New distance algorithms for optimisation and contact mechanics problems

Mattia Montanari

Exeter College
Department of Engineering Science
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas Term, 2017

Dedicate nor to gold,
nor to love, nor to heaven.

Abstract

NEW DISTANCE ALGORITHMS FOR OPTIMISATION
AND CONTACT MECHANICS PROBLEMS

Exeter College, University of Oxford
Michaelmas Term, 2017

This thesis presents a new solution to a difficult problem: computing the minimum distance between bodies in virtual environments. This problem encompasses fields like computer graphics, robotics and engineering but, because each subject has specific requirements, the algorithms are hardly transferable from a field to another. For instance, in computer animations the major concern is speed, whereas applications in the field of computational mechanics require the highest possible level of accuracy. Furthermore, engineering simulations involve a large number of geometrical primitives often of different types, such as quadrics, polytopes and splines, that displace, deform and change topology. For these reasons, distance queries on arbitrary bodies are a major computational bottleneck to large engineering simulations.

A versatile framework that improves the state-of-the-art distance algorithms for applications in computational mechanics is presented. The improvements are demonstrated throughout this thesis; firstly, by facilitating the coupling of different numerical methods, such as discrete and finite elements. Secondly, by exploiting spatial coherence and the data caching capability of modern hardware architectures to accelerate the solution of distance queries. Lastly, by addressing the numerical instabilities caused by floating-point arithmetic.

The contribution of this research is a set of three algorithms that can either work independently or combined into a hierarchical framework. These are: (i) the Signed Volumes method: a new recursive procedure for point–simplex distance queries; (ii) an improved version of the Gilbert-Johnson-Keerthi (GJK) algorithm for faster and more accurate distance queries between convex bodies; and, (iii) an innovative hierarchy of bounding volumes for arbitrary representations of concave objects.

To demonstrate the applicability of the new algorithms, they have been implemented into an in-house solver for the analysis of impacts as well as into a third-party software package. Benchmark tests, analytical calibration, comparison with commercial software and verification with published works demonstrate the improvements brought by the novel algorithms. Finally, two applications are presented: a contact mechanics problem for the packing of sand grains with realistic morphology, and a morphology optimisation problem for the generation of representative volume elements (RVEs) of polycrystalline materials. The results show that both applications benefit hugely in terms of computing time and accuracy from the novel algorithms.

Acknowledgements

This is the end of a four years journey that would not have been possible without Prof. Nik Petrinic and Dr Ettore Barbieri. They made this experience professionally rewarding and truly life-changing. It was during the countless discussions with them, often late in Begbroke, that I formulated those key questions underpinning this thesis.

I am truly thankful to all academics of the group of Solid Mechanics and Materials Engineering for their support. Special thanks go to: Prof. Alan Cocks, Prof. David Hills, Prof. Clive Siviour and Dr Alice Cicirello. In many occasions they have helped me or challenged me — which lately I discovered being two sides of the same coin.

I want to express my profound gratitude to the people of the Impact Engineering Team. Robert Gerlach, with whom I worked in my first month, welcomed me and gave me important advises. Antonio Pellegrino, who took Robert's place in the "aquarium", inspired me on several occasions and has always been a firm point of reference. Petros Siegkas helped me understanding and modelling ballistic tests. Simone Falco put me on the right track for writing the last chapter of this thesis. Kovthaman Murugaratnam explained many details about binary trees. Francesco De Cola shared with me all his knowledge about granular media. Nicola Bombace saved me a lot of time by pointing to the best references on wave propagation. Karen Bamford solved all logistic problems that I encountered during these years. Colin first, and Hisham afterwards fixed my workstations, license servers and disks.

Special thanks also go to the many friends who never let me down. Some of them kept me awake at night, talking and talking. Others walked with me by the shore, on the ridge of a mountain, in the streets and parks of Oxford. With other I wrote songs, wove pieces of picture, organised pranks, explored pubs, tasted cheeses, played table football and "polish the horse shoes". Together we travelled a lot and we ate a lot, but the best part certainly was cooking together. How many nights spent playing computer games? And all those weddings: the best one was when I had red tie and socks on. Anything related to mountain was the best distraction from this thesis really because I had good friends skiing, hiking or climbing with me. With other friends I dreamed of creating something, and we did it over a cup of coffee or tea. When I faced difficulties, and had tough times I knew I could count on friends for a late burrito. Even when I could not see my friends they were right next to me. There have been friends for all occasions, and Giulia has been present in all of these.

Finally, thanks to my family. We could not be better. Together.

Dissemination

The work presented in this thesis was extracted from research articles either published or submitted for publication by the author prior the completion of his DPhil. A detailed list is shown below.

Peer-reviewed articles

1. Montanari, M., Petrinic, N. (submitted for review to SoftwareX in April 2018) OpenGJK for C, C# and Matlab: reliable solutions to distance queries between complex shapes in three-dimensional space.
2. Montanari, M., Petrinic, N. (under development for Computer Methods in Applied Mechanics and Engineering) A new intrinsic approach to contact search.
3. Montanari, M., Petrinic, N., Barbieri, E. (2017) Improving the GJK algorithm for faster and more reliable distance queries between convex objects. *ACM Journal Transactions on Graphics* 36, 3, Article 30, 17 pages.

