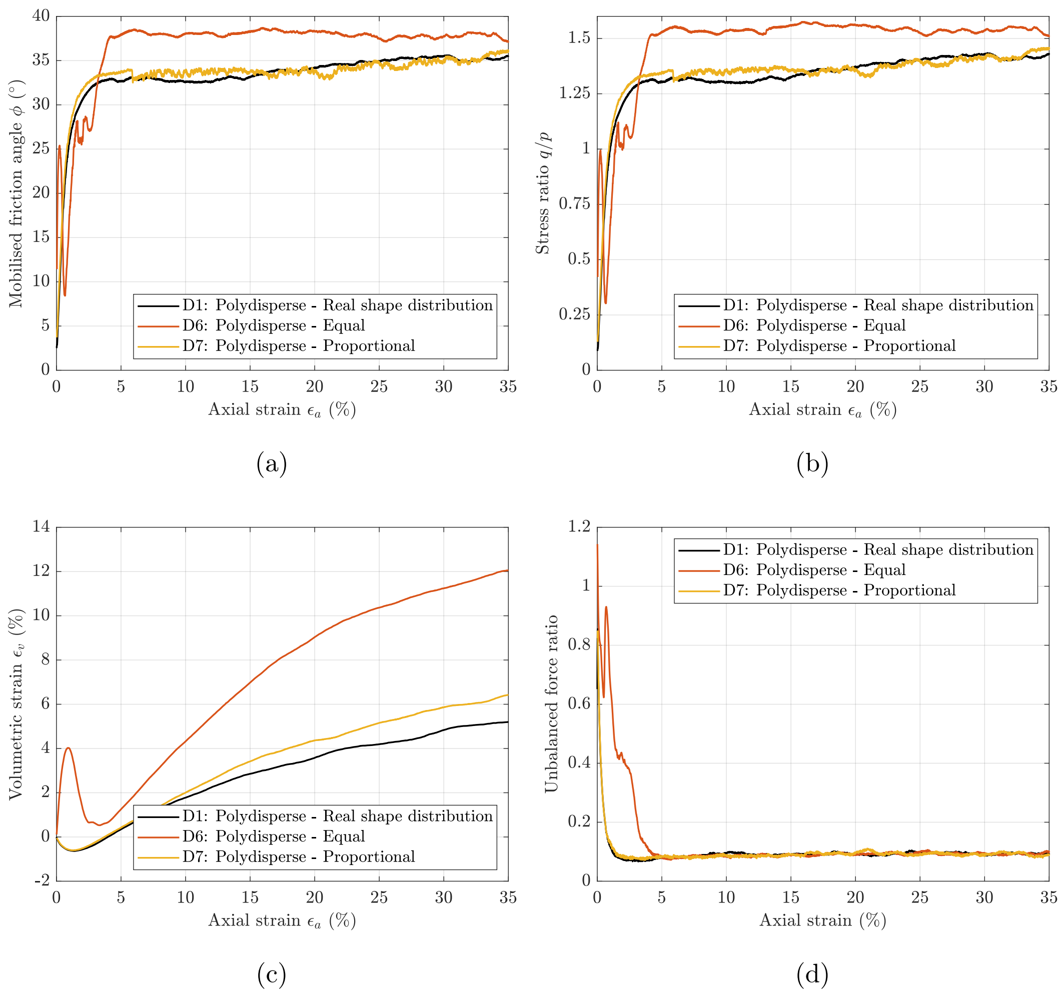


Figure 6.15: Results of triaxial tests varying the particle shape distribution (a) Mobilised friction angle vs axial strain; (b) Stress ratio vs axial strain; (c) Volumetric strain vs axial strain; (d) Unbalanced force ratio vs axial strain.

distribution. Then, two simplified polydisperse samples were tested, made of only four different particles, with each particle representing one of the shape classes of the system proposed in Chapter 2. It was found that if these four particles are used in proportions to the classification of the fully polydisperse material, using the system proposed in Chapter 2, they can provide an adequate representation of its shear strength.

Chapter 7

Concluding remarks

7.1 General remarks

This thesis aimed to contribute to the study of image-informed modelling of particulate systems, via the development of analytical and numerical tools. The importance of particle morphology across scales is undeniable, demonstrated in a plethora of experimental, numerical and analytical studies and applications.

High-resolution image acquisition of particles is nowadays becoming more accessible and low-cost, enhanced by recent technological advances. As a result, developing accessible and user-friendly numerical tools which can process this increasing influx of imaging data in an automated and robust way is an integral step in utilising imaging data for particle shape characterisation or to inform parameters of numerical simulations.

The traditional characterisation of particle morphology is still to date highly subjective. Thus, a comparison of the most prominent characterisation indices from the literature is essential in deciding which indices are informative and correlate to mechanical, rheological or hydraulic behaviour.

This thesis aimed to facilitate a link between particle morphology and mechanical behaviour at the microscopic and mesoscopic scales, by developing numerical tools to link particle image acquisition and shape characterisation with numerical modelling. To this end, the `SHAPE` code was shared in an open-source manner, aiming to enrich the discussion on shape characterisation, offering transparent calculation of several shape parameters for different aspects of particle morphology. SHAPE was developed out of the lack of open-source shape analysers of three-dimensional particles and the necessity to have controlled parameters and assumptions during shape characterisation, a procedure where standardisation is lacking and user-dependency is high. The efforts towards image-informed numerical modelling of particulate systems were further supported within this thesis via the code development of `CLUMP`, yet another one open-source code, which generates multi-sphere particles from available imaging data of various types. `CLUMP` was developed out of pure necessity to generate multi-sphere particles in a systematic and predictable way, balancing morphological accuracy and computational efficiency. To achieve this, a new clump-generation approach was developed for the purposes of this thesis, which is based on the Euclidean distance transform of three-dimensional images. Reflecting on the combined utility provided by `SHAPE` and `CLUMP`, the work produced as part of this thesis aimed to provide intelligible and comprehensive numerical tools for the characterisation and simplification of irregular particles and shed light on gray areas arising from the practicalities of performing shape characterisation.

Considering the ever-growing amount of literature introducing new indices to characterise particle morphology, it is paramount to isolate the indices that show a correlation to physical behaviour. During this thesis, a new system for the characterisation and classification of particle form was developed. Using these indices to assemble packings of low shape polydispersity led to a good estimation of shear strength, indicating that these indices have the capacity to characterise particle form in a comprehensive way.

Two codes for the modelling of sharp and rounded non-spherical particles were maintained and further developed. As part of this thesis, their user-friendliness was enhanced, new contact laws were developed for them and their contact detection algorithms were expanded to work in periodic space. This allowed for a downsizing of the numerical models of the granular samples analysed in this thesis by simulating them inside a periodic cell, where boundary effects do not exist and cannot affect the reliability of the numerical simulations.

The mechanical bulk behaviour of railway ballast was characterised, aided by the morphological features of individual particles. A methodology to represent the shear strength of materials with irregular particles was laid out and simplified samples of low polydispersity were produced, which demonstrated same levels of shear strength.

It should be noted that the findings of this thesis apply to the mechanical characterisation of dense packings of railway ballast during quasi-static, triaxial tests and should not be generalised for more complex settings. More work is required for packings at a loose state, as well as packings undergoing dynamic processes. For instance, a different level of shape polydispersity might be needed to characterise the rheological behaviour of irregular particles in a granular flow. The proposed methodology of creating samples which follow a specific statistical shape distribution can be used to establish different levels of needed shape polydispersity for different application, loading and kinematic settings.

Overall, this thesis aimed not only to answer research questions in itself, but to also equip colleagues within the shape-characterisation and numerical-modelling research communities with open-source numerical tools, which allow for a quantified characterisation of physical materials and an image-informed generation of numerical models.

7.2 Specific remarks

Specific conclusions are drawn for each chapter of this thesis, aiming to highlight the impact and contributions made, in the form of tangible outcomes.

Chapter 2 compared several indices used to characterise particle form.

- Mapping shape indices on a Zingg plot (Zingg, 1935), where particles of all possible aspect ratios can be represented, revealed a means of studying how the value of each shape parameter varies for different particles in a quantified manner.

- Plotting the degree of true sphericity (Wadell, 1932), for both cuboids and ellipsoids, along with the intercept sphericity (Krumbein, 1941) and the maximum projection sphericity (Sneed and Folk, 1958) on a Zingg plot led to the conclusion that the latter two indices measure the compactness of a particle, rather than its morphological resemblance to a sphere, as they take same values for ellipsoids and cuboids with same aspect ratio.

- It is shown that all the analysed measures of sphericity take high values (>0.60) for the particles of most natural materials, such as geomaterials, which appear at the upper right triangular region of a Zingg plot.

- Also, it is demonstrated that the intercept sphericity and maximum projection sphericity take very similar values for all aspect ratios.

- A comparison of three sets of indices for particle flatness, elongation and compactness (Kong and Fonseca, 2018; Bagi and Orosz, 2020; Potticary et al., 2015), highlighted that every of the existing set bears different limitations.
- Albeit simple to define, the indices of Kong and Fonseca (2018) do not represent flatness and elongation as percentages of an overall form, and inherit the limitation of the Zingg system in misclassifying very flat and very elongated particles as bladed.
- The indices of Bagi and Orosz (2020) and Potticary et al. (2015) are considered an improvement, as they correspond to percentages of an overall form (i.e. they add up to unity), but take counter-intuitive values in regions of very flat and very elongated particles, while the distribution of their values on Zingg plot indicates that they cannot be used to define a classification system.
- A proposed set of indices for flatness, elongation and compactness resolve the issues identified in the aforementioned indices, while the distribution of their values on a Zingg plot shows they can be used to define a classification system which is in close agreement with the Zingg system for most regular particles of medium to high compactness, while improving on the classification of very flat and very elongated particles.

Chapter 3 introduced `SHAPE`, an open-source code for shape characterisation and simplification of three-dimensional particles.

- `SHAPE` allows for the automated processing of thousands of particles within minutes and with consistent analysis parameters, minimising user-dependency and user-induced error.
- The user is in control of the analysis parameters, and can compare different methods and their effect on values of the shape descriptors, as various options are offered to calculate particle characteristics. For instance, the main particle dimensions used to calculate several indices of particle form can be calculated using a fitted ellipsoid or an oriented bounding box. The latter can be calculated either using the Principal Component Analysis (PCA) or a box of minimal (i) volume, (ii) surface area or (iii) sum of edges. This feature also offers modelling versatility, as different methods can be more appropriate for particles of different morphology.

- A transparent calculation and comparison of different sets of indices quantifying sphericity, flatness, elongation and compactness, roundness and angularity, offers a rich and wide range of results, where the user can decide which index is more representative for the morphology of the analysed particles, instead of employing a single index, as all indices have limitations and an objectively best index does not exist.

- `SHAPE` connects imaging, characterisation and generation of numerical models seamlessly, as simplified polyhedra can be generated in an automated manner for varying fidelity levels. Simplification of a single sand grain showed that a simplified convex polyhedral particle with as many as 25 faces led to adequate preservation of particle form when compared to the original fidelity level. Similar analysis for a sample of 50 ballast grains showed that polyhedra with up to 50 faces preserved their form adequately, with few exceptions, prompting the conclusion that each material is different and a universally adequate number of faces cannot be proposed for all particles. This highlights the need for tools like `SHAPE`, where automated characterisation and simplification of particles in bulk scales are possible and computationally efficient.

- Shape characterisation of 3D images of an ellipsoid with various levels of noise and blur, varied in a parametric manner, showed that image quality affected the obtained morphological results by as much as 25%, for the analysed range of parameters.

Chapter 4 introduced `CLUMP`, an open-source code for the generation and analysis of three-dimensional, multi-sphere particles.

- `CLUMP` allows for the automated generation of multi-sphere particles of several thousands particles within minutes, given imaging data from various sources.

- Implementations of two of the most popular clump-generation techniques are available, for axisymmetric (Favier et al., 1999) and irregular particles (Ferellec and McDowell, 2010).

- The strengths and limitations of the method proposed by Ferellec and McDowell (2010) were identified, as it can generate particles with reduced artificial asperities, but only for large numbers of spheres per clump, as the generation of each sphere happens at a random point at the particle surface.

- A new clump-generation approach was proposed, based on the Euclidean distance transform of three-dimensional images. The new method provides systematic particle generation, where each new sphere is smaller than the previous one, and is generated where the mass of the particle is least represented, utilising the underlying principles of the Euclidean distance transform. The new method can generate clumps with controlled degrees of overlap, supporting both clumps of overlapping spheres and clusters of touching spheres.

- A method was provided to tessellate the surface of a clump, along with an algorithmic implementation, which allows for a full shape characterisation of multi-sphere particles. Using this feature, a modeller can characterise the actual particles used in a simulation, and not just the scans of the original fidelity level. This approach provides a means of quantifying if a clump is an oversimplification of the real particle from a morphological standpoint.

- Triaxial tests of rice and sand grains using clumps generated with different approaches demonstrated that the clump-generation approach has an effect on the observed shear strength of granular materials. In particular, modelling a rice grain with an axisymmetric clump created using the method of Favier et al. (1999) led to significantly different behaviour compared to clumps generated with other methods, as the axisymmetric clump had zero flatness and thus did not represent the morphology of the real rice grain.

Chapter 5 discussed the `Potential Particles` and `Potential Blocks` codes, developed within YADE (Smilauer et al., 2021), which were further developed as part of this thesis.

- A documentation of these codes was developed as part of this thesis, aiming to clarify the meaning of all modelling parameters involved and their intended way of usage. This aimed to widen the user-base of the code, as little practical information was available on how to use these codes properly before the development of this documentation.

- The source code of the `Potential Blocks` was further developed to support automatic calculation of the vertices, volume, centroid and inertia tensor for each new particle, in an automated manner.

- The contact detection algorithms and contact laws of both codes were modified to become compatible with the periodic boundaries already developed in `YADE`.
- The contact laws of both codes were expanded, via the development of a linear contact law for the `Potential Blocks` and a non-linear contact law for the `Potential Particles`, utilising a calculation of the contact area.
- A new script was developed in OpenGL to achieve accurate and lightweight visualisation of `Potential Blocks`, as initially visualisation was available only in a third party format (`vtk`).
- Energy calculations were developed for both codes, corresponding to elastic stored strain energy and dissipation due to viscous damping and sliding friction, aiming to quantify the transference of energy from one form to another.
- An investigation of energy conservation was carried out for the `Potential Blocks` code and different contact scenarios, noting conservation of energy for face-to-face contacts and energy imbalance for more complex contact scenarios. More work is required to identify the source of the latter, i.e. whether it is related to the methods calculating the contact characteristics (such as contact point, contact normal, contact law) or to the methods measuring the various energy components.
- The `Potential Particles` code was employed to simulate a 3D-printed material in a round robin study for the angle of repose for plane-strain and axial symmetric and repose states. In both cases, considering a convex potential particle to simulate the originally concave particles led to an underestimation of the angle of repose by more than 5°. Varying other parameters, such as the ratio of shear to normal stiffness (k_s/k_n) or the density of the material did not show a strong effect on the resulting angle of repose values, for the studied range of parameters.

Chapter 6 carried out a morphological and mechanical characterisation of railway ballast, via quantitative shape characterisation and image-informed discrete element simulations.

- A new methodology was proposed to generate representative element volumes in the DEM using realistic particle shapes, which represents both particle size and shape. The methodology is based on the principle that simplified particles should follow the same statistical particle shape distribution as the particles at their original fidelity level.

- Parametric drained triaxial tests were conducted in periodic space for packings of railway ballast using multi-sphere particles. Periodic boundaries were employed to simulate perfect triaxial conditions, without the influence of boundary effects. A sample size of 5000 clumps led to representative behaviour, which was tested for different initial packing arrangements.
- Comparing the results for clumps with 10, 20, 30, 40 and 50 spheres per clump, all fidelity levels led to similar levels of shear strength, when they followed the original particle shape distribution. This finding indicates that very simple clumps of even 10 spheres can led to representative behaviour, as long as they follow the same morphological statistical "profile" of the material at the original fidelity level.
- A parametric set of simulations for varying inter-particle friction angle showed increased peak strength for increasing friction coefficients, while a friction angle of 45° displayed unrealistic behaviour, with the critical state shear strength declining below the strength of samples with smaller inter-particle friction angles.
- Four monodisperse simulations using only (i) flat, (ii) compact, (iii) bladed and (iv) elongated particles did not manage to approximate adequately the shear strength of the material with the real shape distribution.
- A simplified polydisperse sample made of these four particles with equal percentages of appearance also failed to approach the real shear strength of the material.
- Another simplified polydisperse sample was simulated, where each of the four particles was used in proportion to the percentages of flat, compact, bladed and elongated particles for the original fidelity level, using the classification system proposed in Chapter 2. This sample generation approach led to a good estimation of the shear strength of the material with real shape distribution, using only four different particle shapes. This shows that models of simplified polydispersity and simplified fidelity can provide a good estimation of shear

strength, as long as they follow the proposed logic of shape-informed sample generation.

7.3 Suggestions for future work

This section proposes ideas for future work, which were considered out of the scope of this thesis. Particle shape characterisation and image-informed numerical modelling are two active areas of research, with a wide spectrum of possible applications. This thesis laid down a path for the combination of analytical, experimental and numerical tools, in the context of quantitative shape characterisation, and how they can be used to generate representative numerical models informed by real data.