Conference papers

1. Liping., L., Benson, D., Montanari, M., Petrinic, N. (2018). Recent developments in isogeometric analysis with solid elements in LS-DYNA. In Proc. of the *15th International LS-DYNA User Conference 2018, Dearborn, Michigan.*
2. Montanari, M. , Liping., L., Petrinic, N. (2017). Isogeometric models for impact analysis with LS-DYNA. In Proc. of the *11th European LS-DYNA conference 2017, Salzburg, Australia.*
3. Montanari, M. , Siegkas, P., Pellegrino, A., Petrinic, N. (2016). NURBS-based isogeometric analysis for ballistic evaluation of titanium plates. In Proc. of the *ECCOMAS 2016 - 7th European Congress on Computational Methods in Applied Sciences and Engineering* Vol. 4, Pages 8047-8056.
4. Montanari, M., Pellegrino, A., Siegkas, P., Barbieri, E., Petrinic, N. (2016). Systematic study of p-refined isogeometric analysis patches with application in ballistic test of titanium plates. In Proc. of the *1st Int. Conf. on Impact Loading of Structures and Materials.*

Talks

1. Invited speaker at SIGGRAPH 2017. Talk title: *Improving the GJK algorithm for faster and more reliable distance queries between convex objects.* Los Angeles, California. Aug. 2017. Available from *ACM Journal Transactions on Graphics* 36, 4, Article 151a.
2. Montanari M., De Cola F., Murugaratnam K., Barbieri E., Petrinic N. (2015). *Multi-level contact detection search for packing simulations of granular media.* 8th International Congress of Croatian Society of Mechanics, 1-2 October 2015, Opatija, Croatia.

Contents

Notation

Symbols used in this thesis, with the exception of those used only once

Roman Symbols

$\mathbf{A}$, $\mathbf{M}$	Squared matrices
aff	Affine hull
conv	Convex hull
$\mathbf{c}$	Center of sphere
C	Cofactor of a matrix
C	Curve
c	Wave travelling speed
d	Euclidean distance
n, r	Dimension of $\mathbb{R}$
E	Young modulus
e	Void ratio
r	Fictitious simplex
h	Support funtion
H	Mesh size
I	Set of indices
i, j, l, k	Summation indices
K	Set of vectors
k	k-th iteration of iterative algorithm
M	First minor of a matrix
m, n	Number of bodies, sub-bodies or points
O	Origin of $\mathbb{R}^n$

$\mathbf{qp}$ ... Vectors

P Convexity

P, Q, ... Sets of points

p, q, s ... Points

$\mathbb{R}$ Real coordinate space

τ Simplex

s Support mapping

t Time

u Displacement

V Voronoi region

v Velocity

$\mathbf{w}$ ... Position vector of point w

W Smallest set of vertices supporting the point of minim norm

z Witness point

Z Coordinatoin number

z_b Size of buffer zone

Greek Symbols

Δt Time increment

Ω Body or sub-body

ε penalty factor, machine precision

ε_c Tolerance on convexity

ε_d Tolerance on distance for the broad search

ε_{rel} Relative tolerance of an algorithm

ε_{tol} Abosolute tolerance of an algorithm

Γ Frontier of Ω

κ Tuple of vertices defining a Voronoi region

$\boldsymbol{\lambda}$ Column array of barycentric coordinates

λ Barycentric coordinate

μ Volume form

$\nu(\Omega)$ Point of minimum norm of Ω

ρ Radius of sphere, or density

Ξ Knot vector for NURBS

Acronyms / Abbreviations

3D, 2D Three- and two-dimensional

AABB Axis-aligned bounding box

ADT Alternating digital tree

CSO Configuration space obstable

CAD Computer aided design

CPU Central processing unity

DCT Diffraction contrast tomography

DEM Distinct element method

FEM Finite element method

GJK Gilberg-Johnson-Keerthi (algorithm)

GPU Graphics processing unity

NaN Not a number

NURBS Non-uniform rational B-spline

PDF Probability density function

RVE Representative volume element

SIMD Single instructruction, multiple data

Other Symbols

$\mathcal{O}$ Big O, optimisation function

Chapter 1

Introduction

1.1 Distance queries in computational mechanics

1.1.1 General

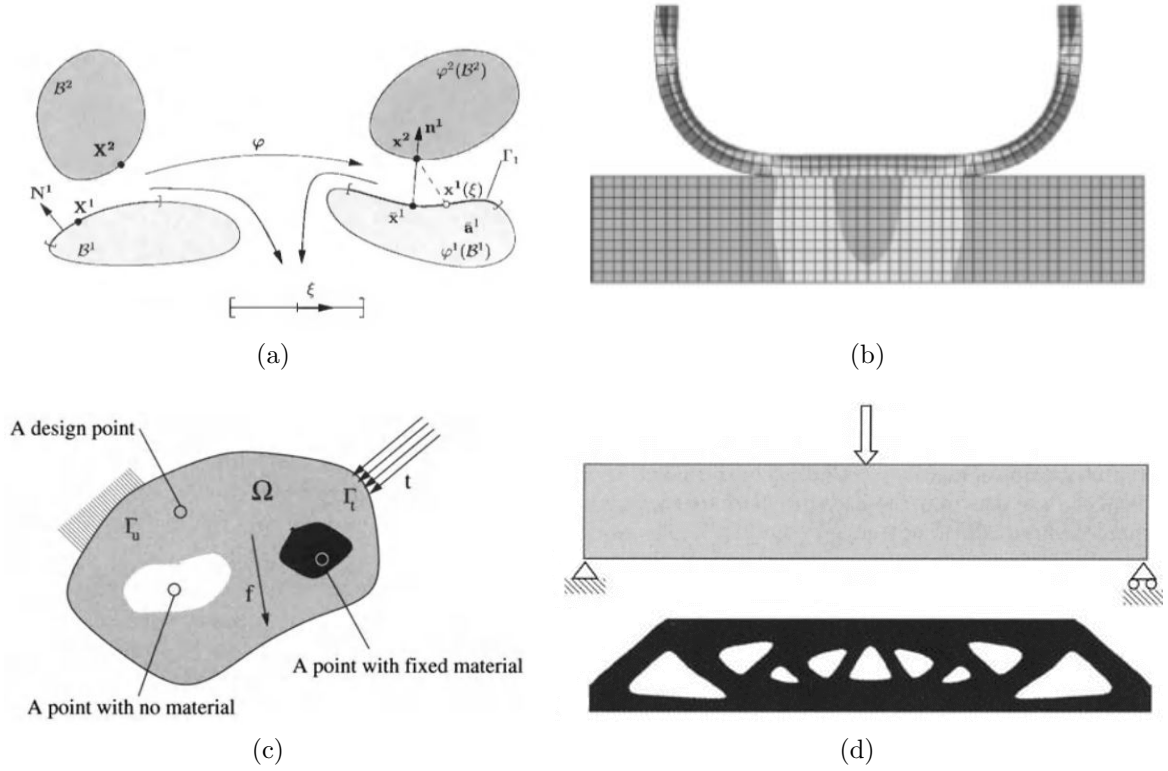
Engineers need to measure distances. Distance queries quantify how far apart objects are, and based on this information engineers take decisions all the time.

The most common metric used to measure distances in mechanics is the *Euclidean distance*. Its simplest interpretation is the length of a straight line segment connecting two points. Given p and q, two points in the three-dimensional (3D) space $\mathbb{R}^3$, the Euclidean distance is defined as follows:

$$d(p,q) = \sqrt{\sum_{i=1}^{3}(p_i - q_i)^2}. \tag{1.1}$$

Effectively the norm of a vector $\mathbf{qp}$, $d(p,q)$ finds countless applications in engineering.

Of particular interest are those problems which require the measure of the *minimum* distance between two objects. This class of problems is of utmost importance to mechanical engineers and is therefore the focal point of this thesis.



(a) (b) (c) (d)

Figure 1.1: Formulation of a contact problem (a) and numerical simulation of composite tube sliding on flat surface (b) from (247). Formulation of an optimisation problem (c) and application to structural engineering problem (d) from (16).

Two examples of engineering applications are depicted in Figure 1.1; these rely, in different ways, on the distance expressed by Eq. (1.1). The first one formulates, see Figure 1.1(a), a contact mechanics problem and shows an application in which two objects collide and deform as the minimum distance between them is null (Figure 1.1(b)). The second one concerns an optimisation problem, see Figure 1.1(c), in which the use of distance queries is more subtle. In fact, an optimisation process may target the stress distribution, as in Figure 1.1(d), or the distance between material regions.

As more and more engineers solve these problems using computational techniques, nowadays computers need to measure distances. Hence the rise of *distance algorithms*.

1.1.2 Distance algorithms

A distance algorithm is that part of a computer program that solves distance queries. In computational mechanics, since the late 1970s, these algorithms are widely used in contact mechanics, optimisation, mesh generation and real-time simulations.

Despite the broad spectrum of applications, distance algorithms tend to be highly specialised and to solve a single specific problem. Their applicability is therefore very limited and this is because each application has particular requirements. For example, the contact analysis in Figure 1.1(b) involves many small squared elements that remain in contact for a prolonged period, whereas the voids in the optimisation problem of Figure 1.1(d) have arbitrary shapes and they are always at a distance from each other. These differences yield to specific requirements.

The desirable characteristics of distance algorithms are: speed, scalability and accuracy (64). Distance queries should be solved as quickly as possible to limit the overall computing time. They should also be scalable to address, within a specific application, various scenarios, such as: rigid or deformable bodies, and from few units to millions of bodies. Finally, for mechanical problems whose solutions depend directly on distance queries, accuracy is the most important requirement.