7.3.1 Improvement of the classification system

Following the discussion in Chapter 2, the proposed classification system can be further refined for cases where $co > fl$ that are currently classified as flat and cases where $co > el$ that are currently classified as elongated. However, the proposed classification system in its current form led to a robust prediction of mechanical behaviour in triaxial testing. An improvement of the system could incorporate compactness into the classification process, like the system of Sneed and Folk (1958), as it is currently only involving flatness and elongation and to increase the number of classes, considering intermediate classes of particle shape, such as "compact-platy", "compact-bladed", "compact-elongated", like the system of Sneed and Folk (1958). Such a modification could be useful in other applications.

7.3.2 Further development of SHAPE and CLUMP

Releasing the `SHAPE` and `CLUMP` codes in an open-source manner aimed to enhance transparency of calculations during shape characterisation, a research area with high degrees of subjectivity and user dependency. The further development of these codes by future users, with the addition of new shape indices and methods to calculate auxiliary geometries (for `SHAPE`) or new methods to generate multi-sphere particles (for `CLUMP`) can offer a deeper comparison between methods and help identify delicate points and good/bad practises during particle shape characterisation and clump generation. The developed codes are applicable to problems across disciplines, such as characterising biomaterials, e.g. cells, microplastics, environmental catalysts, battery materials, particulates and other pollutants.

7.3.3 Development of Potential Blocks & Potential Particles

Following the discussion in Chapter 5, it is worthwhile to investigate the reasons that led energy to not be conserved for complex contact scenarios, and whether it is an issue with theoretical roots or an artifact of faulty energy measurements. Developing more contact laws for the `Potential Blocks` and the `Potential Particles` codes can make them applicable to a new range of applications. In particular, developing a modified Hertzian-like, curvature-dependent contact law for the *Potential Particles*, based on the values of the local Hessian matrix around the overlap region of each contact, as in Harkness (2009), can lead to more comparable results with models of spheres, where using the Hertz-Mindlin contact law (no-slip solution) is the norm. The current non-linear contact law relies on two volumetric stiffness parameters, a normal and a shear one (k_n, k_s), which are not comparable to the elastic parameters used in the Hertzian formulation (E, ν), while the relationship of force to contact area has a different exponent. A curvature-dependent calculation of contact forces can create the opportunity to use well-established contact laws for adhesive contacts, such as the JKR model (Johnson, 1987), models for bonded particles or contact laws for rough contacts (e.g. Otsubo et al., 2017).

7.3.4 Other modelling techniques for irregular particles

The triaxial tests in Chapter 6 were carried out using multi-sphere particles, which cannot represent angular particle morphologies for small numbers of spheres. Convex polyhedra are the most common modelling approach after multi-sphere particles, but can underestimate the interlocking capabilities of the material, while overestimate the volume of the particles. Alternative modelling methods, such as the µFE (Nadimi et al., 2020) or the LS-DEM (Kawamoto et al., 2018), have the capacity to model interlocking with a higher fidelity, via a more accurate representation of particle shape in the numerical models, including concave particles. An open-source implementation of the latter has been recently developed in YADE (Duriez and Galusinski, 2021), where the computational cost of the method has been quantified (Duriez and Bonelli, 2021).

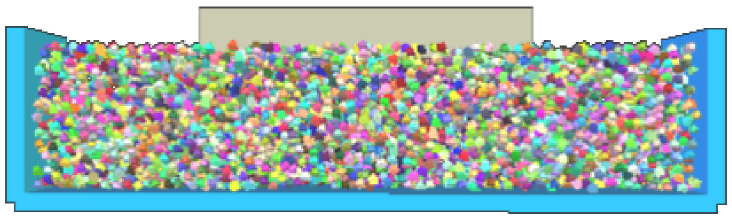
7.3.5 Incorporating roundness and angularity

This thesis focused on the influence of particle form on the shear strength of granular materials with irregular grains. Incorporating roundness and angularity into a future investigation can provide insights on its contribution to critical state shear strength, as it is expected to play a role in the interlocking among particles. Harkness and Zervos (2019) demonstrated preliminary data of the effect of angularity on the shear strength at critical state. Implementing a characterisation and classification of angularity can also inform the methodology for the generation of Representative Element Volumes that take into account particle shape parameters.

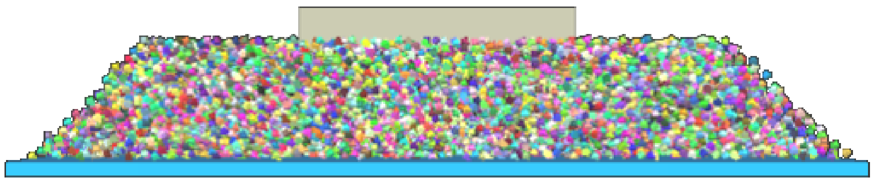
7.3.6 Modelling Boundary Value Problems

After the mechanical characterisation of the material at the REV level as shown in Chapter 6, the material parameters are established for the particle size and shape distribution that was considered. Using the same combination of material parameters, and particle size and shape, larger assemblies of the material of interest can be generated, which are expected to have similar mechanical performance, due to the rigorous selection of particles during packing generation. This opens up the possibility to simulate boundary value problems (BVPs), in this case stemming from the field of railway engineering, like in Xiao et al. (2020).

Figure 7.1 demonstrates two three-dimensional models of railway ballast, for tracks with lateral confinement and for tracks with a shoulder. These models can be used to study the load dispersion along the depth of a ballasted track, using a single or multiple sleepers, while considering irregular particles. Periodic boundary conditions can be employed to minimise boundary effects in the direction across the longitudinal axis of the track. Employing periodic boundaries allows for a downsizing of the model in terms of number of particles, offering a computationally affordable solution for the study of boundary value problems using three-dimensional models.



(a)



(b)

Figure 7.1: DEM modelling of ballasted tracks: (a) Track with lateral confinement; (b) track with shoulder.

Bibliography

V. Abaqus. 6.14 documentation. *Dassault Systèmes Simulia Corporation*, 651:6–2, 2014.

S. Abdel-Hamid, F. Alshihabi, and G. Betz. Investigating the effect of particle size and shape on high speed tableting through radial die-wall pressure monitoring. *International journal of pharmaceutics*, 413(1-2):29–35, 2011.

S. Ahmed, J. Harkness, L. Le Pen, W. Powrie, and A. Zervos. Numerical modelling of railway ballast at the particle scale. *International Journal for Numerical and Analytical Methods in Geomechanics*, 40(5):713–737, 2016. doi: 10.1002/nag.2424.

H. M. B. Al-Hashemi and O. S. B. Al-Amoudi. A review on the angle of repose of granular materials. *Powder technology*, 330:397–417, 2018.

M. Alizadeh, A. Hassanpour, M. Pasha, M. Ghadiri, and A. Bayly. The effect of particle shape on predicted segregation in binary powder mixtures. *Powder Technology*, 319:313–322, 2017.

F. N. Altuhafi, M. R. Coop, and V. N. Georgiannou. Effect of particle shape on the mechanical behavior of natural sands. *Journal of Geotechnical and Geoenvironmental Engineering*, 142(12):04016071, 2016.

N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, 1998.

J. E. Andrade, K.-W. Lim, C. F. Avila, and I. Vlahinić. Granular element method for computational particle mechanics. *Computer Methods in Applied Mechanics and Engineering*, 241:262–274, 2012.

D. André, I. Iordanoff, J.-L. Charles, and J. Néauport. Discrete element method to simulate continuous material by using the cohesive beam model. *Computer Methods in Applied Mechanics and Engineering*, 213:113–125, Mar. 2012. doi: 10.1016/j.cma.2011.12.002.

V. Angelidakis, S. Nadimi, M. Otsubo, and S. Utili. CLUMP: A Code Library to generate Universal Multi-sphere Particles. *SoftwareX*, 15:100735, 2021a.

V. Angelidakis, S. Nadimi, and S. Utili. SHape Analyser for Particle Engineering (SHAPE): Seamless characterisation and simplification of particle morphology from imaging data. *Computer Physics Communications*, 265:107983, 2021b.

V. Angelidakis, S. Nadimi, and S. Utili. Elongation, flatness and compactness indices to characterise particle form. *Powder Technology*, 396:689–695, 2022.

A. Angus, L. A. A. Yahia, R. Maione, M. Khala, C. Hare, A. Ozel, and R. Ocone. Calibrating friction coefficients in discrete element method simulations with shear-cell experiments. *Powder Technology*, 372:290–304, 2020.

D. Antypov and J. Elliott. On an analytical solution for the damped hertzian spring. *EPL (Europhysics Letters)*, 94(5):50004, 2011.

K. Bagi and Á. Orosz. A new variable for characterising irregular element geometries in experiments and DEM simulations. In *ECMS*, pages 256–260, 2020.

D. Barreto and C. O'Sullivan. The influence of inter-particle friction and the intermediate stress ratio on soil response under generalised stress conditions. *Granular Matter*, 14(4):505–521, 2012.

P. Barrett. The shape of rock particles, a critical review. *Sedimentology*, 27(3): 291–303, 1980.

S. J. Blott and K. Pye. Particle shape: a review and new methods of characterization and classification. *Sedimentology*, 55(1):31–63, 2008.

C. Boon. *Distinct Element Modelling of Jointed Rock Masses: Algorithms and Their Verification.* PhD thesis, University of Oxford, 2013. URL `http://www2.eng.ox.ac.uk/civil/publications/theses/boon-1`.

C. Boon, G. Houlsby, and S. Utili. A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method. *Computers and Geotechnics*, 44:73 – 82, 2012. ISSN 0266-352X. doi: 10.1016/j.compgeo.2012.03.012.

C. Boon, G. Houlsby, and S. Utili. A new contact detection algorithm for three-dimensional non-spherical particles. *Powder Technology*, 248:94 – 102, 2013. ISSN 0032-5910. doi: 10.1016/j.powtec.2012.12.040. Discrete Element Modelling.

C. Boon, G. Houlsby, and S. Utili. A new rock slicing method based on linear programming. *Computers and Geotechnics*, 65:12 – 29, 2015a. ISSN 0266-352X. doi: 10.1016/j.compgeo.2014.11.007.

C. Boon, G. Houlsby, and S. Utili. New insights into the 1963 vajont slide using 2D and 3D distinct-element method analyses. *Geotechnique*, 64(10):800–816, 2015b.

G. Bradshaw and C. O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics (TOG)*, 23(1):1–26, 2004.

British Standards Institution. Particle size analysis – Image analysis methods – Part 2: Dynamic image analysis methods. Standard BS ISO 13322-2:2006, 2006.

British Standards Institution. Representation of results of particle size analysis – Part 6: Descriptive and quantitative representation of particle shape and morphology. Standard BS ISO 9276-6:2008, 2008.

British Standards Institution. Tests for geometrical properties of aggregates – Part 3: Determination of particle shape – Flakiness index. Standard BS EN 933-3:2012, 2012.

British Standards Institution. Particle size analysis – Image analysis methods – Part 1: Static image analysis methods. Standard BS ISO 13322-1:2014, 2014.

G. Buscarnera and I. Einav. The mechanics of brittle granular materials with co-evolving grain size and shape. *Proceedings of the Royal Society A*, 477(2249): 20201005, 2021.

R. A. Caulk, J. Kozicki, D. Kunhappan, R. Maurin, E. P. Montellà, T. Sweijen, C. Yuan, and B. Chareyre. Yade's (undeniable) transformation into a model project of optimization, multi-physics couplings, and user support. In *8th International Conference on Discrete Element Methods*. University of Twente, 2019.

R. Caulkin, X. Jia, C. Xu, M. Fairweather, R. A. Williams, H. Stitt, M. Nijemeisland, S. Aferka, M. Crine, A. Léonard, et al. Simulations of structures in packed columns and validation by x-ray tomography. *Industrial & Engineering Chemistry Research*, 48(1):202–213, 2009.

R. Caulkin, W. Tian, M. Pasha, A. Hassanpour, and X. Jia. Impact of shape representation schemes used in discrete element modelling of particle packing. *Computers & Chemical Engineering*, 76:160–169, 2015.

J. A. Champion, Y. K. Katare, and S. Mitragotri. Particle shape: a new design parameter for micro-and nanoscale drug delivery carriers. *Journal of controlled release*, 121(1-2):3–9, 2007.

G.-C. Cho, J. Dodds, and J. C. Santamarina. Particle shape effects on packing density, stiffness, and strength: Natural and crushed sands. *Journal of Geotechnical and Geoenvironmental Engineering*, 132(5):591–602, 2006. doi: 10.1061/(ASCE)1090-0241(2006)132:5(591).

N. Cho, C. Martin, and D. Sego. A clumped particle model for rock. *International journal of rock mechanics and mining sciences*, 44(7):997–1010, 2007.

P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136. Salerno, Italy, 2008.

C. Clayton, C. Abbireddy, and R. Schiebel. A method of estimating the form of coarse particulates. *Geotechnique*, 59(6):493–501, 2009.

P. A. Cundall. Formulation of a three-dimensional distinct element model — Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks. In *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, volume 25(3), pages 107–116. Pergamon, 1988.

P. A. Cundall and O. D. Strack. A discrete numerical model for granular assemblies. *geotechnique*, 29(1):47–65, 1979.

G. Domokos, F. Kun, A. A. Sipos, and T. Szabó. Universality of fragment shapes. *Scientific reports*, 5:9147, 2015.

C. Dong and G. Wang. Curvatures estimation on triangular mesh. *Journal of Zhejiang University-Science A*, 6(1):128–136, 2005.

F. Dubois and M. Jean. LMGC90 une plateforme de développement dédiée à la modélisation des problèmes d'interaction. In *Actes du sixieme colloque national en calcul des structures*, volume 1, pages 111–118, 2003.

J. Duriez and S. Bonelli. Precision and computational costs of Level Set-Discrete Element Method (LS-DEM) with respect to DEM. *Computers and Geotechnics*, 134:104033, 2021.

J. Duriez and C. Galusinski. A Level Set-Discrete Element Method in YADE for numerical, micro-scale, geomechanics with refined grain shapes. *Computers & Geosciences*, 157:104936, 2021.

J. Eliáš. Simulation of railway ballast using crushable polyhedral particles. *Powder Technology*, 264:458–465, 09 2014. doi: 10.1016/j.powtec.2014.05.052.

Q. Fang and D. A. Boas. Tetrahedral mesh generation from volumetric binary and grayscale images. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1142–1145. Ieee, 2009.

A. Farsi, J. Xiang, J.-P. Latham, M. Carlsson, H. Stitt, and M. Marigo. Packing simulations of complex-shaped rigid particles using FDEM: An application to catalyst pellets. *Powder Technology*, 380:443–461, 2021.

J. Favier, M. Abbaspour-Fard, M. Kremmer, and A. Raji. Shape representation of axi-symmetrical, non-spherical particles in discrete element simulation using multi-element model particles. *Engineering computations*, 1999.

Y. Feng and Y. Tan. The minkowski overlap and the energy conserving contact model for discrete element modelling of convex non-spherical particles. *International Journal for Numerical Methods in Engineering*, 2021.

Y. Feng, K. Han, and D. Owen. Energy-conserving contact interaction models for arbitrarily shaped discrete elements. *Computer Methods in Applied Mechanics and Engineering*, 205:169–177, 2012.

J.-F. Ferellec and G. R. McDowell. A method to model realistic particle shape and inertia in DEM. *Granular Matter*, 12(5):459–467, 2010.

J. Fonseca. *The evolution of morphology and fabric of a sand during shearing*. PhD thesis, Imperial College London, UK, 2011.

J. Fonseca, C. O'Sullivan, M. Coop, and P. Lee. Non-invasive characterization of particle morphology of natural sands. *Soils and Foundations*, 52(4):712 – 722, 2012. ISSN 0038-0806. doi: 10.1016/j.sandf.2012.07.011.

J. F. Gamble, M. Tobyn, and R. Hamey. Application of image-based particle size and shape characterization systems in the development of small molecule pharmaceuticals. *Journal of Pharmaceutical Sciences*, 104(5):1563–1574, 2015.

R. Gao, X. Du, Y. Zeng, Y. Li, and J. Yan. A new method to simulate irregular particles by discrete element method. *Journal of Rock Mechanics and Geotechnical Engineering*, 4(3):276–281, 2012.

X. Garcia, J. P. Latham, J. S. Xiang, and J. Harrison. A clustered overlapping sphere algorithm to represent real particles in discrete element modelling. *Geotechnique*, 59(9):779–784, 2009.

M. Gardner, J. Kolb, and N. Sitar. Parallel and scalable block system generation. *Computers and Geotechnics*, 89:168–178, 2017.

D. Geldart, E. Abdullah, A. Hassanpour, L. Nwoke, and I. Wouters. Characterization of powder flowability using measurement of angle of repose. *China Particuology*, 4(3-4):104–107, 2006.

L. Giaccari. Surface reconstruction from scattered points cloud – MATLAB Central File Exchange, 2020. URL `https://www.mathworks.com/matlabcentral/fileexchange/63730-surface-reconstruction-from-scattered-points-cloud`. [Retrieved July 26, 2020].

N. Govender, D. N. Wilke, and S. Kok. Blaze-DEMGPU: Modular high performance DEM framework for the gpu architecture. *SoftwareX*, 5:62–66, 2016.

S. Haeri. Optimisation of blade type spreaders for powder bed preparation in additive manufacturing using DEM simulations. *Powder Technology*, 321:94–104, 2017.

K. J. Hanley, C. O'Sullivan, and X. Huang. Particle-scale mechanics of sand crushing in compression and shearing using DEM. *Soils and Foundations*, 55(5):1100–1112, 2015.