The design and the implementation of distance algorithms is often very involved. The design requires notions of computer science, mathematics and a good understanding of the underlying mechanical problem. In fact, high performance can only be achieved by exploiting the architecture of modern processor units and by tailoring the algorithm to the specific requirements of the particular application.

1.1.3 Motivations for further research

Despite the nearly forty years of development, distance algorithms are still a major computational bottleneck for the solution of many mechanical problems.

The community of mechanical engineers has overlooked the correlation between speed, accuracy and robustness of distance algorithms and the impact that these have on the solution of mechanical problems. For instance, inadequate accuracy can ruin the whole simulation, or a drastic reduction of computing time may be achieved by minimising the number of operations taken by distance algorithms. To date, very little is known about what is the most effective way to reduce the computational cost of distance algorithms; for this reason, furthers studies are needed.

In recent years, a significant effort has been made to couple numerical methods such as finite elements, finite volumes, mesh-free, distinct elements etc..., but distance algorithms are a persisting barrier. Each method comes with a particular mathematical formulation that requires distance algorithms to be flexible but, since these are specialised for specific problems, their limited applicability makes the coupling troublesome.

Overall, faster and more versatile distance algorithms will enable engineers to solve problems that today are either too expensive or too complex.

1.2 Aim and objectives

The present work aims to advance the state-of-the-art in distance algorithms for computational mechanics, particularly in terms of speed and versatility.

The objectives of this thesis are:

- To review existing methodologies in the field of computational mechanics.
- To identify limitations and pitfalls of existing methodologies.
- To test, by means of established and/or new benchmarks, the performance of existing methodologies.
- To design new algorithms that are faster and more versatile.
- To verify logic, performance and implementation of new algorithms.
- To implement standalone libraries which facilitate the adoption of the new algorithms into existing software packages.
- To validate the claims on capability and versatility by means of applications in contact mechanics and topology optimisation.
- To disseminate the main findings and, whenever possible, the source code of the new algorithms.

1.3 Research strategy

In the preface to their book (172), Nocedal and Wright wrote that "*Knowledge of the capabilities and limitations of these* (optimisation) *algorithms leads to a better understanding of their impact on various applications, and points the way to future research on improving and extending optimization algorithms and software*". Distance algorithms indeed are optimisation procedures, and the preceding quote perfectly summarises the research strategy of the present work.

In particular, this thesis focuses on algorithms aimed at tackling distance queries at three levels: body (non-convex), sub-body (semi-convex) and primitives (elementary shapes). These three are studied and validated independently, but they are eventually

combined together into a hierarchical framework for the solution of complex problems.

1.4 Novelties

This thesis presents new insights on distance algorithms and their design for solving problems in computational mechanics. The algorithms are based on new studies, never published before, that relate accuracy and robustness. These provide a better understanding from which three novel methods are designed:

1. A robust and accurate method for the solution of point–simplex (elementary shapes) distance queries.
2. A routine that reduces the computing time for evaluating Eq. (1.1) when both p and q belong to arbitrary convex bodies.
3. An innovative intrinsic approach to broad contact search for non-convex, semi-convex and deformable bodies.

1.5 Thesis layout

The thesis is structured as follows:

Chapter 2 presents the literature review. This encompasses the fields of robotics, computer graphics, computer vision, computational mechanics and computational geometry. Furthermore, it provides basic notions on distance algorithms to understand the principal features and pitfalls of existing methods.

Chapter 3 presents three new algorithms for the solution of distance queries. For each one of these is provided: an introductory section, a detailed description of the

formulation and the implementation of the method.

Chapter 4 presents benchmark tests, analytical calibration, comparison with commercial software and verification with published works to validate the novel algorithms.

Chapter 5 presents an application in which the new algorithms are used to solve a contact mechanics problem. In particular, the mechanical packing of sand grains under gravitational load is considered.

Chapter 6 presents the solution of a topology optimisation problem that aims to generate representative volume elements (RVEs) for multi-scale simulations.

Chapter 7 provides concluding remarks and recommendations for future works.

Chapter 2

Literature Review

This chapter reviews the most relevant distance algorithms published in the literature. The survey begins by presenting the broad spectrum of applications in which such algorithms play a crucial role. This review is concludes that each application has specific requirements: applications in the field of computational mechanics appear to require a higher level of accuracy than the algorithms employed, for example, in computer animations. These requirements are often difficult to fulfil and give rise to the algorithm design issues reviewed in the second part of this chapter. The most efficient methods published in the literature are *hierarchical frameworks*: a chain of algorithms that tackle distance queries from a coarse to a fine level. One can usually distinguish three major levels. The first one approximates arbitrary bodies with bounding volumes, and sorts them in spatial data structure as these deform and displace. The second is restricted to simpler and strictly convex shapes. The third considers only basic primitives, the simplest geometrical entities describing a solid body, such as spheres and triangles. This chapter discusses each level, drawing particular attention on robustness and performance of existing distance algorithms.