C. Hare, T. Bonakdar, M. Ghadiri, and J. Strong. Impact breakage of pharmaceutical tablets. *International journal of pharmaceutics*, 536(1):370–376, 2018.

J. Harkness. Potential particles for the modelling of interlocking media in three dimensions. *International journal for numerical methods in engineering*, 80(12): 1573–1594, 2009.

J. Harkness and A. Zervos. Some effects of particle shape on the mechanical behaviour of granular materials. In *8th International Conference on Discrete Element Methods*. University of Twente, 2019.

R. Hart, P. Cundall, and J. Lemos. Formulation of a three-dimensional distinct element model — Part II. Mechanical calculations for motion and interaction of a system composed of many polyhedral blocks. In *International journal of rock mechanics and mining sciences & Geomechanics abstracts*, volume 25(3), pages 117–125. Elsevier, 1988.

Y. He, A. Hassanpour, M. A. Behjani, and A. E. Bayly. A novel stiffness scaling methodology for discrete element modelling of cohesive fine powders. *Applied Mathematical Modelling*, 90:817–844, 2021.

H. Hertz. Über die berührung fester elastischer körper. *Journal für die reine und angewandte Mathematik*, 92(156-171):22, 1882.

C. S. Hodges, Y. Ding, and S. Biggs. The influence of nanoparticle shape on the drying of colloidal suspensions. *Journal of colloid and interface science*, 352(1): 99–106, 2010.

G. Houlsby. Potential particles: a method for modelling non-circular particles in DEM. *Computers and Geotechnics*, 36(6):953 – 959, 2009. ISSN 0266-352X. doi: 10.1016/j.compgeo.2009.03.001.

X. Huang. *Exploring critical-state behaviour using DEM.* PhD thesis, Imperial College London London, 2014.

X. Huang, K. J. Hanley, C. O'Sullivan, and C. Y. Kwok. Exploring the influence of interparticle friction on critical state behaviour using DEM. *International Journal for Numerical and Analytical Methods in Geomechanics*, 38(12):1276–1297, 2014.

W. K. Illenberger. Pebble shape (and size!). *Journal of Sedimentary Research*, 61 (5):756–767, 1991.

Itasca Consulting Group, Inc. 3DEC - Three-dimensional Distinct Element Code, Ver. 5.2. Minneapolis: Itasca, 2016.

Itasca Consulting Group, Inc. PFC – Particle Flow Code, Ver. 6.0. Minneapolis: Itasca, 2018.

T. Itoh and Y. Wanibe. Particle shape distribution and particle size–shape dispersion diagram. *Powder Metallurgy*, 34(2):126–134, 1991.

K. Iwashita and M. Oda. *Mechanics of granular materials: an introduction.* CRC press, 1999.

R. P. Jensen, P. J. Bosscher, M. E. Plesha, and T. B. Edil. DEM simulation of granular media—structure interface: effects of surface roughness and particle shape. *International Journal for Numerical and Analytical Methods in Geomechanics*, 23 (6):531–547, 1999.

X. Jia and R. A. Williams. A packing algorithm for particles of arbitrary shapes. *Powder technology*, 120(3):175–186, 2001.

X. Jia, M. Gan, R. A. Williams, and D. Rhodes. Validation of a digital packing algorithm in predicting powder packing densities. *Powder Technology*, 174(1-2): 10–13, 2007.

M. Jiang, Z. Yang, D. Barreto, and Y. Xie. The influence of particle-size distribution on critical state behavior of spherical and non-spherical particle assemblies. *Granular Matter*, 20(4):1–15, 2018.

K. L. Johnson. *Contact mechanics.* Cambridge University Press, 1987.

H. P. Jostad, H. Khoa, K. Karapiperis, and J. Andrade. Can LS-DEM be used to simulate cyclic behavior of sand? In *International Conference of the International Association for Computer Methods and Advances in Geomechanics*, pages 228–235. Springer, 2021.

J. Katagiri. A novel way to determine number of spheres in clump-type particle-shape approximation in discrete-element modelling. *Géotechnique*, 69(7):620–626, 2019.

R. Kawamoto, E. Andò, G. Viggiani, and J. E. Andrade. All you need is shape: Predicting shear banding in sand with LS-DEM. *Journal of the Mechanics and Physics of Solids*, 111:375–392, 2018.

S. P. K. Kodicherla, G. Gong, L. Fan, S. Wilkinson, and C. K. Moy. Investigations of the effects of particle morphology on granular material behaviors using a multi-sphere approach. *Journal of Rock Mechanics and Geotechnical Engineering*, 12 (6):1301–1312, 2020.

D. Kong and J. Fonseca. Quantification of the morphology of shelly carbonate sands using 3D images. *Géotechnique*, 68(3):249–261, 2018. doi: 10.1680/jgeot.16.P.278.

J. Korsawe. Minimal bounding box – MATLAB Central File Exchange, 2020. URL `https://www.mathworks.com/matlabcentral/fileexchange/18264-minimal-bounding-box`. [Retrieved July 23, 2020].

J. Kozicki and F. V. Donze. A new open-source software developed for numerical simulations using discrete modeling methods. *Computer Methods in Applied Mechanics and Engineering*, 197(49-50):4429–4443, 2008.

W. C. Krumbein. Measurement and geological significance of shape and roundness of sedimentary particles. *Journal of Sedimentary Research*, 11(2):64–72, 1941.

L. Le Pen, W. Powrie, A. Zervos, S. Ahmed, and S. Aingaran. Dependence of shape on particle size for a crushed rock railway ballast. *Granular Matter*, 15(6): 849–861, 2013.

C.-Q. Li, W.-J. Xu, and Q.-S. Meng. Multi-sphere approximation of real particles for DEM simulation based on a modified greedy heuristic algorithm. *Powder Technology*, 286:478–487, 2015.

Y. Li, M. Otsubo, R. Kuwano, and S. Nadimi. Quantitative evaluation of surface roughness for granular materials using gaussian filter method. *Powder Technology*, 2021.

J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra and its applications. *Computer Aided Geometric Design*, 25(7):503–522, 2008.

D. Liu, C. O'Sullivan, and J. A. H. Carraro. The influence of particle size distribution on the stress distribution in granular materials. *Géotechnique*, pages 1–37, 2021. doi: 10.1680/jgeot.21.00127.

J. L. Lubkin. The torsion of elastic spheres in contact. *Journal of Applied Mechanics*, 1951.

D. Markauskas, R. Kačianauskas, A. Džiugys, and R. Navakas. Investigation of adequacy of multi-sphere approximation of elliptical particles for DEM simulations. *Granular Matter*, 12(1):107–123, 2010.

V. Marzulli, C. Sandeep, K. Senetakis, F. Cafaro, and T. Pöschel. Scale and water effects on the friction angles of two granular soils with different roughness. *Powder Technology*, 377:813–826, 2021.

T. Matsushima and H. Saomoto. Discrete element modeling for irregularly-shaped sand grains. In *NUMGE 2002. 5th European Conference Numerical Methods in Geotechnical Engineering*, pages 239–246, 2002.

H.-G. Matuttis and J. Chen. *Understanding the discrete element method: simulation of non-spherical particles for granular and multi-body systems*. John Wiley & Sons, 2014.

M. Mehrabi, A. Hassanpour, and A. Bayly. An x-ray microtomography study of particle morphology and the packing behaviour of metal powders during filling, compaction and ball indentation processes. *Powder Technology*, 385:250–263, 2021.

P. Micó. stlTools – MATLAB Central File Exchange, 2020. URL `https://www.mathworks.com/matlabcentral/fileexchange/51200-stltools`. [Retrieved July 23, 2020].

R. D. Mindlin. Compliance of elastic bodies in contact. *Journal of Applied Mechanics*, 16(3):259–268, 1949.

R. D. Mindlin and H. Deresiewicz. Elastic spheres in contact under varying oblique forces. *Journal of Applied Mechanics*, 1953.

N. Minh and Y. Cheng. A DEM investigation of the effect of particle-size distribution on one-dimensional compression. *Géotechnique*, 63(1):44–53, 2013.

C. Modenese. *Numerical study of the mechanical properties of lunar soil by the discrete element method.* PhD thesis, Oxford University, UK, 2013.

G. Mollon, V. Richefeu, P. Villard, and D. Daudon. Discrete modelling of rock avalanches: sensitivity to block and slope geometries. *Granular Matter*, 17(5): 645–666, 2015.

T. Morimoto, M. Otsubo, and J. Koseki. Microscopic investigation into liquefaction resistance of pre-sheared sand: Effects of particle shape and initial anisotropy. *Soils and Foundations*, 61(2):335–351, 2021.

S. Nadimi and J. Fonseca. Enhancing soil sample preparation by thermal cycling. *Géotechnique*, 66(11):953–958, 2016.

S. Nadimi and J. Fonseca. Single-grain virtualization for contact behavior analysis on sand. *Journal of Geotechnical and Geoenvironmental Engineering*, 143(9): 06017010, 2017a.

S. Nadimi and J. Fonseca. On the torsional loading of elastoplastic spheres in contact. In *EPJ Web of Conferences*, volume 140, page 05001. EDP Sciences, 2017b.

S. Nadimi and J. Fonseca. A micro finite-element model for soil behaviour: numerical validation. *Géotechnique*, 68(4):364–369, 2018.

S. Nadimi, T. Shire, and J. Fonseca. Comparison between a μFE model and DEM for an assembly of spheres under triaxial compression. In *EPJ Web of Conferences*, volume 140, page 15002. EDP Sciences, 2017.

S. Nadimi, M. Otsubo, J. Fonseca, and C. O'Sullivan. Numerical modelling of rough particle contacts subject to normal and tangential loading. *Granular Matter*, 21 (4):108, 2019.

S. Nadimi, J. Fonseca, E. Andò, and G. Viggiani. A micro finite-element model for soil behaviour: experimental evaluation for sand under triaxial compression. *Géotechnique*, 70(10):931–936, 2020.

W. Nan, M. Ghadiri, and Y. Wang. Analysis of powder rheometry of ft4: Effect of particle shape. *Chemical Engineering Science*, 173:374–383, 2017.

W. Nan, M. Pasha, T. Bonakdar, A. Lopez, U. Zafar, S. Nadimi, and M. Ghadiri. Jamming during particle spreading in additive manufacturing. *Powder Technology*, 338:253–262, 2018.

T.-T. Ng. Input parameters of discrete element methods. *Journal of Engineering Mechanics*, 132(7):723–729, 2006.

D. H. Nguyen, É. Azéma, F. Radjai, and P. Sornay. Effect of size polydispersity versus particle shape in dense granular media. *Physical Review E*, 90(1):012202, 2014.

T. D. Nguyen and S. J. Plimpton. Aspherical particle models for molecular dynamics simulation. *Computer Physics Communications*, 243:12–24, 2019.

T. T. Nguyen and B. Indraratna. The role of particle shape on hydraulic conductivity of granular soils captured through Kozeny–Carman approach. *Géotechnique Letters*, 10(3):398–403, 2020.

L. Orefice and J. G. Khinast. Deformable and breakable DEM particle clusters for modelling compression of plastic and brittle porous materials—model and structure properties. *Powder Technology*, 368:90–104, 2020.

Á. Orosz, V. Angelidakis, and K. Bagi. Surface orientation tensor to predict preferred contact orientation and characterise the form of individual particles. *Powder Technology*, 394:312–325, 2021. ISSN 0032-5910. doi: https://doi.org/10.1016/j.powtec.2021.08.054. URL `https://www.sciencedirect.com/science/article/pii/S0032591021007415`.

M. Otsubo, C. O'Sullivan, K. J. Hanley, and W. W. Sim. The influence of particle surface roughness on elastic stiffness and dynamic response. *Géotechnique*, 67(5):452–459, 2017.

M. Otsubo, J. Liu, Y. Kawaguchi, T. T. Dutta, and R. Kuwano. Anisotropy of elastic wave velocity influenced by particle shape and fabric anisotropy under k0 condition. *Computers and Geotechnics*, 128:103775, 2020.

N. Ouhbi, C. Voivret, G. Perrin, and J.-N. Roux. 3d particle shape modelling and optimization through proper orthogonal decomposition. *Granular Matter*, 19(4): 1–14, 2017.

M. Pasha, C. Hare, M. Ghadiri, A. Gunadi, and P. M. Piccione. Effect of particle shape on flow in discrete element method simulation of a rotary batch seed coater. *Powder Technology*, 296:29–36, 2016.

Y. Petrov. Ellipsoid fit – MATLAB Central File Exchange, 2020. URL `https://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit`. [Retrieved July 23, 2020].

S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995.

A. Podlozhnyuk, S. Pirker, and C. Kloss. Efficient implementation of superquadric particles in discrete element method within an open-source framework. *Computational Particle Mechanics*, 4(1):101–118, Jan 2017. ISSN 2196-4386. doi: 10.1007/s40571-016-0131-6. URL `https://doi.org/10.1007/s40571-016-0131-6`.

M. Potticary, A. Zervos, and J. Harkness. An investigation into the effect of particle platyness on the strength of granular materials using the discrete element method. In *PARTICLES IV: proceedings of the IV International Conference on Particle-Based Methods: fundamentals and applications*, pages 767–778. CIMNE, 2015.

M. Potticary, A. Zervos, and J. Harkness. The effect of particle elongation on the strength of granular materials. In *Proceedings of the 24th UK Conference of the Association for Computational Mechanics in Engineering.* Cardiff University, 2016.

M. C. Powers. A new roundness scale for sedimentary particles. *Journal of Sedimentary Research*, 23(2):117–119, 1953.

M. Price, V. Murariu, and G. Morrison. Sphere clump generation and trajectory comparison for real particles. *Proceedings of Discrete Element Modelling 2007*, 2007.

R. Rorato, M. Arroyo, A. Gens, E. Andò, and G. Viggiani. Particle shape distribution effects on the triaxial response of sands: a DEM study. In *Micro to MACRO Mathematical Modelling in Soil Mechanics*, pages 277–286. Springer, 2018.

R. Rorato, M. Arroyo, E. Andò, and A. Gens. Sphericity measures of sand grains. *Engineering geology*, 254:43–53, 2019.

C. Sandeep, V. Marzulli, F. Cafaro, K. Senetakis, and T. Pöschel. Micromechanical behavior of dna-1a lunar regolith simulant in comparison to ottawa sand. *Journal of Geophysical Research: Solid Earth*, 124(8):8077–8100, 2019.

J. Santamarina and G.-C. Cho. Soil behaviour: The role of particle shape. In *Advances in geotechnical engineering: The Skempton conference: Proceedings of a three day conference on advances in geotechnical engineering, organised by the Institution of Civil Engineers and held at the Royal Geographical Society, London, UK, on 29–31 March 2004*, pages 604–617. Thomas Telford Publishing, 2004.

A. N. Schofield and P. Wroth. *Critical state soil mechanics*, volume 310. McGraw-hill London, 1968.

T. Schwager and T. Pöschel. Coefficient of restitution and linear–dashpot model revisited. *Granular Matter*, 9(6):465–469, 2007.

A. Semechko. Exact minimum bounding spheres and circles. GitHub, 2020. URL `https://www.github.com/AntonSemechko/Bounding-Spheres-And-Circles`. [Retrieved July 23, 2020].

A. Semechko. Rigid body parameters of closed surface meshes. GitHub, 2021. URL `https://github.com/AntonSemechko/Rigid-Body-Parameters`. [Retrieved January 6, 2021].

U. V. Shah, V. Karde, C. Ghoroi, and J. Y. Heng. Influence of particle properties on powder bulk behaviour and processability. *International journal of pharmaceutics*, 518(1-2):138–154, 2017.

H. Si. Tetgen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):1–36, 2015.

I. Smalley. The expected shapes of blocks and grains. *Journal of Sedimentary Research*, 36(2):626–629, 1966.

V. Smilauer, V. Angelidakis, E. Catalano, R. Caulk, B. Chareyre, W. Chèvremont, S. Dorofeenko, J. Duriez, N. Dyck, J. Elias, B. Er, A. Eulitz, A. Gladky,

N. Guo, C. Jakob, F. Kneib, J. Kozicki, D. Marzougui, R. Maurin, C. Modenese, G. Pekmezi, L. Scholtès, L. Sibille, J. Stransky, T. Sweijen, K. Thoeni, and C. Yuan. *Yade documentation.* The Yade Project, Nov. 2021. doi: 10.5281/zenodo.5705394. URL `https://doi.org/10.5281/zenodo.5705394`.

E. D. Sneed and R. L. Folk. Pebbles in the lower colorado river, texas a study in particle morphogenesis. *The Journal of Geology*, 66(2):114–150, 1958.

B. Soltanbeigi, A. Podlozhnyuk, C. Kloss, S. Pirker, J. Y. Ooi, and S.-A. Papanicolopulos. Influence of various DEM shape representation methods on packing and shearing of granular assemblies. *Granular Matter*, 23(2):1–16, 2021.

B. Suhr and K. Six. Simple particle shapes for DEM simulations of railway ballast: influence of shape descriptors on packing behaviour. *Granular matter*, 22(2):1–17, 2020.

R. Taghavi. Automatic clump generation based on mid-surface. In *Proceedings, 2nd international FLAC/DEM symposium, Melbourne*, pages 791–797, 2011.

M. R. Tamadondar and A. Rasmuson. The effect of carrier surface roughness on wall collision-induced detachment of micronized pharmaceutical particles. *AIChE Journal*, 66(1):e16771, 2020.

TC105. Round robin test of angle of repose (AOR), 2021. URL `http://geotech.civil.yamaguchi-u.ac.jp/tc105/`.

The CGAL Project. *CGAL User and Reference Manual.* CGAL Editorial Board, 5.0.2 edition, 2020. URL `https://doc.cgal.org/5.0.2/Manual/packages.html`.

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

C. Thornton, S. J. Cummins, and P. W. Cleary. An investigation of the comparative behaviour of alternative contact force models during inelastic collisions. *Powder technology*, 233:30–46, 2013.

H. Wadell. Volume, shape, and roundness of rock particles. *The Journal of Geology*, 40(5):443–451, 1932.

L. Wang, J.-Y. Park, and Y. Fu. Representation of real particles for DEM simulation using x-ray tomography. *Construction and Building Materials*, 21(2):338–346, 2007.

X. Wang, Z.-Y. Yin, D. Su, X. Wu, and J. Zhao. A novel approach of random packing generation of complex-shaped 3d particles with controllable sizes and shapes. *Acta Geotechnica*, pages 1–22, 2021.

C. K. Wentworth. A laboratory and field study of cobble abrasion. *The Journal of Geology*, 27(7):507–521, 1919.

M. Wiebicke, E. Andò, I. Herle, and G. Viggiani. On the metrology of interparticle contacts in sand from x-ray tomography images. *Measurement Science and Technology*, 28(12):124007, 2017.

A. Williams and N. Caldwell. Particle size and shape in pebble-beach sedimentation. *Marine Geology*, 82(3-4):199–215, 1988.

J. R. Williams and A. P. Pentland. Superquadrics and modal dynamics for discrete elements in interactive design. *Engineering Computations*, 1992.

J. Xiao, D. Zhang, K. Wei, and Z. Luo. Shakedown behaviors of railway ballast under cyclic loading. *Construction and building materials*, 155:1206–1214, 2017.

J. Xiao, X. Zhang, D. Zhang, L. Xue, S. Sun, J. Stránskỳ, and Y. Wang. Morphological reconstruction method of irregular shaped ballast particles and application in numerical simulation of ballasted track. *Transportation Geotechnics*, 24:100374, 2020.

H. Yang, B. A. Baudet, and T. Yao. Characterization of the surface roughness of sand particles using an advanced fractal approach. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2194):20160524, 2016.

S. Zhao and J. Zhao. A poly-superellipsoid-based approach on particle morphology for DEM modeling of granular media. *International Journal for Numerical and Analytical Methods in Geomechanics*, 43(13):2147–2169, 2019.

S. Zhao and J. Zhao. SudoDEM: Unleashing the predictive power of the discrete element method on simulation for non-spherical granular particles. *Computer Physics Communications*, 259:107670, 2021.

S. Zhao, T. Evans, and X. Zhou. Effects of curvature-related DEM contact model on the macro-and micro-mechanical behaviours of granular soils. *Géotechnique*, 68(12):1085–1098, 2018.

J. Zheng and R. D. Hryciw. Roundness and sphericity of soil particles in assemblies by computational geometry. *Journal of Computing in Civil Engineering*, 30(6): 04016021, 2016.

Q. Zheng, Z. Zhou, and A. Yu. Contact forces between viscoelastic ellipsoidal particles. *Powder technology*, 248:25–33, 2013.

B. Zhou, J. Wang, and B. Zhao. Micromorphology characterization and reconstruction of sand particles using micro x-ray tomography and spherical harmonics. *Engineering geology*, 184:126–137, 2015.

T. Zingg. *Beitrag zur schotteranalyse.* PhD thesis, ETH Zurich, Switzerland, 1935.

Appendix A

SHAPE: Scripts and core algorithmic operations

This appendix collects some of the main scripts introducing functions and classes of the SHape Analyser for Particle Engineering (SHAPE). Additional information and examples can be found in Angelidakis et al. (2021b) and in the code repository: `www.github.com/vsangelidakis/SHAPE`.

Figure A.1: `volume_centroid_inertiaTensor.m`

```
function [volume, centroid, current_inertia_tensor, inertia_tensor, principalOrientations] =
     volume_centroid_inertiaTensor(nodes, elements, calculateInertia)
%% Volume, centroid, current inertia tensor and principal inertia tensor and orientations of 3-D
     tetrahedral mesh

%% INPUT
%   nodes           :   (Nv,3) Vertices of tetrahedral mesh, where Nv the number of vertices
%   elements        :   (Ne,4) Elements of tetrahedral mesh, where Ne the number of elements
%   calculateInertia:   (bool) Whether to calculate the principal inertia tensor and orientations
%% OUTPUT
%   volume          :   volume of particle (tetrahedral mesh)
%   centroid        :   centroid of particle (tetrahedral mesh)
%   current_inertia_tensor: inertia tensor of particle for its current orientation
%   inertia_tensor  :   principal inertia tensor of particle
%   principalOrientations: orientation of principal axis system of the particle, relative to the current
       coordination system

%% Calculation of "Centroid" & "Volume"
volume=0;
vx=0; vy=0; vz=0;

v=zeros(1,size(elements,1));
xcm = zeros(1,size(elements,1));
ycm = zeros(1,size(elements,1));
zcm = zeros(1,size(elements,1));

for i=1:size(elements,1)
    a=nodes(elements(i,1),:)';
    b=nodes(elements(i,2),:)';
    c=nodes(elements(i,3),:)';
    d=nodes(elements(i,4),:)';

    v(i)=abs((1/6)*det([a-d, b-d, c-d]));
    volume=volume+v(i);

    xcm(i) = mean([a(1),b(1),c(1),d(1)]);
    ycm(i) = mean([a(2),b(2),c(2),d(2)]);
    zcm(i) = mean([a(3),b(3),c(3),d(3)]);

    vx=vx+v(i)* xcm(i);
    vy=vy+v(i)* ycm(i);
    vz=vz+v(i)* zcm(i);
end

%% Particle centroid=[x,y,z]
x=vx/volume;
y=vy/volume;
z=vz/volume;
centroid=[x,y,z];

if calculateInertia==false
    inertia_tensor=zeros(3);
    principalOrientations=zeros(3);
else
    %% Centering of the particle to its centroid to calculate inertia tensor
    nodes(:,1)=nodes(:,1)-x;
    nodes(:,2)=nodes(:,2)-y;
    nodes(:,3)=nodes(:,3)-z;
%   P1=nodes;

    %% Calculation of Inertia Tensor to the Centroid of the Particle
    Ixx = zeros(1,size(elements,1));
    Iyy = zeros(1,size(elements,1));
    Izz = zeros(1,size(elements,1));
    Ixy = zeros(1,size(elements,1));
    Ixz = zeros(1,size(elements,1));
    Iyz = zeros(1,size(elements,1));

    for i=1:length(elements)
        a=nodes(elements(i,1),:)';
        b=nodes(elements(i,2),:)';
        c=nodes(elements(i,3),:)';
        d=nodes(elements(i,4),:)';

        x1=a(1); y1=a(2); z1=a(3);
        x2=b(1); y2=b(2); z2=b(3);
        x3=c(1); y3=c(2); z3=c(3);
        x4=d(1); y4=d(2); z4=d(3);

        % Inertia tensor of each tetrahedron to the centroid of the particle (Tonon, 2005)
        % a
        Ixx(i)=6*v(i)*(  y1^2 + y1*y2 + y2^2 + y1*y3 + y2*y3 +...
            + y3^2 + y1*y4 + y2*y4 + y3*y4 + y4^2 + z1^2 + z1*z2+...
            + z2^2 + z1*z3 + z2*z3 + z3^2 + z1*z4 + z2*z4 + z3*z4 + z4^2)/60;
        % b
        Iyy(i)=6*v(i)*(  x1^2 + x1*x2 + x2^2 + x1*x3 + x2*x3 +...
            + x3^2 + x1*x4 + x2*x4 + x3*x4 + x4^2 + z1^2 + z1*z2+...
            + z2^2 + z1*z3 + z2*z3 + z3^2 + z1*z4 + z2*z4 + z3*z4 + z4^2)/60;
        % c
        Izz(i)=6*v(i)*(  x1^2 + x1*x2 + x2^2 + x1*x3 + x2*x3 +...
            + x3^2 + x1*x4 + x2*x4 + x3*x4 + x4^2 + y1^2 + y1*y2 +...
            + y2^2 + y1*y3 + y2*y3 + y3^2 + y1*y4 + y2*y4 + y3*y4 + y4^2)/60;
        % a'
        Iyz(i)=6*v(i)*(  2*y1*z1 + y2*z1 + y3*z1 + y4*z1 + y1*z2 +...
            + 2*y2*z2 + y3*z2 + y4*z2 + y1*z3 + y2*z3 + 2*y3*z3+...
            +   y4*z3 + y1*z4 + y2*z4 + y3*z4 + 2*y4*z4)/120;
        % c'
        Ixy(i)=6*v(i)*(  2*x1*y1 + x2*y1 + x3*y1 + x4*y1 + x1*y2 +...
            + 2*x2*y2 + x3*y2 + x4*y2 + x1*y3 + x2*y3 + 2*x3*y3+...
            +   x4*y3 + x1*y4 + x2*y4 + x3*y4 + 2*x4*y4)/120;
        % b'
        Ixz(i)=6*v(i)*(  2*x1*z1 + x2*z1 + x3*z1 + x4*z1 + x1*z2 +...
            + 2*x2*z2 + x3*z2 + x4*z2 + x1*z3 + x2*z3 + 2*x3*z3+...
            +   x4*z3 + x1*z4 + x2*z4 + x3*z4 + 2*x4*z4)/120;
        %                      clear a b c d
    end

    %Inertia tensor of the particle superposing the inertia tensors of each tetrahedron
    Ixx_final=0; for i=1:length(elements), Ixx_final=Ixx_final+Ixx(i); end
    Iyy_final=0; for i=1:length(elements), Iyy_final=Iyy_final+Iyy(i); end
    Izz_final=0; for i=1:length(elements), Izz_final=Izz_final+Izz(i); end

    Iyz_final=0; for i=1:length(elements), Iyz_final=Iyz_final+Iyz(i); end
    Ixy_final=0; for i=1:length(elements), Ixy_final=Ixy_final+Ixy(i); end
    Ixz_final=0; for i=1:length(elements), Ixz_final=Ixz_final+Ixz(i); end

    current_inertia_tensor=[  Ixx_final -Ixy_final -Ixz_final
                             -Ixy_final  Iyy_final -Iyz_final
                             -Ixz_final -Iyz_final  Izz_final];

    %% Principal Inertia Tensor & New axis system
    % when the principal inertia tensor has been already calculated.
    [principalOrientations,inertia_tensor]=eig(current_inertia_tensor); % Principal Moments of Inertia:
         Eigenvalues
    %      P1=P1*DirP; % Rotation of the vertices to the principal orientations: Eigenvectors.

end
```

Figure A.2: `surface_area.m`

```
function [surfaceArea] = surface_area(nodes, faces)
%% INPUT
%   nodes : Nodes of surface mesh (Np,3), Np the number of nodes
%   faces : Faces of surface mesh (Nf,3), Nf the number of faces
%% OUTPUT
%   surfaceArea :   Surface area of 3-D triangular (surface) mesh

    surfaceArea=0;
    for i=1:size(faces,1)

        a=nodes(faces(i,1),:)';
        b=nodes(faces(i,2),:)';
        c=nodes(faces(i,3),:)';

        n1=b-a; n2=c-a;

        areaTr=0.5*norm(cross(n1',n2'));
        surfaceArea=surfaceArea+areaTr;
    end
end
```

Figure A.3: `Sphericity_Wadell.m`

```
function [sphericity] = Sphericity_Wadell(Volume, Surface_area)
%% Degree of true sphericity proposed by Wadell (1932)
%   Volume: Volume of tetrahedral mesh
%   Surface_area: Surface (outer) area of tetrahedral mesh
    sphericity = 6*Volume/((6*Volume/pi)^(1/3)*Surface_area);
```

```
end
```

Figure A.4: `Sphericity_Krumbein.m`

```
function [sphericity] = Sphericity_Krumbein(c,b,a)
%% Intercept sphericity proposed by Krumbein (1941)
%  c,b,a: Short, Intermediate and Long dimension of a particle (aka S,I,L)
    sphericity = (b*c/a^2)^(1/3);
end
```

Figure A.5: `Convexity.m`

```
function [convexity] = Convexity(Volume, Volume_CH)
% Function to calculate the Convexity index
%% INPUT
%   Volume      : Volume of the particle
%   Volume_CH   : Volume of convex hull of the particle
%% OUTPUT
%   convexity: Convexity index (0,1]
    convexity=Volume/Volume_CH;
end
```

Figure A.6: `Form_parameters.m`

```
function [c_over_b, b_over_a] = Form_parameters_Zingg(c,b,a)
%% Form parameters proposed by Zingg (1935)
%  c,b,a: Short, Intermediate and Long dimension of a particle (aka S,I,L)
    c_over_b=c/b;
    b_over_a=b/a;
end


function [flatness, elongation] = Form_parameters_Kong_and_Fonseca(c,b,a)
%% Form parameters proposed by Kong and Fonseca (2018)
%  c,b,a: Short, Intermediate and Long dimension of a particle (aka S,I,L)
    flatness   = (b-c)/b;
    elongation = (a-b)/a;
end


function [flatness, elongation] = Form_parameters_Potticary_et_al(c,b,a)
%% Form parameters proposed by Potticary et al (2015)
%  c,b,a: Short, Intermediate and Long dimension of a particle (aka S,I,L)
    flatness     = 2*(b-c)/(a+b+c);
    elongation   =   (a-b)/(a+b+c);
end
```

Figure A.7: `SurfaceOrientationTensor.m`

```
function [C, F, R, eigenValues, eigenVectors] = SurfaceOrientationTensor(nodes, faces)
% Surface orientation tensor as introduced by Bagi & Orosz (2020)
%% INPUT
```

```
%   nodes: Nodes of surface mesh (Np,3), Np the number of nodes
%   faces: Faces of surface mesh (Nf,3), Nf the number of faces

%% OUTPUT
% C: Compactness (or equancy)
% F: Flakiness (or flatness or platyness)
% R: Rodness (or elongation)

facesNo=size(faces,1); % Number of faces

n=zeros(facesNo,3);
Area=zeros(facesNo,1);
for i=1:size(faces,1)
    A=nodes(faces(i,1),:);
    B=nodes(faces(i,2),:);
    C=nodes(faces(i,3),:);

    e_AB=B-A;
    e_BC=C-B;

    v=cross(e_AB,e_BC);  % normal vector of triangle i
    Area(i)=0.5*norm(v); % surface area of triangle i
    n(i,1:3)=v/norm(v);  % normal vector of triangle i (normalised)
end

%% Surface orientation tensor
f=zeros(3);
for k=1:facesNo
    f = f + Area(k)*(n(k,:)'*n(k,:)); % Outer product
end
f=f/sum(Area); % Normalise to the total surface area

%% Eigenvalues & Shape indices
[vectors,eigen]=eig(f,'vector');
[eigenValues,index]=sort(eigen,'descend'); % Sort eigenvalues in descending order

f1=eigenValues(1); % Largest eigenvalue
f2=eigenValues(2); % Intermediate eigenvalue
f3=eigenValues(3); % Smallest eigenvalue

eigenVectors=[vectors(:,index(1)),vectors(:,index(2)),vectors(:,index(3))]; % Sort eigenVectors

%% Compactness - Flakiness - Rodness
C =      f3/f1; % Compactness
F = (f1-f2)/f1; % Flakiness
R = (f2-f3)/f1; % Rodness
end
```

Figure A.8: `Roughness_parameters.m`

```
function [Sa] = Sa(Z)
%% INPUT
%   Z   :   (MxN) Elevation of rough surface points given on an M x N grid
```

```
%% OUTPUT
%   Sa  :   Arithmetical mean height of rough surface
[i,j]=size(Z);
Zm=1./(i*j)*sum(Z(:));
Sa=1./(i*j)*sum(sum(abs(Z-Zm)));
end

function [Sq] = Sq(Z)
%% INPUT
%   Z   :   (MxN) - Elevation of rough surface points given on an M x N grid
%% OUTPUT
%   Sq  :   Root mean square of rough surface height
[i,j]=size(Z);
Zm=1./(i*j)*sum(Z(:));
Sq=(1./(i*j)*sum(sum((Z-Zm).^2)))^0.5;
end

function [St,Sp,Sv] = St(Z)
%% INPUT
%   Z   :   (MxN) Elevation of rough surface points given on an M x N grid
%% OUTPUT
%   St  :   Total height of rough surface
%   Sp  :   Maximum peak height of rough surface
%   Sv  :   Maximum pit height of rough surface
Sp=max(Z(:));
Sv=min(Z(:));
St=Sp-Sv;
end

function [Sdq] = Sdq(Z,dx,dy)
%% INPUT
%   Z   :   (MxN) Elevation of rough surface points given on an M x N grid
%   dx  :   (scalar): Step size of grid along X axis (in length units)
%   dy  :   (scalar): Step size of grid along Y axis (in length units)
%% OUTPUT
%   Sdq :   Root mean square gradient of rough surface
[i,j]=size(Z);
Sdq=(1./((i-1)*(j-1))*(sum(sum((diff(Z,1,2)/dx).^2))+sum(sum(diff(Z,1,1)/dy)).^2))^0.5;
end

function [Ssk] = Ssk(Z,Sq)
%% INPUT
%   Z   :   (MxN) - Elevation of rough surface points given on an M x N grid
%   Sq  :   (scalar) Root mean square of rough surface height
%% OUTPUT
%   Ssk :   Skewness of rough surface
[i,j]=size(Z);
Zm=1./(i*j)*sum(Z(:));
Ssk=1/(Sq^3)/(i*j)*sum(sum((Z-Zm).^3));
end

function [Sku] = Sku(Z,Sq)
%% INPUT
```

```
%   Z   :   (MxN) Elevation of rough surface points given on an M x N grid
%   Sq  :   (scalar) Root mean square of rough surface height
%% OUTPUT
%   Sku :   Kurtosis of rough surface
[i,j]=size(Z);
Zm=1./(i*j)*sum(Z(:));
Sku=1/(Sq^4)/(i*j)*sum(sum((Z-Zm).^4));
end

function [Ssc,Rs] = Ssc(Z,dx,dy)
%% INPUT
%   Z   :   (MxN) Elevation of rough surface points given on an M x N grid
%   dx  :   (scalar): Step size of grid along X axis (in length units)
%   dy  :   (scalar): Step size of grid along Y axis (in length units)
%% OUTPUT
%   Ssc: Mean curvature of summits
[i,j]=size(Z);
Ssc=-0.5*(1./((i-2)*(j-2))*(sum(sum(diff(Z,2,2)/dx^2))+sum(sum(diff(Z,2,1)/dy^2))));
Rs=1/Ssc;
end
```