2.1 Applications of distance algorithms

Algorithms for the solution of distance queries are ubiquitous in virtual environments, encompassing applications in computer games, computer animations, computer aided design (CAD), computer vision and computer simulations.

Most, if not all, numerical methods in computational mechanics rely on distance algorithms. These numerical methods include, but are not limited to: the finite element method (FEM) (14, 261), the distinct element method (DEM) (51, 98, 127), extended FEM (X-FEM) (216), proper generalised decomposition (PGD) (44, 118), boundary element method (BEM) (17), isogeometric analysis (IGA) (101), mesh-free methods (59, 128), granular element method (GEM) (129), numerical manifold method (NMM) (210) and combined finite-discrete element method (158). These approaches need distance algorithms to solve, for example, collision queries in contact mechanics to model friction (121, 207), fractures (18, 134, 182), impacts (19) and fluid–structure interaction problems (69). Contact detection however is not the only application of distance algorithms in computational mechanics. Mesh generation (25, 188), tracking of discontinuities (17) and neighbour search (125) are some of the interdisciplinary applications in common other disciplines.

Robotics has a long history in the development of distance algorithms and it has influenced all other disciplines. The ancestors of modern procedures are simple contact detection algorithms for convex bodies (35, 47, 142, 150, 223). General frameworks for contact detection in 3D arose starting from the late '70s (31); a decade thereafter modern distance procedures (39, 80, 89, 132, 204) and four-dimensional collision detection schemes (36) became available. Lately, the success of CAD software brought a sudden

prosperity to distance algorithms. In particular, CAD introduced the need for computing the accurate distance between a point and an arbitrary surface (94, 191). Today CAD packages are mature products, and the research activities on distance algorithms are now driven by the entertainment industry.

The computer graphics community is the most active in the development of new distance algorithms. These are used in rending computer animations and for operating computer games. For example, ray tracing is a common rendering technique that produces photorealistic images, and it necessitates high-performance distance algorithms and dedicated hardware to deliver smooth visual experiences at 60 renderings per second (149, 171).

In order to fulfil the high frame rate requirement in real-time and to give the user a realistic perception of the scene, the computer graphics community has specialised not only the software, but also the hardware. Nowadays, distance queries are commonly accelerated by graphic processing units (GPUs) (64, 116, 184, 211, 250). These have added more computing power, and hence enabled more realistic renderings. Similarly, physical-based game engines have been developed to improve the user experience (24, 85). Despite the use of GPUs, the computing power remains limited and computer games cannot afford to deliver less than 60 images per seconds. Alike for other real-time and computer graphics applications, game developers had to introduce approximations in the physics of the scenes to meet this requirement (153, 228).

Apart from rare exceptions, the highly specialised distance algorithms developed for computer graphics are either not suitable or not applicable to computational mechanics, and there are many reasons for that. First and foremost accuracy, which is often not adequate to solve engineering problems. Another barrier is the use of GPUs

since the engineering and medical communities still rely heavily on CPUs. Moreover, computer games involve much larger scenes (called *world*) and the simulated time is not comparable to the one simulated in computational mechanics. Therefore, the particular scenarios arising in computational mechanics translate in specific requirements for the design of distance algorithms.

The requirements specific to distance algorithms for computational mechanics applications are outlined in the following section.

2.2 Algorithm design issues

The desirable characteristics of a distance algorithm for engineering applications are versatility, robustness and low computational cost. In practise, however, computing the Euclidean distance between two sets of points is non-trivial. Each set is usually non-finite and defines a *body* whose representation varies for different applications; finding the closest points between two bodies is the first difficulty. Furthermore, if the domain comprises m bodies, there are $\mathcal{O}(m^2)$ possible contact configurations; testing each one of these is not a viable solution. Another difficulty is that the algorithms run on finite-precision machines prune to rounding and cancellation errors. This section examines the implications that such difficulties have on the design of distance algorithms.

2.2.1 Representation of solids

A peculiar characteristic of computational mechanics is that solid bodies may be represented in many different ways. In other fields there are essentially no options: in computer graphics polygons (usually triangles) are used, whilst in CAD and robotics,

non-uniform rational B-splines (NURBS) are the most common ones. The representations employed in engineering problems are quadrics (e.g. spheres, ellipses and cylinders), polytopes (e.g. polygonal meshes, tetrahedrons and hexahedrons), splines (e.g. B-splines, T-splines and NURBS) and, more rarely, compounds of these. The large variety of geometrical representations makes the design of a versatile distance algorithm particularly difficult.