Figure A.9: `OrientedBoundingBox_PCA_SVD.m`

```
function [cornerpoints,rotmat,volume,surface,center,dimensions]=OBB_PCA_SVD(X)
%% Function to calculate OBB using SVD/PCA, based on: https://www.mathworks.com/matlabcentral/answers
    /405327-reparameterize-3d-points-with-respect-to-pca-vector#answer_324419?s_tid=prof_contriblnk
%% INPUT
% X: Vertices

%% OUTPUT
%  cornerpoints - (8x3) the cornerpoints of the bounding box.
%
%  rotmat - (3x3) rotation matrix for mapping of the pointcloud into a box which is axis-parallel (use
    inv(rotmat) for inverse mapping).
%
%  volume - (scalar) volume of the minimal box itself.
%
%  surface - (scalar) surface of the minimal box as found.
%
%  center - (1x3) centroid of bounding box
%
%  dimensions - (1x3) dimensions of the bounding box

% Perform PCA on X
X_ave=mean(X,1);              % centroid
dX=bsxfun(@minus,X,X_ave);    % center the vertices
C=dX'*dX;                     % covariance matrix
[U,~]=svd(C);

U(:,3)=cross(U(:,1),U(:,2)); % make sure there is no reflection

% Transformation that aligns centroid of X with the origin and its
% principal axes with Cartesian basis vectors
```

```
T1=eye(4); T1(1:3,4)=-X_ave(:);
T2=eye(4); T2(1:3,1:3)=U';
T=T2*T1;

% Apply T to X to get Y
Y=X;
Y(:,4)=1;
Y=(T*Y')';
Y(:,4)=[];

% PCA-based bounding box (for better visualization)
BBo=unit_cube;
L=max(Y)-min(Y);

V=bsxfun(@times,L,BBo.vertices);
V=bsxfun(@plus,V,min(Y));
BBo.vertices=V; % BB around Y

V(:,4)=1;
V=(T\V')';
V(:,4)=[];
BB=BBo;
BB.vertices=V; % BB around X; same as BBo but rotated and translated

v1=V(1,:); v2=V(2,:); v4=V(4,:); v5=V(5,:); % Isolate vertices used to calculate dimensions of OBB
edgeX=v2-v1; edgeY=v4-v1; edgeZ=v5-v1;
R=[edgeX'/norm(edgeX), edgeY'/norm(edgeY), edgeZ'/norm(edgeZ)]; % Each column represents a unit vector
     of the box
D=[norm(edgeX),norm(edgeY),norm(edgeZ)];

[S_obb, ind_S] = min(D); D(ind_S)=-1;
[L_obb, ind_L] = max(D);
ind_I=6 - ind_S - ind_L;
I_obb = D(ind_I);

cornerpoints=V;
rotmat=[R(:,ind_S), R(:,ind_I), R(:,ind_L)]; % Sorted unit vectors, following the order of S,I,L axes
volume=prod([S_obb, I_obb, L_obb]);
surface=2*(S_obb*I_obb+S_obb*L_obb+I_obb*L_obb);
center=mean(cornerpoints);
dimensions=[S_obb, I_obb, L_obb];
end

function fv=unit_cube
% Create axis-aligned unit cube
fv.vertices=[ 0 0 0; 1 0 0; 1 1 0; 0 1 0;
              0 0 1; 1 0 1; 1 1 1; 0 1 1 ];

fv.faces=[ 1 4 3 2; 5 6 7 8; 2 3 7 6;
           3 4 8 7; 1 5 8 4; 1 2 6 5 ];
end
```

Figure A.10: `Particle.m`

```
classdef Particle < dynamicprops
    % PARTICLE: Class containing all the information of a particle
    properties
        Original
    end

    methods
        function obj = Particle(Vertices,Faces,Voxelated_image,Texture,options)
            % Whether to display warnings
            if strcmp(options.warning,'on')
                warning on
            elseif strcmp(options.warning,'off')
                warning off
            else
                warning on
                warning('warning must be "on" or "off"')
            end

            %% Workflow for different inputs: meshes and point clouds vs voxelated images
            if isempty(Vertices)==false % Vertices are given as input
                shp=alphaShape(Vertices,inf);   % Create convex hull as an alpha-shape with radius=inf
                Volume_CH=volume(shp);          % Volume_of_convex_hull
                obj.Original=Particle_type(Vertices,Faces,[],Texture,options,Volume_CH);

                if options.useConvexHull
                    obj.addprop('Convex_hull');
                    [F, V] = boundaryFacets(shp);
                    obj.Convex_hull=Particle_type(V,F,[],[],options); % []: Voxelated_image
                end

            else % Voxelated_image is given as input
                if ~isa(Voxelated_image.img,'uint8')
                    Voxelated_image.img=uint8(Voxelated_image.img); % Transform voxelated image to uint8
                          array
                end
                obj.Original=Particle_type([],[],Voxelated_image,Texture,options,0);

                if options.useConvexHull
                    obj.addprop('Convex_hull');
                    shp=alphaShape(obj.Original.Mesh.Surface_mesh.Vertices,inf);
                    [F, V] = boundaryFacets(shp);
                    obj.Convex_hull=Particle_type(V,F,[],[],options); % []: Voxelated_image
                end
            end
        end

        %% Method to simplify particle geometry
        function Simplify(obj,options) %Simplify(obj,options)
            % %             obj=Particle;
            if ~options.useConvexHull
                Pm_ini=obj.Original.Mesh.Surface_mesh.Vertices;
                Fm_ini=obj.Original.Mesh.Surface_mesh.Faces;
            else
                Pm_ini=obj.Convex_hull.Mesh.Surface_mesh.Vertices;
                Fm_ini=obj.Convex_hull.Mesh.Surface_mesh.Faces;
            end

            for numFaces=options.Simplify.numFaces
                % Add dynamic property in the Particle class
                pSimplified=obj.addprop(['Faces_No_',num2str(numFaces)]);

                % CGAL — resample
                keepratio=numFaces/length(Fm_ini);
                [P_simplified,F_simplified]=meshresample(Pm_ini,Fm_ini,keepratio); %[node,elem]

                % Generate convex simplified particle
                shp=alphaShape(P_simplified,inf); % Convex hull;

                if options.useConvexHull
                    [F_simplified, P_simplified]=boundaryFacets(shp); % Boundary faces of convex hull
                end

                obj.(pSimplified.Name)=Particle_type(P_simplified,F_simplified,[],[],options,volume(shp)
                    );
            end
        end
    end
end
```

Figure A.11: `Auxiliary_geometries.m`

```
classdef Auxiliary_geometries
    %AUXILIARY_GEOMETRIES: Used for shape characterisation:
    %   AABB (Axis—aligned bounding box for the current orientation)
    %       Extrema (Coordinates of two extreme points)
    %       Dimensions (length of edges)
    %   OBB (Oriented bounding box)
    %       Extrema (Coordinates of two extreme points)
    %       Dimensions (length of edges)
    %       Centroid
    %       Volume
    %       Surface_area
    %       Orientation
    %   Fitted_ellipsoid (using least squares)
    %       Extrema (Coordinates of two extreme points)
    %       Dimensions (length of axes)
    %       Centroid
    %       Orientation
    %   Minimal_bounding_sphere (using Welzl's or Ritter's algorithm)
    %       Radius
    %       Centre
    %   Maximal_inscribed_sphere (using a Euclidean map)
    %       Radius
    %       Centre

    properties
        AABB
        OBB
        Fitted_ellipsoid
        Minimal_bounding_sphere
        Maximal_inscribed_sphere
    end

    methods
        function obj = Auxiliary_geometries(ms,options) %ms,geom,options
            %AUXILIARY_GEOMETRIES Constructor to calculate characteristics of bounding volumes and
                 fitted ellipsoid
            ver_surf=ms.Surface_mesh.Vertices;

            img=ms.Voxelated_image.img;

            %% AABB
            obj.AABB.Extrema=[min(ver_surf); max(ver_surf)];
            obj.AABB.Centroid=mean(obj.AABB.Extrema);

            %% OBB
            minimalOBB={'minVolume','minSurfaceArea','minSumEdges'};
            [LIA, LOC] = ismember(options.Auxiliary_Geometries.OBB.method, minimalOBB);
            if LIA
                %% Minimal OBB (minimal volume or surface area or sum of edges)
                metric={'v','s','e'}; % Cell serving as dictionary to minimalOBB variable
                [R,cornerpoints,volume,surface,~] = minboundbox(ver_surf(:,1),ver_surf(:,2),ver_surf
                     (:,3),metric{LOC},3);
                E1=cornerpoints(1,:); E2=cornerpoints(2,:); E4=cornerpoints(4,:); E5=cornerpoints(5,:);
                D(1)=norm(E1-E2); D(2)=norm(E1-E4); D(3)=norm(E1-E5);

                [S_obb, ind_S] = min(D); D(ind_S)=-1;
                [L_obb, ind_L] = max(D);
                ind_I=6 - ind_S - ind_L;
                I_obb = D(ind_I);

                obj.OBB.cornerpoints=cornerpoints;
                obj.OBB.rotmat=[R(:,ind_S), R(:,ind_I), R(:,ind_L)]; % Sorted unit vectors, following
                     the order of S, I, L axes
                obj.OBB.volume=volume;
                obj.OBB.surface=surface;
                obj.OBB.center=mean(cornerpoints);
                obj.OBB.dimensions=[S_obb, I_obb, L_obb];
                clear ind_S ind_L

            elseif strcmp(options.Auxiliary_Geometries.OBB.method,'PCA_points')
                switch options.Auxiliary_Geometries.OBB.points
                    case 'Surface_points'
                        ver=ver_surf;
                    case 'Tetrahedra_points'
                        ver=ms.Tetrahedral_mesh.Vertices;
                    case 'Voxel_points'

                        % Here I center the voxel coordinates to the centroid of the voxelated image
                        voxelData=ms.Voxelated_image.img;
                        [data(:,1),data(:,2),data(:,3)] = ind2sub(size(voxelData),find(voxelData>0));
                        tempData=data*ms.Voxelated_image.voxel_size(1);
                        ver=tempData - mean(tempData);% + geom.Centroid;
                    otherwise
                        error('options.Auxiliary_Geometries.OBB.points must be either: "Surface_points", "Tetrahedra_points" or "Voxel_points".')
                end
                [obj.OBB.cornerpoints,obj.OBB.rotmat,obj.OBB.volume,obj.OBB.surface,obj.OBB.center,obj.OBB.dimensions]=OBB_PCA_SVD(ver);
            else
                error('options.Auxiliary_Geometries.OBB.method must be either "PCA_points", "minVolume", "minSurfaceArea" or "minSumEdges"')
            end

            %% Fitted ellipsoid
            [center, radii, evecs, v, chi] = ellipsoid_fit(ver_surf,'');

            [S_eli, ind_S] = min(radii);  radii(ind_S)=-1;
            [L_eli, ind_L] = max(radii);
            ind_I=6 - ind_S - ind_L;
            I_eli = radii(ind_I);

            % Multiply the radii with 2 to get the length of the axes of the ellipsoid (double the radii).
            S_eli = S_eli * 2;
            I_eli = I_eli * 2;
            L_eli = L_eli * 2;

            rotmat=[evecs(:,ind_S), evecs(:,ind_I), evecs(:,ind_L)];

            obj.Fitted_ellipsoid.center=center';
            obj.Fitted_ellipsoid.rotmat=rotmat;
            obj.Fitted_ellipsoid.dimensions=[S_eli, I_eli, L_eli];
            obj.Fitted_ellipsoid.v=v;
            obj.Fitted_ellipsoid.chi=chi;

            %% Minimal bounding sphere
            try
                % Calculate the exact bounding sphere, using Welzl's algorithm
                [R,C,Xb]=ExactMinBoundSphere3D(ver_surf);
                obj.Minimal_bounding_sphere.Xb=Xb; % Points used to calculate the sphere
            catch
                % Calculate an approximate bounding sphere, using Ritter's algorithm
                [R,C]=ApproxMinBoundSphereND(ver_surf);
            end
                obj.Minimal_bounding_sphere.radius=R; % Circumradius
                obj.Minimal_bounding_sphere.center=C; % Center

            %% Maximal inscribed sphere
            edtImage = bwdist(~img);        % Euclidean map (Euclidean distance transformation)
            radius = max(edtImage(:)); %-1  % Inradius in voxel units
```

```
            [xCenter, yCenter, zCenter]= ind2sub(size(img),find(edtImage == radius)); % Center in voxel
                units

            % TODO: If the particle is convex, use linear programming, to specify the inradius using the
                 Chebychev center.

            %% Instead of using the first element, find the element closest to the centroid!
            xCenter=xCenter(1);
            yCenter=yCenter(1);
            zCenter=zCenter(1);

            dx=ms.Voxelated_image.voxel_size;

            xC=obj.AABB.Extrema(1,1)—dx(1)+xCenter∗dx(1); % Remap voxels of centroid to Cartesian space
            yC=obj.AABB.Extrema(1,2)—dx(2)+yCenter∗dx(2);
            zC=obj.AABB.Extrema(1,3)—dx(3)+zCenter∗dx(3);

            obj.Maximal_inscribed_sphere.radius=radius∗mean(dx); % Transform radius to Cartesian space
            obj.Maximal_inscribed_sphere.center=[xC, yC, zC];
        end
    end
end
```

Figure A.12: `Mesh.m`

```
% I can put the classes in a separate file
classdef Mesh
    % MESH: Discretised geometric representations of the particle:
    %   Surface_Mesh
    %   Tetrahedral_Mesh
    %   Voxelated_image
    %   Surface_texture

    properties
        Surface_mesh
        Tetrahedral_mesh
        Voxelated_image
        Surface_texture
    end

    methods
        function obj = Mesh(Vertices,Faces,Voxelated_image,Texture,options)
            % MESH Constructor from point cloud, mesh (surface or tetrahedral) or voxelated image

            %% Safeguard input
            if (isempty(Vertices)==false || isempty(Faces)==false) && isempty(Voxelated_image)==false %
                Check input variables: Do not allow simultaneous definition of Vertices—Faces and
                Voxelated_image
                error('Too many input arguments. Define either: "Vertices" or "Vertices" and Faces or "
                    Voxelated_image"')
            end

            if isempty(Vertices) && isempty(Faces)==false % Check input variables: Do not allow
```

```
         definition of Faces without defining Vertices (the inverse is allowed)
            error('If Faces are defined, Vertices must be defined as well.')
        end

        %% If the user provides vertices (i.e. the input is either a point cloud, a surface or a tetrahedral mesh)
        if isempty(Vertices)==false
            if nargin==1 || isempty(Faces) % Point cloud, no faces are given
                if strcmp(options.Mesh.reconstructPointCloudMethod,'Crust') % Use Crust method to reconstruct the particle surface
                    [Faces,~]=MyRobustCrust(Vertices);
                elseif strcmp(options.Mesh.reconstructPointCloudMethod,'Delaunay') % Use Delaunay triangulation to reconstruct the particle surface
                    TR = delaunayTriangulation(Vertices);
                    [Faces,Vertices] = freeBoundary(TR);
                else
                    error('options.Mesh.reconstructPointCloudMethod must be either "Crust" or "Delaunay"'); % or "alphaShape"
                end
            end

            if size(Faces,2)==3
                %% Surface mesh, vertices and faces are given
                assignin('base','ISO2MESH_TETGENOPT',[' -A -Q -q1.414a' num2str(options.Mesh.surf2mesh.maxvol)]) % tetgen settings

                if options.meshcheckrepair % Repair mesh
                    [Vertices,Faces]=stlSlimVerts(Vertices,Faces);              % stlTools: Remove duplicate vertices
                    [Vertices,Faces]=meshcheckrepair(Vertices,Faces,'meshfix'); % iso2mesh: repair a closed surface using the meshfix utility. It can remove self-intersecting elements and fill holes
                end

                [Pm,Fmtetra,~]=s2m(Vertices,Faces,1.0,options.Mesh.surf2mesh.maxvol,'tetgen',[],[]); % Transform surface mesh to tetrahedral mesh

                obj.Surface_mesh.Vertices=Vertices;
                obj.Surface_mesh.Faces=Faces;

                obj.Tetrahedral_mesh.Vertices=Pm;
                obj.Tetrahedral_mesh.Faces=Fmtetra(:,1:4);

            elseif size(Faces,2)==4
                %% Tetrahedral mesh, vertices and elements are given
                tri=triangulation(Faces,Vertices);
                [F,P] = freeBoundary(tri);

                if options.meshcheckrepair % Repair mesh
                    [P,F]=stlSlimVerts(P,F);                 % stlTools: Remove duplicate vertices
                    [P,F]=meshcheckrepair(P,F,'meshfix');   % iso2mesh: repair a closed surface using the meshfix utility. It can remove self-intersecting elements and fill holes
                    end

                    obj.Surface_mesh.Vertices=P;
                    obj.Surface_mesh.Faces=F;
                    obj.Tetrahedral_mesh.Vertices=tri.Points;
                    obj.Tetrahedral_mesh.Faces=tri.ConnectivityList;
                end

                %% Transform surface mesh to voxelated image (of a single particle)
                [imgTemp, map]=s2v(obj.Surface_mesh.Vertices,obj.Surface_mesh.Faces,options.Mesh.Voxelated_Image.div);
                imgTemp2=imfill(imgTemp); % Fill the interior of the particle
                img=zeros(size(imgTemp2)+2); % Enlarge the image by 2 voxels per direction, to ensure that outer voxels are empty (zero)
                img(2:end-1,2:end-1,2:end-1)=imgTemp2;
                clear imgTemp imgTemp2

                obj.Voxelated_image.img=img; clear img;
                obj.Voxelated_image.map=map;
                obj.Voxelated_image.voxel_size=[map(1,1) map(2,2) map(3,3)];

            elseif isempty(Vertices) && isempty(Faces) && isempty(Voxelated_image)==false % Voxelated image is given
                opt=2; %see vol2mesh function in iso2mesh
                method='cgalmesh';
                isovalues=[]; %see vol2mesh function in iso2mesh
                assignin('base','ISO2MESH_TETGENOPT',[' -A -Q -q1.414a' num2str(options.Mesh.surf2mesh.maxvol)])
                [node,elem,face]=v2m(Voxelated_image.img,isovalues,opt,options.Mesh.surf2mesh.maxvol,method);

                node=node*Voxelated_image.voxel_size(1); % Transform from voxel space to Cartesian space

                obj.Surface_mesh.Vertices=node(:,1:3);
                obj.Surface_mesh.Faces=face(:,1:3);

                obj.Tetrahedral_mesh.Vertices=node(:,1:3);
                obj.Tetrahedral_mesh.Faces=elem(:,1:4);

                obj.Voxelated_image.img=Voxelated_image.img;
                obj.Voxelated_image.voxel_size=Voxelated_image.voxel_size;
            end

            if isempty(Texture)==false
                obj.Surface_texture=Texture;
            end
        end
    end
end
```

Appendix B

CLUMP: Function scripts

This appendix collects the main functions of the Code Library to generate Multi-sphere Particles (CLUMP). Additional information and examples can be found in Angelidakis et al. (2021a) and in the code repository: `www.github.com/vsangelidakis/CLUMP`.

Figure B.1: `GenerateClump_Favier.m`

```
function [mesh, clump]=GenerateClump_Favier( inputGeom, N, varargin )
%% Implementation of the clump—generation concept proposed by Favier et al. (1999) [1]
% 2021 \copyright V. Angelidakis, S. Nadimi, M. Otsubo, S. Utili.
% [1] Favier, J.F., Fard, M.H., Kremmer, M. and Raji, A.O., 1999. Engineering Computations: Int J for
     Computer—Aided Engineering, 16(4), pp.467—480.

%% The main concept of this methodology:
% 1. We import the geometry of a particle either as a surface mesh or a
%    voxelated 3D image.
% 2. If a voxelated image is given, we transform it into a surface mesh,
%    providing its vertices and faces (vertex connectivity).
% 3. We calculate the inertial characteristics of the particle and center
%    it to its centroid and align it to its principal axes.
% 4. We create a number of N points along the longest particle axis and
%    identify the particle vertices belonging to each of the newly formed
%    (N+1) spans.
% 5. We generate a sphere centered to each of the N points. The radius of
%    each sphere is calculated based on the distances of the vertices
%    within the span of interest. The default behaviour considers the
%    minimum distance, although an optional parameter (varargin) exists
%    that takes the values 'min' (default), 'avg' and 'max'.

%% INPUT:
%   —inputGeom: Input geometry, given in one of the formats below:
%               1. Directory of .stl file (for surface meshes)
%               2. Directory of .mat file (for binary voxelated images)
%               3. Struct with fields {vertices,faces} (for surface meshes)
%               4. Struct with fields {img,voxel_size} (for binary voxelated images)
%                  where
%                   — img:          [Nx x Ny x Nz] voxelated image
%                   — voxel_size:   [1x3] voxel size in Cartesian space
%
%   — N:        Number of spheres to be generated.
%
%   — chooseDistance: Preferred method to specify the radii of the
%                     spheres, which can be either the minimum ('min'), the
%                     average ('avg') or the maximum ('max') distance of
%                     the vertices within the span of interest.
%
%   — output:   File name for output of the clump in .txt form  (optional)*.
%               If not assigned, a .txt output file is not created.
%
%   —visualise: Whether to plot the clump and mesh (optional)*.
%
% * varargin can contain the 'chooseDistance', the 'output' and the
%   'visualise' variables. They are all optional.

%% OUTPUT:
%   — mesh  :   structure containing all relevant parameters of polyhedron
%               mesh.vertices
%               mesh.faces
%               mesh.centroid
%               mesh.volume
%               mesh.inertia
%               mesh.inertiaPrincipal
%               mesh.orientationsPrincipal
%
%   — clump :   structure containing all relevant clump parameters
%               clump.positions     :   M—by—3 matrix containing the
%                                       position of each generated sphere.
%               clump.radii         :   M—by—1 vector containing the radius
%                                       of each generated sphere
%               clump.minRadius     :   Minimum generated sphere (might
%                                       differ from rmin)
%               clump.maxRadius     :   Maximum generated sphere
%
%               clump.numSpheres    :   Total number of spheres
%
%   — output :  txt file with centroids and radii, with format: [x,y,z,r]

%% EXAMPLE
%   inputGeom='ParticleGeometries/Cylinder.stl'; N=20; chooseDistance='min'; output='FA_cylinder.txt';
     visualise=true;
%   [mesh, clump]=GenerateClump_Favier( inputGeom, N, chooseDistance, output, visualise );

%% Define variables based on the type of the optional parameters (varargin)
chooseDistance='min'; % Default method to choose radius.
for i=1:length(varargin)
    switch class(varargin{i})
        %       case 'double'
        %           seed=varargin{i}; rng(seed);    % Fixed seed to achieve reproducible (random)
             results
        case 'char'
            if strcmp(varargin{i},'min') || strcmp(varargin{i},'avg') || strcmp(varargin{i},'max')
                chooseDistance=varargin{i};
            else
                output=varargin{i};
            end
        case 'logical'
            visualise=varargin{i};
        otherwise
            error('Wrong optional parameter type.')
    end
end

%% Main body of the function
%% Import Dependencies
addpath(genpath('../lib')) % Add path to dependencies (external codes)

%% Configure input particle geometry based on the variable type of inputGeom
switch class(inputGeom)
    case 'char'
        if strcmp(inputGeom(end—3:end),'.stl') % input file is .stl (surface mesh)
            [P,F,~] = stlRead(inputGeom);
        elseif strcmp(inputGeom(end—3:end),'.mat')  % input file is .mat (voxelated image)
```

```
            vox=load(inputGeom);
            temp=fieldnames(vox); temp=temp{1}; vox=vox.(temp); clear temp;
            voxel_size=vox.voxel_size;
%           vox

            opt=2; %see vol2mesh function in iso2mesh
            isovalues=[]; %see vol2mesh function in iso2mesh
            [P,F]=v2s(vox.img,isovalues,opt,'cgalmesh');
            P=P*voxel_size(1,1);
        else
            error('Not recognised inputGeom format.')
        end
    case 'struct'
        if isfield(inputGeom,'Vertices') || isfield(inputGeom,'vertices')  % input file is struct
             containing surface mesh
            try P=inputGeom.Vertices; catch, P=inputGeom.vertices; end
            try F=inputGeom.Faces;    catch, F=inputGeom.faces;   end
        elseif isfield(inputGeom,'img')  % input file is struct containing voxelated image
            voxel_size=inputGeom.voxel_size;
            opt=2; %see vol2mesh function in iso2mesh
            isovalues=[]; %see vol2mesh function in iso2mesh
            [P,F]=v2s(inputGeom.img,isovalues,opt,'cgalmesh');
            P=P*voxel_size(1,1);
        else
            error('Not recognised inputGeom format.')
        end
    case 'triangulation'
        try
            F=inputGeom.ConnectivityList;
            P=inputGeom.Points;
        catch
            error('Not recognised -triangulation- format.')
        end
    otherwise
        error('Not recognised inputGeom format.')
end

% Ensure all face normals are oriented coherently, pointing outwards
TR2=triangulation(F,P);
[TR,~]=ConsistentNormalOrientation(TR2); %numInvFaces

[RBP,~]=RigidBodyParams(TR);
% Attention: For cubic particles, with no elongation, the RigidBodyParams
% function mis-identifies the principal planes as the ones of the diagonal.
% This is the same mistake the PCA typically does for particles with three
% equal dimensions. Thankfully, the method of Favier et al (1999) is meant
% to be used for elongated and somewhat axi-symmetric particles.

% % Plot original particle
%   patch('Faces',F,'Vertices',P,'FaceColor','r') %'r'
%   axis equal; camlight

%% Center particle around its centroid and align it along its principal axes.
```

```
rot=RBP.PAI;

% Transform rotation matrix to align longest axis along X direction
temp=rot(:,1);
rot(:,1)=rot(:,3);
rot(:,3)=temp;

P=P-RBP.centroid;
P=P*rot;

% % Plot particle centered around its centroid and aligned to its principal axes
%   patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.3) %'r'
%   axis equal; camlight

X_extremas=[min(P(:,1)),max(P(:,1))];

a=X_extremas(1);
b=X_extremas(2);
nSegments=N;
%Example:
endPoints = linspace(a,b,nSegments + 1);   %4 endpoints for 3 segments
start = endPoints(1:end-1);                %3 starting points
stop = endPoints(2:end);                   %3 stopping points
midPoints = stop - ((stop(1)-start(1))/2); %3 middle points

%   scatter(endPoints,zeros(length(endPoints)),'r','filled') % endPoints
%   scatter(midPoints,zeros(length(midPoints)),'b','filled') % midPoints

minDistance=zeros(1,length(midPoints));
minDx=zeros(1,length(midPoints));
for i=1:length(midPoints) % For each midpoint
    count=1;
    % Find vertices within each sector
    for j=1:size(P,1) % for each point on the particle surface (in principal axes)
        if P(j,1)>=endPoints(i) && P(j,1)<=endPoints(i+1)
            p{i}(count,1:3)=P(j,1:3);
            count=count+1;
        end
    end
    %   scatter3(p{1,i}(:,1),p{1,i}(:,2),p{1,i}(:,3),20,i/length(midPoints)*rand(1,3),'filled'); hold on
    %20*i

    % Find closest distance of midpoint to any surface vertex
    xM=midPoints(i);   yM=0; zM=0;
    x1=endPoints(1);   %y1=0; z1=0;
    x2=endPoints(end); %y2=0; z2=0;

    % Closest distance between midpoint and particle surface
    minDistance(i)=min(sqrt( (P(:,1)-xM).^2 + (P(:,2)-yM).^2 + (P(:,3)-zM).^2 ));

    % Closest distane between midpoint and particle X limits
    minDx(i)=min( abs(x1-xM) , abs(x2-xM) );

    %   minDx(i)=min(sqrt( (x1—xM).^2 + (y1—yM).^2 + (z1—zM).^2 ),...
    %       sqrt( (x2—xM).^2 + (y2—yM).^2 + (z2—zM).^2 ) );
end

%% Build "clump" structure
clump=struct;
clump.positions=[];
clump.radii=[];

%   radius=zeros(1,midPoints);
radius=zeros(1,length(midPoints));
for i=1:length(midPoints)
    if ~isempty(p{1,i})
        distance=sqrt( (p{1,i}(:,1)—midPoints(i)).^2 + (p{1,i}(:,2)—0).^2 + (p{1,i}(:,3)—0).^2 );
        switch chooseDistance
            case 'min'
                radius(i)=min(distance);    % Minimum distance
            case 'avg'
                radius(i)=mean(distance);   % Average distance
            case 'max'
                radius(i)=max(distance);    % Maximum distance
        end
        % [min(distance),mean(distance),max(distance)]
        % Limit radius to not exceed the particle length (along X axis)
        radius(i)=min(radius(i),minDx(i));

        % The code below uses the minimum distance to the particle
        % surface, not the smallest to the particle X limits
        %       radius(i)=min(radius(i),minDistance(i));
        %       radius(i)=minDistance(i);

        clump.positions(i,:)=[midPoints(i),0,0];
        clump.radii(i,1)=radius(i);
    else
        error('The number of particle vertices is small for the requested number of spheres. Either
             remesh the particle surface or choose a smaller number of spheres.')
    end
end

%% Transform the mesh and the clump coordinates back to the initial (non—principal) system
P=P*rot'; % use either the transposed rot' or inverse inv(rot) rotation matrix to return to the initial
     coordinate system
P=P+RBP.centroid;

clump.positions=clump.positions*rot';
clump.positions=clump.positions+RBP.centroid;

%% Build "mesh" and finalise "clump" structures
mesh=struct;
mesh.vertices=P;
mesh.faces=F;
mesh.centroid=RBP.centroid;
mesh.volume=RBP.volume;
mesh.inertia=RBP.inertia_tensor;
mesh.inertiaPrincipal=RBP.eigs;
mesh.orientationsPrincipal=RBP.PAI;

[clump.minSphere.centroid, clump.minSphere.radius]=min(clump.radii);
[clump.maxSphere.centroid, clump.maxSphere.radius]=max(clump.radii);
clump.numSpheres=length(clump.radii);

%% Plot clump and mesh (optional)
if visualise
    patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.5,'EdgeColor','none');
    axis equal
    camlight
    box on; grid on; hold on
    alpha 0.4

    %% Plot spheres
    for j=1:length(clump.radii)
        [X,Y,Z]=sphere;
        xSph=X*clump.radii(j);
        ySph=Y*clump.radii(j);
        zSph=Z*clump.radii(j);

        color=rand(1,3);
        xC=clump.positions(j,1);
        yC=clump.positions(j,2);
        zC=clump.positions(j,3);

        surf(xSph+xC,ySph+yC,zSph+zC,'EdgeColor','none','FaceAlpha',0.6,'FaceColor',color)
    end
end

%% Export clump (optional)
% Output is offered in the generic format x_y_z_r. For more specialised
% formats, try the exportClump module.
if isempty(output)==false
    dlmwrite(output, [clump.positions, clump.radii], 'delimiter', ',', 'precision', '%10f')
end

end
```