Let us illustrate with an example the consequences that different representations have on the distance algorithms. Firstly, recall that a subset $\Omega \in \mathbb{R}^n$ is *convex* if, for any two points $p, q \in \Omega$, it exists a point s in the same body defined as: $s = (1 - \lambda)\, p + \lambda\, q$, with $0 \leq \lambda \leq 1$. Then consider a pair of disjoint convex bodies Ω_P and Ω_Q in $\mathbb{R}^n$; for simplicity $\Omega_Q = \{O\}$ and only the representation of Ω_P is arbitrary. The distance between the two bodies is a general form of Eq. (1.1):

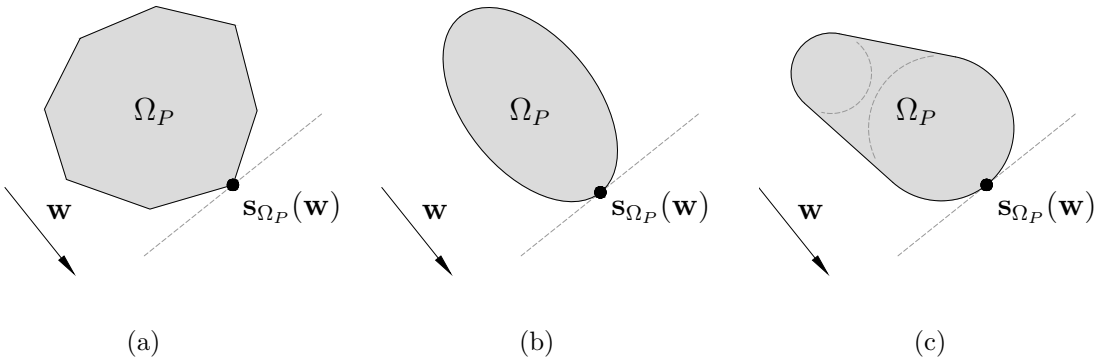
$$d(\Omega_P, \Omega_Q) = \min\{\|p - q\| : p \in \Omega_P,\, q \in \Omega_Q\} \tag{2.1}$$

where the Euclidean norm $\|\cdot\|$ of the vector $p - q$ is given by $\|\mathbf{qp}\| = \sqrt{\mathbf{qp} \cdot \mathbf{qp}}$. Since Ω_Q is a singleton, Eq. (2.1) is equal to $d(\Omega_P, 0)$, which is the norm of a unique vector $z_P - O = \mathbf{w}$ with norm non-null. The unknown z_P is the furthest point of Ω_P from the hyperplane defined by $\mathbf{w}$ along the direction $\mathbf{w}$ and is obtained by evaluating the so-called *support function*. The support function $h_{\Omega_P} : \mathbb{R}^n \to \mathbb{R}$ is defined for all points of a convex body Ω_P as:

$$h_{\Omega_P}(\mathbf{w}) = \max\{(p - O) \cdot \mathbf{w} \;:\; p \in \Omega_P\}. \tag{2.2}$$

The solution of h_{Ω_P} is a point, not necessarily unique, $s_{\Omega_P}(\mathbf{w}) \in \Omega_P$ called *support mapping*. From this is defined a vector $\mathbf{s}_{\Omega_P}(\mathbf{w}) = s_{\Omega_P}(\mathbf{w}) - O$ that verifies:

$$h_{\Omega_P}(\mathbf{w}) = \mathbf{s}_{\Omega_P}(\mathbf{w}) \cdot \mathbf{w} \tag{2.3}$$



(a) (b) (c)

Figure 2.1: Illustration of support mapping $\mathbf{s}(\mathbf{w})$ for polygon (a), ellipse (b) and compound (c) along a given direction $\mathbf{w}$.

By substituting Eq. (2.2) into Eq. (2.3) one obtains that:

$$\max\{(O-p)\cdot\mathbf{w} \ : \ p\in\Omega_P\} = \mathbf{s}_{\Omega_P}(\mathbf{w}). \tag{2.4}$$

This implies that $s_{\Omega_P}(\mathbf{w})$ is a solution of the distance query whose computation, however, it depends on the nature of the representation of Ω_P. See Figure 2.1 for an illustration of Eq. (2.3) on different representations (e.g. polytope, ellipse, compound of spheres).

This example demonstrated that the difficulty in designing a versatile algorithm lies in the evaluation of $s_\Omega(\mathbf{w})$ for an arbitrary discretisation of Ω. In what follows are reviewed the formulae to compute $\mathbf{s}_\Omega(\mathbf{w})$ for common representations of convex bodies. It is worth recalling that in practical applications non-convex bodies are always decomposed into convex sub-bodies.

Quadrics The support mapping of a sphere in $\mathbb{R}^3$, or a disk in $\mathbb{R}^2$, with radius $\rho\in\mathbb{R}^+$ and centred in $\mathbf{c}$, may be computed with the following formula:

$$\mathbf{s}_{\text{sphere}}(\mathbf{w}) = \mathbf{c} + \rho\frac{\mathbf{w}}{\|\mathbf{w}\|}. \tag{2.5}$$

Quadrics have been successfully employed in DEM simulations of granular media (146), manufacturing processes (163, 213), bed reactors (13), shallow flows (32), fracture (5) and erosion (81). See (88) for treatment of cylinders, ellipsoids and cones.