Figure B.2: `GenerateClump_Ferellec_McDowell.m`

```
function [mesh, clump]=GenerateClump_Ferellec_McDowell( inputGeom, dmin, rmin, rstep, pmax, varargin )
%% Implementation of the clump—generation concept proposed by Ferellec and McDowell (2010) [1]
% 2021 \copyright V. Angelidakis, S. Nadimi, M. Otsubo, S. Utili.
% [1] Ferellec, J.F. and McDowell, G.R., 2010. Granular Matter, 12(5), pp.459—467. DOI 10.1007/s10035
    —010—0205—8

%% The main concept of this methodology:
% 1. We import the geometry of a particle either as a surface mesh or a
%    voxelated 3D image.
% 2. If a voxelated image is given, we transform it into a surface mesh,
%    providing its vertices and faces (vertex connectivity).
% 3. We calculate the normal of each vertex pointing inwards.
% 4. For a random vertex on the particle surface, we start creating
%    tangent spheres with incremental radii along the vertex normal,
%    starting from 'rmin', with a step of 'rstep', until they meet the
%    surface of the particle.
% 5. We select a new vertex randomly, which has a distance larger
%    than 'dmin' from the existing spheres and do the same.
% 6. When a percentage 'pmax' of all the vertices is used to generate
%    spheres, the generation procedure stops.
% —  An optional 'seed' parameter is introduced, to generate reproducible
%    clumps.

%% Influence of parameters
% rmin: (0,inf) Larger rmin will lead to a smaller number of spheres
% dmin: [0,inf) Larger dmin will lead to a smaller number of spheres
% pmax: (0,1]   Larger pmax will lead to a larger number of spheres

% Pros:     The authors of this methodology claim efficiency and preservation
%           of flat faces (reduced artificial roughness compared to other techniques).
% Cons:     The methodology is mesh—dependent, as spheres are generated at the
%           vertices of the input mesh.
% Warning:  The authors of this methodology advise that if the initial mesh
%           is very finely discretised, an adequately large rmin value
%           should be used, to guard the process against "parasitic
%           spheres", i.e. very small spheres which might result to
%           numerical instabilities when using DEM.

%% INPUT:
%   —inputGeom: Input geometry, given in one of the formats below:
%               1. Directory of .stl file (for surface meshes)
%               2. Directory of .mat file (for binary voxelated images)
%               3. Struct with fields {vertices,faces} (for surface meshes)
%               4. Struct with fields {img,voxel_size} (for binary voxelated images)
%                  where
%                   — img:          [Nx x Ny x Nz] voxelated image
%                   — voxel_size:   [1x3] voxel size in Cartesian space
%
%   — dmin  :   Minimum allowed distance between new vertex of the surface
%               mesh and existing spheres. If left zero, this distance is
%               not cheched.
%
```

```
%   — rmin  :   Minimum radius of sphere to be generated. For coarse
%               meshes, the actual minimum radius might be >rmin.
%
%   — rstep :   Step used to increase the radius in each iteration, until
%               the generated sphere meets another point of the particle.
%
%   — pmax  :   Percentage of vertices which will be used to generate
%               spheres. The selection of vertices is random.
%
%   — seed  :   Seed value, used to achieve reproducible (random) results (optional)*
%
%   — output:   File name for output of the clump in .txt form  (optional)*.
%               If not assigned, a .txt output file is not created.
%
%   —visualise: Whether to plot the clump and mesh (optional)*.
%
% * varargin can contain either of the optimal variables "seed", "output",
% visualise" or else: seed=varargin{1}; output=varargin{2}; visualise=varargin{3}.

%% OUTPUT:
%   — mesh  :   structure containing all relevant parameters of polyhedron
%               mesh.vertices
%               mesh.faces
%               mesh.centroid
%               mesh.volume
%               mesh.inertia
%               mesh.inertiaPrincipal
%               mesh.orientationsPrincipal
%
%   — clump :   structure containing all relevant clump parameters
%               clump.positions     :   M—by—3 matrix containing the
%                                       position of each generated sphere.
%               clump.radii         :   M—by—1 vector containing the radius
%                                       of each generated sphere
%               clump.minRadius     :   Minimum generated sphere (might
%                                       differ from rmin)
%               clump.maxRadius     :   Maximum generated sphere
%
%               clump.numSpheres    :   Total number of spheres
%
%   — output :  txt file with centroids and radii, with format: [x,y,z,r]

%% EXAMPLE
% inputGeom='Hexahedron_Fine_Mesh.stl'; dmin=0.01;  rmin=0.01; rstep=0.001; pmax=1.0;   seed=5; output='
     hexaFine.txt'; visualise=true;
% [mesh, clump]=clumpGenerator_Ferellec_McDowell( inputGeom, dmin, rmin, rstep, pmax, seed, output,
     visualise );

%% Define variables based on the type of the optional parameters (varargin)
output=[];
visualise=false;
for i=1:length(varargin)
    switch class(varargin{i})
```

```
        case 'double'
            seed=varargin{i}; rng(seed);    % Fixed seed to achieve reproducible (random) results
        case 'char'
            output=varargin{i};
        case 'logical'
            visualise=varargin{i};
        otherwise
            error('Wrong optional parameter type.')
    end
end

%% Main body of the function
%% Import Dependencies
addpath(genpath('../lib')) % Add path to dependencies (external codes)

%% Configure input particle geometry based on the variable type of inputGeom
switch class(inputGeom)
    case 'char'
        if strcmp(inputGeom(end-3:end),'.stl') % input file is .stl (surface mesh)
            [P,F,~] = stlRead(inputGeom);
        elseif strcmp(inputGeom(end-3:end),'.mat')  % input file is .mat (voxelated image)
            vox=load(inputGeom);
            temp=fieldnames(vox); temp=temp{1}; vox=vox.(temp); clear temp;
            voxel_size=vox.voxel_size;

            opt=2; %see vol2mesh function in iso2mesh
            isovalues=[]; %see vol2mesh function in iso2mesh
            [P,F]=v2s(vox.img,isovalues,opt,'cgalmesh');
            P=P*voxel_size(1,1);
        else
            error('Not recognised inputGeom format.')
        end
    case 'struct'
        if isfield(inputGeom,'Vertices') || isfield(inputGeom,'vertices')  % input file is struct
             containing surface mesh
            try P=inputGeom.Vertices; catch, P=inputGeom.vertices; end
            try F=inputGeom.Faces;    catch, F=inputGeom.faces;   end
        elseif isfield(inputGeom,'img')  % input file is struct containing voxelated image
            voxel_size=inputGeom.voxel_size;
            opt=2; %see vol2mesh function in iso2mesh
            isovalues=[]; %see vol2mesh function in iso2mesh
            [P,F]=v2s(inputGeom.img,isovalues,opt,'cgalmesh');
            P=P*voxel_size(1,1);
        else
            error('Not recognised inputGeom format.')
        end
    case 'triangulation'
        try
            F=inputGeom.ConnectivityList;
            P=inputGeom.Points;
        catch
            error('Not recognised -triangulation- format.')
        end
    otherwise
        error('Not recognised inputGeom format.')
end

% Create struct with fields faces/vertices (patch format)
FV=struct();
FV.vertices=P;

% Ensure all face normals are oriented coherently, pointing outwards
TR2=triangulation(F,P);
[TR2_fix,~]=ConsistentNormalOrientation(TR2); %numInvFaces
FV.faces=TR2_fix.ConnectivityList;

% Invert normals to point inwards, to grow tangent spheres
N=-TR2_fix.vertexNormal;

% Calculate Rigid Body Parameters (RBP): Centroid, Volume, Inertia Tensor
[RBP,~]=RigidBodyParams(FV);

% Visualise the normal vectors
% quiver3(P(:,1),P(:,2),P(:,3),N(:,1)*5,N(:,2)*5,N(:,3)*5); axis equal

% P=P-RBP.centroid; % Center the particle to its centroid.

Pmax=1:length(P);   % List of vertices indices

Vertices=Pmax(randperm(length(Pmax)));  % Shuffle indices of vertices (random selection)
% Vertices=Pmax;                         % Ordered indices of vertices (ordered selection)

tol=rmin/1000;  % Tolerance so that the starting vertex is considered outside the sphere

%% Build "mesh" structure
mesh=struct;
mesh.vertices=P;
mesh.faces=F;
mesh.centroid=RBP.centroid;
mesh.volume=RBP.volume;
mesh.inertia=RBP.inertia_tensor;
mesh.inertiaPrincipal=RBP.eigs;
mesh.orientationsPrincipal=RBP.PAI;

%% Build "clump" structure
clump=struct;
clump.positions=[];
clump.radii=[];

counter=1;
iCount=1;
for k=Pmax
    i=Vertices(iCount);
    r=rmin;
    reachedMaxRadius=false; % The maximum radius is reached when the sphere becomes large enough to
         include a point of the mesh

    x=P(i,1); % Vertex used to generate sphere
    y=P(i,2);
    z=P(i,3);

    n=N(i,:); % Vertex normal facing inwards

    %% Check if vertex is closer than dmin to the surface of one of the existing spheres
    if iCount>1 && dmin>0
        dcur=min(sqrt( (x—clump.positions(:,1)).^2 + (y—clump.positions(:,2)).^2 + (z—clump.positions
             (:,3)).^2 ) — clump.radii(:) );    % Distances of all points to the center of the sphere
        if dcur<dmin
            iCount=iCount+1;
            continue
        end
    end

    %% Alternative, non—vectorised loop for dmin check
    %       skipVertex=false;
    %       for m=1:length(clump.radii)
    %           dcur=min(sqrt( (x—clump.positions(m,1)).^2 + (y—clump.positions(m,2)).^2 + (z—clump.
        positions(m,3)).^2 ) — clump.radii(m) );
    %           if abs(dcur)<clump.radii(m) && dcur<dmin
    %               skipVertex=true;
    %           end
    %       end
    %
    %       if skipVertex
    %           iCount=iCount+1;
    %           continue
    %       end

    %   scatter3(x, y, z,'r') % Uncomment to visualise each point that is used to generate spheres

    while reachedMaxRadius==false % while the sphere has not reached the particle surface
        sphMin=1e15; % Minimum value of potential function
        while sphMin>—tol
            xC=x+r*n(1);
            yC=y+r*n(2);
            zC=z+r*n(3);

            distance=sqrt( (P(:,1)—xC).^2 + (P(:,2)—yC).^2 + (P(:,3)—zC).^2 );  % Distances of all
                 points to the center of the sphere
            sph=(distance/r).^2—1;                                               % Value of spherical
                 potential function (negative for points inside the sphere)
            sphMin=min(sph);

            r=r+rstep; % Grow radius for next step
        end
        reachedMaxRadius=true;
        indMin=find(sph==sphMin);

        pointInside=P(indMin(1),:);

        %       for k=1:length(indMin) % Uncomment to visualise the points of the mesh that are included
              in the current sphere
        %           scatter3(P(indMin(k),1), P(indMin(k),2), P(indMin(k),3),'b')
        %       end

        vAB=[pointInside(1)—x, pointInside(2)—y, pointInside(3)—z]; % Vector from starting point to
             point with min distance to the center of the sphere
        vAD=dot(vAB,n)/norm(n);                                      % Projection of previous vector on
             the current normal vector
        %       theta =atan2( vecnorm(cross(n,vAB,2),2,2) , dot(n,vAB,2) ); % Angle between vAB and
             normal vector n

        AB=norm(vAB);
        AD=norm(vAD);
        %       BD=sqrt(AB^2—AD^2);

        radius=AB^2/AD/2;

        xC=x+radius*n(1);
        yC=y+radius*n(2);
        zC=z+radius*n(3);

        clump.positions(counter,:)=[xC,yC,zC];
        clump.radii(counter,1)=radius;
        counter=counter+1;

    end
    %% Check whether the maximum percentage of vertices has been used
    pcur=length(clump.radii)/length(P); % Current percentage of vertices used
    if pcur<pmax
        iCount=iCount+1;
    else
        break
    end
end

[clump.minSphere.centroid, clump.minSphere.radius]=min(clump.radii);
[clump.maxSphere.centroid, clump.maxSphere.radius]=max(clump.radii);
clump.numSpheres=length(clump.radii);

%% Plot clump and mesh (optional)
if visualise
    %   patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.15,'EdgeColor','none','EdgeAlpha
         ',0.1) ;%[0.5,0.5,0.5]
    %   patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.1,'EdgeColor','none','EdgeAlpha
         ',0.15);
    patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.2,'EdgeColor','none','EdgeAlpha',0.4); %
         [0,0.4,0]
    axis equal
    camlight
    % hL1=camlight('headlight');
    % set(hL1,'style','infinite','position',mesh.centroid*2)
```

```matlab
    % set(gca,'visible','off')
    box on; grid on; hold on
    alpha 0.5

    %% Plot normals (they should point inwards)
    % quiver3(P(:,1),P(:,2),P(:,3),N(:,1),N(:,2),N(:,3)) % Uncomment to visualise normal vectors of
         vertices. They should all point inwards

    %% Plot spheres
    for j=1:length(clump.radii)
        [X,Y,Z]=sphere;
        xSph=X*clump.radii(j);
        ySph=Y*clump.radii(j);
        zSph=Z*clump.radii(j);

        color=rand(1,3);
        %   color='g';
        xC=clump.positions(j,1);
        yC=clump.positions(j,2);
        zC=clump.positions(j,3);

        surf(xSph+xC,ySph+yC,zSph+zC,'EdgeColor','none','FaceColor','g','FaceAlpha',1,'FaceColor',color)
    end
end

%% Export clump (optional)
% Output is offered in the generic format x_y_z_r. For more specialised
% formats, try the exportClump module.
if isempty(output)==false
    dlmwrite(output, [clump.positions, clump.radii], 'delimiter', ',', 'precision', '%10f')
end

end
```

Figure B.3: `GenerateClump_Euclidean_3D.m`

```
function [mesh, clump]=GenerateClump_Euclidean_3D( inputGeom, N, rMin, div, overlap, varargin )
%% Clump generator using the Euclidean map for voxelated, 3D particles
% 2021 \copyright V. Angelidakis, S. Nadimi, M. Otsubo, S. Utili.

%% The main concept of this methodology:
% 1. We import the geometry of a particle either as a surface mesh or a
%    voxelated 3D image.
% 2. If a mesh is given, we transform it into a voxelated representation,
%    i.e. a binary 3D image, where each voxel belonging to the particle is
%    equal to zero.
% 3. The Euclidean distance transform of the 3D image is computed and
%    the radius of the largest inscribed sphere is found as the maximum
%    value of the Euclidean transform of the voxelated image.
% 4. The voxels corresponding to the inscribed sphere are then set equal to
%    one. This methodology can also generate overlapping spheres, if only a
%    percentage of the voxels of each new sphere are set equal to one,
%    instead of all of them.
% 5. This process is repeated until a user-defined number of spheres 'N' is
%    found or until the user-defined minimum radius criterion has been met,
%    as the spheres are generated in decreasing sizes.

%% Influence of parameters
% N:        [1,inf)  Larger N will lead to a larger number of spheres
% rMin:     (0,inf)  Larger rMin will lead to a smaller number of spheres
% div:      (5,inf]  Larger div will lead to better shape resolution in voxel space
% overlap:  [0,1)    Larger overlap will lead to larger spheres overall

%% INPUT:
%   -inputGeom: Input geometry, given in one of the formats below:
%               1. Directory of .stl file (for surface meshes)
%               2. Directory of .mat file (for binary voxelated images)
%               3. Struct with fields {vertices,faces} (for surface meshes)
%               4. Struct with fields {img,voxel_size} (for binary voxelated images)
%                  where
%                   - img:          [Nx x Ny x Nz] voxelated image
%                   - voxel_size:   [1x3] voxel size in Cartesian space
%
%   - N:        Number of spheres to be generated.
%
%   - rMin:     Minimum allowed radius: When this radius is met, the
%               generation procedure stops even before N spheres are
%               generated.
%
%   - div:      Division number along the shortest edge of the axes-aligned
%               bounding box (AABB) of the particle during voxelisation,
%               i.e. during the transformation of an input surface mesh
%               into a 3D image. It controls the resolution of the
%               voxelated representation of the particle. Not used when a
%               3D image is provided directly as input. If not given,
%               div=50 (default value in iso2mesh).
%
%   - overlap:  Overlap percentage: [0,1): 0 for non-overlapping spheres,
%               0.4 for 40% overlap of radii, etc.
%
%   — output:   File name for output of the clump in .txt form (optional)∗.
%               If not assigned, a .txt output file is not created.
%
%   —visualise: Whether to plot the clump and mesh (optional)∗.
%
% ∗ varargin can contain either of the optional variables "output",
% "visualise" or else: output=varargin{1}; visualise=varargin{2}.

%% OUTPUT:
%   — mesh  :   structure containing all relevant parameters of input polyhedron
%               mesh.vertices
%               mesh.faces
%               mesh.centroid
%               mesh.volume
%               mesh.inertia
%               mesh.inertiaPrincipal
%               mesh.orientationsPrincipal
%
%   — clump :   structure containing all relevant clump parameters
%               clump.positions     :   M—by—3 matrix containing the
%                                       position of each generated sphere.
%               clump.radii         :   M—by—1 vector containing the radius
%                                       of each generated sphere
%               clump.minRadius     :   Minimum generated sphere (might
%                                       differ from rmin)
%               clump.maxRadius     :   Maximum generated sphere
%
%               clump.numSpheres    :   Total number of spheres
%
%   — output :  txt file with centroids and radii, with format: [x,y,z,r]

%% EXAMPLE
% inputGeom='Hexahedron_Coarse_Mesh.stl'; N=24; rMin=0; div=102; overlap=0.6; output='EU_octaCoarse.txt
    '; visualise=true;
% [mesh, clump]=GenerateClump_Euclidean_3D( inputGeom, N, rMin, div, overlap, output, visualise );

%% Define variables based on the type of the optional parameters (varargin)
output=[];
visualise=false;
for i=1:length(varargin)
    switch class(varargin{i})
        case 'char'
            output=varargin{i};
        case 'logical'
            visualise=varargin{i};
        otherwise
            error('Wrong optional parameter type.')
    end
end

%% Main body of the function
```

```matlab
%% Import dependencies
addpath(genpath('../lib'))  % Add path to dependencies (external codes)

%% Configure input particle geometry based on the variable type of inputGeom
switch class(inputGeom)
    case 'char'
        if strcmp(inputGeom(end-3:end),'.stl') % input file is .stl (surface mesh)
            [P,F,~] = stlRead(inputGeom);

            % Calculate Rigid Body Parameters (RBP)
            FV=struct();    FV.vertices=P;  FV.faces=F; [RBP,~]=RigidBodyParams(FV);

            % Transform surface mesh to voxelated image
            [imgTemp, map]=s2v(P,F,div);

            imgTemp2=fillholes3d(imgTemp,2); % This causes some loss of accuracy around the value of 2 voxels (uses imclose), but is needed to ensure that imfill below works properly.
            imgTemp2=imfill(imgTemp2); % Fill the interior of the particle with true values (1).
            % imgTemp2=imfill(imgTemp); % Fill the interior of the particle with true values (1).
            img=zeros(size(imgTemp2)+2); % Expand the image by 2 voxels in each direction, to ensure the boundary voxels are false (zeros).
            img(2:end-1,2:end-1,2:end-1)=imgTemp2;
            clear imgTemp imgTemp2

            % Ensure the voxel size is the same in all 3 directions -> Might be an overkill, but still
            if abs((map(1,1)-map(2,2))/map(1,1))<1e-6 || abs((map(2,2)-map(3,3))/map(2,2))<1e-6
                voxel_size=map(1,1);
            else
                warning('The affine transformation from voxels to Cartesian dimensions is not the same in all directions. Voxel size is not the same in X,Y,Z! Potentially wrong radii in Cartesian units!')
                voxel_size=map(1,1);
            end

        elseif strcmp(inputGeom(end-3:end),'.mat')  % input file is .mat (voxelated image)
            vox=load(inputGeom);
            temp=fieldnames(vox); temp=temp{1}; vox=vox.(temp); clear temp;
            img=vox.img;
            voxel_size=vox.voxel_size;

            opt=2; %see vol2mesh function in iso2mesh
            isovalues=[]; %see vol2mesh function in iso2mesh
            [P,F]=v2s(vox.img,isovalues,opt,'cgalmesh');
            P=P*voxel_size(1,1);

            % Calculate Rigid Body Parameters (RBP)
            FV=struct();    FV.vertices=P;  FV.faces=F; [RBP,~]=RigidBodyParams(FV);
        else
            error('Not recognised inputGeom format.')
        end
    case 'struct'
        if isfield(inputGeom,'Vertices') || isfield(inputGeom,'vertices')  % input file is struct containing surface mesh
```

```matlab
            try P=inputGeom.Vertices; catch, P=inputGeom.vertices; end
            try F=inputGeom.Faces;    catch, F=inputGeom.faces;   end

            % Calculate Rigid Body Parameters (RBP)
            [RBP,~]=RigidBodyParams(inputGeom);

            % Transform surface mesh to voxelated image
            [imgTemp, map]=s2v(P,F,div);

            imgTemp2=fillholes3d(imgTemp,2); % This causes some loss of accuracy around the value of 2 voxels (uses imclose), but is needed to ensure that imfill below works properly.
            imgTemp2=imfill(imgTemp2); % Fill the interior of the particle with true values (1).
            % imgTemp2=imfill(imgTemp); % Fill the interior of the particle with true values (1).
            img=zeros(size(imgTemp2)+2); % Expand the image by 2 voxels in each direction, to ensure the boundary voxels are false (zeros).
            img(2:end-1,2:end-1,2:end-1)=imgTemp2;
            clear imgTemp imgTemp2

            % Ensure the voxel size is the same in all 3 directions -> Might be an overkill, but still
            if abs((map(1,1)-map(2,2))/map(1,1))<1e-6 || abs((map(2,2)-map(3,3))/map(2,2))<1e-6
                voxel_size=map(1,1);
            else
                warning('The affine transformation from voxels to Cartesian dimensions is not the same in all directions. Voxel size is not the same in X,Y,Z! Potentially wrong radii in Cartesian units!')
                voxel_size=map(1,1);
            end

        elseif isfield(inputGeom,'img')  % input file is struct containing voxelated image
            img=inputGeom.img;
            voxel_size=inputGeom.voxel_size;

            opt=2; %see vol2mesh function in iso2mesh
            isovalues=[]; %see vol2mesh function in iso2mesh
            [P,F]=v2s(inputGeom.img,isovalues,opt,'cgalmesh');
            P=P*voxel_size(1,1);

            % Calculate Rigid Body Parameters (RBP)
            FV=struct();    FV.vertices=P;  FV.faces=F; [RBP,~]=RigidBodyParams(FV);
        else
            error('Not recognised inputGeom format.')
        end
    otherwise
        error('Not recognised inputGeom format.')
end

% figure()
% volshow(img)

% The if statement below fixes a bug in the sign of volume/inertia in the RBP code, if the surface normal vectors point inwards.
if RBP.volume<eps
    RBP.volume=-RBP.volume;
```

```
%   RBP.inertia=-RBP.inertia;
%   RBP.orientationsPrincipal=-1*RBP.orientationsPrincipal;
    disp('Correcting the sign of volume, attributed to inverted normals.') % volume and inertia
end

%% Build "mesh" structure
mesh=struct;
mesh.vertices=P;
mesh.faces=F;
mesh.centroid=RBP.centroid;
mesh.volume=RBP.volume;
mesh.inertia=RBP.inertia_tensor;
mesh.inertiaPrincipal=RBP.eigs;
mesh.orientationsPrincipal=RBP.PAI;

%% Build "clump" structure
clump=struct;
clump.positions=[];
clump.radii=[];

%% Calculate extreme coordinates & centroid of the AABB of the particle
minX=min(P(:,1)); maxX=max(P(:,1)); aveX=mean([minX,maxX]); %ave: centroid of the AABB
minY=min(P(:,2)); maxY=max(P(:,2)); aveY=mean([minY,maxY]);
minZ=min(P(:,3)); maxZ=max(P(:,3)); aveZ=mean([minZ,maxZ]);

%% Center the particle to the centroid of its AABB
P(:,1)=P(:,1)-aveX;
P(:,2)=P(:,2)-aveY;
P(:,3)=P(:,3)-aveZ;

%% Dimensions of the new image
halfSize=[size(img,2)/2, size(img,1)/2, size(img,3)/2];

[dx,dy,dz] = meshgrid(1:size(img,2), 1:size(img,1), 1:size(img,3));

%% Calculate centroid of the voxelated image
stats = regionprops3(img,'Centroid'); % 'all'
centroid=stats.Centroid; % Centroid of the initial particle

counter=1;
intersection=img;

for k=1:N %N:numberofspheres
    edtImage = bwdist(~intersection);   % Euclidean map
    radius = max(edtImage(:));          % Inradius in voxel units

    % Note: rMin is given in Cartesian units, not in voxel units, hence the
    % multiplication with the voxel size below
    if radius*voxel_size(1,1)<rMin % Break the loop if the minimum radius has been met using less than N
        spheres
        warning(['The mimimum radius rMin=',num2str(rMin),' has been met using ', num2str(k-1),' spheres
            '])
        break
```

```matlab
    end

    [yCenter, xCenter, zCenter]= ind2sub(size(intersection),find(edtImage == radius)); % Center in voxel units

    dists = sqrt(sum(bsxfun(@minus,centroid,[xCenter,yCenter,zCenter]).^2,2));
    [~,i]=max(dists); % Index of the inscribed sphere closest (min) / farthest (max) to the centroid

    sph=sqrt( (dx-xCenter(i)).^2 + (dy-yCenter(i)).^2 + (dz-zCenter(i)).^2 ) > (1-overlap)*radius; % Sphere
    intersection=and(intersection,sph); % Append the new sphere in the particle

    xC=xCenter(i)-halfSize(1); %+1
    yC=yCenter(i)-halfSize(2); %+1
    zC=zCenter(i)-halfSize(3); %+1

    clump.positions(counter,:)=[yC,xC,zC]*voxel_size(1,1)+[aveX,aveY,aveZ]; % Here we add [aveX,aveY,aveZ] to return to the initial coordinate system
    clump.radii(counter,1)=radius*voxel_size(1,1);
    counter=counter+1;
end

[clump.minSphere.centroid, clump.minSphere.radius]=min(clump.radii);
[clump.maxSphere.centroid, clump.maxSphere.radius]=max(clump.radii);
clump.numSpheres=length(clump.radii);

%% Plot spheres in voxelised space (if overlap>0, these are not the actual spheres, but the scaled ones, used to facilitate the overlap)
% figure()
% load('config.mat') % "config" is only used to visualise transparent voxelated images
% volshow(intersection,config);

% Restore the mesh in the original coordinate system
P(:,1)=P(:,1)+aveX;
P(:,2)=P(:,2)+aveY;
P(:,3)=P(:,3)+aveZ;

%% Plot clump and mesh in Cartesian space (optional)
if visualise
%   patch('Faces',F,'Vertices',P,'FaceColor','g','EdgeColor','none','FaceAlpha',0.4)
%   patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.15,'EdgeColor',[0.2,0.7,0.2],'EdgeAlpha',0.1);
    patch('Faces',F,'Vertices',P,'FaceColor','g','FaceAlpha',0.2,'EdgeColor','none','EdgeAlpha',0.4); % [0,0.4,0]
    axis equal
    camlight
    box on; grid on
    alpha 0.5

    %% Plot spheres
    [X,Y,Z]=sphere(20);
    for i=1:length(clump.radii)
        hold on
        x=clump.positions(i,1);
        y=clump.positions(i,2);
        z=clump.positions(i,3);
        r=clump.radii(i);
        surf(r*X+x, r*Y+y, r*Z+z, 'FaceColor',rand(1,3), 'EdgeColor','none','FaceAlpha',1)
    end
end

%% Export clump (optional)
% Output is offered in the generic format x_y_z_r.
% For more specialised formats, try the ExportClump module.
if ~isempty(output)
    dlmwrite(output, [clump.positions, clump.radii], 'delimiter', ',', 'precision', '%10f')
end
end
```

Figure B.4: `ExtractSurface.m`

```matlab
function [faces,vertices]=ExtractSurface(clump, N_sphere, N_circle, visualise)
%% Tesselation of the surface of a clump into a surface mesh
% 2021 \copyright V. Angelidakis, S. Nadimi, M. Otsubo, S. Utili.

%% INPUT:
%   — clump    : either "clump" object or N x 4 matrix with columns of [x,y,z,r], where x,y,z the
     centroid of each sphere and r its radius
%   — N_sphere : Number of vertices on the surface of each member—sphere of the clump
%   — N_circle : Number of vertices on the circle defined as the intersection of two overlapping spheres
%   — visualise: Boolean whether to plot the generated surface mesh of the clump surface

%% OUTPUT:
%   — faces    : faces of generated surface mesh
%   — vertices : vertices of generated surface mesh

%% EXAMPLE
%
% N_sphere=400;
% N_circle=200;
% clump=[
%   1,0,0,1.1;
%   2,1,0,1.1;
%   3,0,0,1.2;
%   ];
% [faces,vertices]=ExtractSurface(clump,N_sphere,N_circle,visualise)

%% Check format of input
switch class(clump)
    case 'struct'
        if isfield(clump,'positions') && isfield(clump,'radii')
            if size(clump.positions,2)<3
                error('Invalid format; clump.positions should have size N x 3!')
            end
            if size(clump.radii,2)>1
                error('Invalid format; clump.radii should have size N x 1!')
            end
            spheresList=[clump.positions,clump.radii];

        else
            error('Invalid format! The struct should have fields "positions" and "radii"!')
        end
    case 'double'
        if size(clump,2)~=4
            error('Invalid format, should be x,y,z,r!')
        end
        spheresList=clump;
end

[x,y,z,r]=deal(spheresList(:,1),spheresList(:,2),spheresList(:,3),spheresList(:,4));

%% Main body of the function
%% Import Dependencies
```

```
addpath(genpath('MyCrust'))

%% Contact detection between all spheres (all possible combinations) — Record interactions
interactions=[];
ind=1;
for i=1:size(spheresList,1)—1
    for j=i+1:size(spheresList,1)
        if i==j
            continue
        end
        inContact=sphereContact(spheresList(i,:),spheresList(j,:));
        if inContact
            interactions(ind,1:2)=[i,j];
            ind=ind+1;
        end
    end
end

%% Generate points for each sphere
for i=1:size(spheresList,1)
    [S{i}.vertices,S{i}.faces]=makeSphere(x(i),y(i),z(i),r(i),N_sphere);
end

%% Perform contact detection to detect and delete points of each sphere that are included in other
     spheres, in order to get only the points on the surface of the clump
for i=1:size(interactions,1)

    % For interaction [sphere1,sphere2], check which vertices of sphere1 are inside sphere2
    for j=size(S{1,interactions(i,1)}.vertices,1):—1:1 % start deleting from end to start
        if spherePotential(S{1,interactions(i,1)}.vertices(j,:),spheresList(interactions(i,2),:),true)
            v=S{1,interactions(i,1)}.vertices(j,:);
            %           scatter3(v(:,1),v(:,2),v(:,3),20,'b','filled')
            S{1,interactions(i,1)}.vertices(j,:)=[];
        end
    end

    % For interaction [sphere1,sphere2], check which vertices of sphere2 are inside sphere1
    for j=size(S{1,interactions(i,2)}.vertices,1):—1:1 % start deleting from end to start
        if spherePotential(S{1,interactions(i,2)}.vertices(j,:),spheresList(interactions(i,1),:),true)
            v=S{1,interactions(i,2)}.vertices(j,:);
            %           scatter3(v(:,1),v(:,2),v(:,3),20,'b','filled')
            S{1,interactions(i,2)}.vertices(j,:)=[];
        end
    end

end
% scatter3(S{1,1}.vertices(:,1),S{1,1}.vertices(:,2),S{1,1}.vertices(:,3),'filled')
% scatter3(S{1,2}.vertices(:,1),S{1,2}.vertices(:,2),S{1,2}.vertices(:,3),'filled')


%% Calculate points on the intersection of each pair of interacting spheres
vertices=[];
for i=1:size(interactions,1)
```

```
    n=spheresList(interactions(i,2),1:3)-spheresList(interactions(i,1),1:3); % (not normalised) normal
         vector of each interaction

    d=norm(n); % centroidal distance between sphere1-sphere2 in each interaction
    n=n/norm(n); % normalised normal vector of each interaction

    r1=spheresList(interactions(i,1),4); % radius of sphere1
    r2=spheresList(interactions(i,2),4); % radius of sphere2

    h = sqrt( (2*r1*d)^2 - (r1^2 + d^2 - r2^2)^2 )/(2*d); % Radius of intersection circle
    alph=acos( (r1^1+d^2-r2^2) / (2*r1*d) );
    h1=r1*(1-cos(alph));

    C=spheresList(interactions(i,1),1:3)+n*(r1-h1); % Contact point

    n3=n;
    n1=[n(3) 0 -n(1)]; % Vector perpendicular to n
    if norm(n1)==0
        n1=[n(2) 0 -n(1)];
    end
    n1=n1/norm(n1); % Normalise n1
    n2=cross(n3,n1);
    %   dot(n1,n3)

    % Generate points of intersection circle
    a=-2*pi:pi/(N_circle/4):2*pi;
    % For each circle point.
    px = C(1) + h * (n1(1) * cos(a) + n2(1) * sin(a));
    py = C(2) + h * (n1(2) * cos(a) + n2(2) * sin(a));
    pz = C(3) + h * (n1(3) * cos(a) + n2(3) * sin(a));

    if imag(px(1,1))>0
        %       i
        break
    end

    %   if ~isnan(px(1,1))
    S{1,interactions(i,1)}.circlevertices=[px' py' pz'];
    %   S{1,i}.circlevertices=[px' py' pz'];

    vertices=[vertices;[px' py' pz']];
    %       scatter3(px,py,pz,20,'filled','r')
    %   end
end

%% Collect vertices from all spheres in one variable
for i=1:size(S,2)
    vertices=[vertices;S{1,i}.vertices];
end
vertices = real(unique(vertices,'rows'));

%% Generate mesh using the Crust algorithm (Amenta et al, 1999)
% p=vertices;
```

```matlab
[faces,~]=MyRobustCrust(vertices);
faces=double(faces); % transform from int32 to double

if visualise
    fig=figure('Position',[200 200 600 600]);
    box on; grid on; hold on;
    axis vis3d equal
    patch('Faces',faces,'vertices',vertices,'FaceColor','c','EdgeColor','none')
    %   trisurf(faces,vertices(:,1),vertices(:,2),vertices(:,3),'facecolor','c','edgecolor','none')
    alpha 0.5
    view(3)
    set(gca,'visible','off')
    camlight
    camproj('perspective')
end
end


function inContact=sphereContact(sphere1,sphere2)
%% Function to perform contact detection between two spheres
% inContact: boolean: whether sphere1 and sphere2 intersect
% sphere1:  [1 x 4] [x,y,z,r]:  test sphere 1
% sphere2:  [1 x 4] [x,y,z,r]:  test sphere 2

d0=norm(sphere2(1:3)-sphere1(1:3)); % Centroidal distance of the spheres
if d0<=(sphere1(4)+sphere2(4))
    inContact=true;
else
    inContact=false;
end
% inContact=sqrt( ( (sphere(1)-point(1))^2 + (sphere(2)-point(2))^2 + (sphere(3)-point(3))^2 )/(sphere
     (4))^2 ) - 1 <= 0;
end


function isInside=spherePotential(point,sphere,allowZero)
%% Function to determine whether a point is inside a sphere
% isInside: boolean: whether the test point is inside the sphere of interest
% point:    [1 x 3] x,y,z:  test point
% sphere:   [1 x 4] x,y,z,r sphere of interest
% allowZero: boolean: whether to consider 0 values as contact, i.e. returning true
if allowZero
    isInside=sqrt( ( (sphere(1)-point(1))^2 + (sphere(2)-point(2))^2 + (sphere(3)-point(3))^2 )/(sphere
         (4))^2 ) - 1 <= 0;
else
    isInside=sqrt( ( (sphere(1)-point(1))^2 + (sphere(2)-point(2))^2 + (sphere(3)-point(3))^2 )/(sphere
         (4))^2 ) - 1 < 0;
end
end


function [vertices,faces]=makeSphere(X,Y,Z,radius,N) %radius
%% Function to create a surface mesh of a sphere with radius r, centered at (x,y,z) with N vertices.
```

```matlab
% Returns vertices/faces

vertices=zeros(N,3);
inc=pi*(3-sqrt(5));
off=2/N;
for k=0:N-1
    y=k*off-1+off/2;
    r=sqrt(1-y^2);
    phi=k*inc;
    vertices(k+1,1:3)=[cos(phi)*r*radius, y*radius, sin(phi)*r*radius];
end
vertices=vertices+[X,Y,Z];

faces=convhull(vertices);
% patch('vertices',vertices,'Faces',k,'FaceColor','g')
end
```