Polytopes The support mapping of a non-empty and finite set of m points $P = \{p_1, ..., p_i, ..., p_m\}$, which defines the vertices of a polytope in $\mathbb{R}^3$, verifies:

$$\mathbf{w} \cdot \mathbf{s}_{\text{polytope}}(\mathbf{w}) = \max\{\mathbf{w} \cdot (O - p_i) \ : \ p_i \in \ P\}. \tag{2.6}$$

Polytopes are, of course, extremely common in FEM, but also in DEM simulations (232). See (225) for treatment of axis-aligned boxes, which are an important instance of polytope.

Piecewise polynomials Given a parametric curve $C(u)$ that admits first derivative $C'(u)$ in $\hat{u}$, the support mapping writes:

$$\mathbf{s}_{\text{spline}} = C(\hat{u}) \cdot \mathbf{w} \qquad \text{such that} \qquad C'(\hat{u}) \cdot \mathbf{w} = 0. \tag{2.7}$$

This formulation is used, for example, in high-order finite elements but also in NURBS and T-splines; these underpin emerging methods such as isogeometric analysis (101), GEM (4) and NAFEM (208). See (221) for treatment of NURBS surfaces.

Compounds A large variety of complex objects may be efficiently constructed by combining two or more convex objects with arbitrary representation. By means of the *Minkowski difference*, a compound object K may be defined as:

$$K = P - Q = \{k \ : \ k + Q \subseteq P\} \tag{2.8}$$

and it can be shown that K is convex (240). For example, Eq. (2.8) allows to describe capsules by taking the Minkowski difference between a sphere and a line segment. The

support mapping of any compound body is readily provided by:

$$\mathbf{s}_K(\mathbf{w}) = \mathbf{s}_{P-Q}(\mathbf{w}) = \mathbf{s}_P(\mathbf{w}) - \mathbf{s}_Q(\mathbf{w}). \tag{2.9}$$

If K is a capsule, from Eq. (2.5) and Eq. (2.6), the computation of $\mathbf{s}_K(\mathbf{w})$ involves only three dot products and is therefore very efficient.

The problem is that the literature does not present a distance algorithm sufficiently versatile to handle quadrics, polytopes, splines and compound bodies all at once. Probably the only procedure that attempts to achieve this was published by Benson et al. (20). This method however approximates complex bodies with polygons, thus introducing geometrical approximations when representing splines.

To date, the design of an accurate and truly versatile distance algorithm for applications in computational mechanics is still an open challenge.

2.2.2 Robustness

An algorithm lacking of robustness returns inconsistent results and is therefore not suitable for engineering applications. Because the solution of distance queries involve many arithmetic operations, designing a robust distance algorithm is not a simple task.

There are two factors which can compromise the robustness of a distance algorithm. The first one is the solution of an ill-posed distance query, that is: attempting to solve a problem which does not admit unique solution. The second one is the error that occurs when performing arithmetic operations on real numbers represented by floating-point numbers. These two aspects are separately detailed below.

Well-posed problem Notice that for polytopes (polygons) with parallel facets (edges) a distance query has infinitive solutions. Let us consider two convex bodies

in $\mathbb{R}^2$ represented by two non-empty and finite sets P and Q. One can find, from Eq. (2.1) and Eq. (2.6), that it exists a pair of points $z_P \in P$ and $z_Q \in Q$ such that:

$$d(P, Q) = \|z_P - z_Q\| \tag{2.10}$$

These points are called *witness points* and define the *separating vector* $\mathbf{z_Q z_P}$. If the bodies are polygons with parallel edges, there are at least two pairs of witness points which verify Eq. (2.10). This suggests that the affine space in which the bodies reside is not best suited for computing $d(P, Q)$; indeed, a robust algorithm should recast the original distance query of Eq. (2.1) in an equivalent problem which admits unique solution.

Numerical error The support functions reviewed in Section 2.2.1 are well-defined, but their evaluation with floating-point arithmetic is subject to the *machine accuracy*. In practise this can introduce robustness issues even for well-posed problems. Altogether, a robust distance algorithm must take the limitation of floating-point arithmetic into account, and counterbalance the numerical error and deliver consistent solutions.

2.2.3 Performance

The applicability of a distance algorithm is determined by its performance in terms of CPU time and accuracy. These are closely related to the robustness; in fact, robust and accurate algorithms compute consistent results within a specific tolerance. However, it is important that distance queries are solved as quickly as possible to limit the overall computing time.

CPU time Finding the solution to distance queries is often the major bottleneck in computational mechanics (161, 247, 262). This thesis investigates, in Chapter 5 and Chapter 6, two applications in which distance queries are critical: dynamic contact mechanics simulations and morphology optimisation. For these and other applications reviewed in Section 2.1, distance algorithms can account for the majority of CPU time (70, 87, 165).

Accuracy In real-time applications, the computing performance is the main concern but, as already mentioned, this is achieved by introducing coarse approximations. Computer games and collision avoidance systems have only about 50µs to 250µs to resolve *all* distance queries in a simulation step (64). Currently, this is achieved by reducing the accuracy of distance queries and/or of the representation of solid bodies. Coarse approximations are sometimes desirable in computational mechanics, typically for contact detection, as these allow to reduce the possible contact configurations of m bodies from $\mathcal{O}(m^2)$ to $\mathcal{O}(m \log m)$. However, they are not sufficient to resolve real-life problems; for engineering purposes, a distance algorithm must guarantee a minimum level of accuracy — which usually corresponds to the machine precision.

2.3 Distance methods for arbitrary bodies

Whilst the formulae in Eqs. (2.5)-(2.9) cover the broad range of mathematical descriptions employed in computational mechanics, in practise they are applicable to convex bodies only. The treatment of arbitrary shapes requires special methods specifically designed to handle dynamic simulations and optimisation problems, for example: large deformations, large displacements, fragmentation and morphology changes. This sec-

tion reviews suitable methods published in the literature and summarises their features.

2.3.1 Hierarchical frameworks

It is computationally more efficient to measure the distance between many simple bodies rather than few complex ones. For this reason the shape of solid bodies assume special significance in view of frequent calls to distance algorithms.

The methods designed for handling complex topologies are called *hierarchical frameworks*. These involve various distance algorithms that interpret bodies as a hierarchy of basic primitives or simplified geometries. At the low-level, coarse approximations are introduced, whilst high-levels provide more detailed descriptions. In applications such as ray casting and contact detection, the former is called *broad*, *coarse* or *global* search, the latter *narrow*, *fine* or *local* search. Of course, the higher the level of details, the higher the processing and memory requirements. For this reason, the broad search skims, quickly and cheaply, over the whole computational domain, and passes some information to the subsequent finer search which is more accurate but also more expensive.

All modern hierarchical distance algorithms comprise two key elements. From the pioneering work of Zhi-Hua and Nilsson (259) for 2D applications, until the most recent contact search for Nagata patches (167), the fundamental components of all hierarchical procedures are *bounding volumes* and *space-partitioning schemes*. These are separately detailed below.

Bounding volumes It is a common practise to enclose complex bodies into volumes that admit simple evaluation to their support mapping; this allows to solve

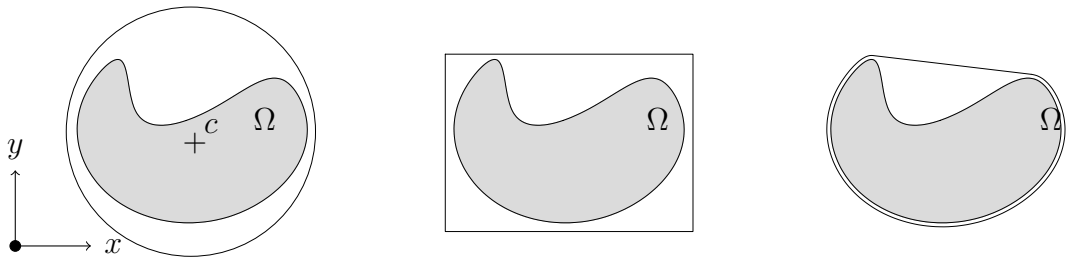


Figure 2.2: Illustration of common bounding volumes for an arbitrary body Ω: sphere (a), axis-aligned bounding box (b) and convex hull (c). All these are extrinsically described by vectors with respect to a frame of reference depicted in the figure.

approximatively, but very quickly, distance queries between complex bodies. The ideal characteristics of a bounding volume are: tight to the enclosed shape, cheap to store in memory and simple to compute. The most common volume types encountered in computational mechanics are illustrated in Figure 2.2 and described below.

- *Spheres*: These are the cheapest to store in memory and are built using randomised algorithms to obtain the smallest sphere (73, 84, 170, 242), however bounding spheres are usually not very tight to the enclosed object. Another limitation is that the support mapping in Eq. (2.5) relies on the centre c, which is an extrinsic quantity that has no relationship with the shape enclosed in the sphere. Consequently, whenever the enclosed body deforms and/or displaces, the bounding sphere needs to be recomputed to updated the vector describing the position of c.
- *Axis-aligned bounding boxes* (AABBs): These are the most common type of bounding volume. Unlike spheres, testing for collision two AABBs requires integers operations which is usually preferred on modern SIMD (single instruction, multiple data) architectures. The generation of AABBs is straightforward and runs in linear time. However, when testing two AABBs, the evaluation of the

support mapping involves two vectors that have no relationship with the enclosed body (225). This shortcoming is common to both bounding spheres and AABBs.

- *Convex hulls*: These are, by definition, the tightest type of bounding volume. The convex hull $\text{conv}(\Omega)$ of an arbitrary body $\Omega \in \mathbb{R}^n$ is the smallest convex set enclosing Ω and is given by the intersection of all convex sets containing Ω. Therefore, if Γ is the frontier of Ω, $\Gamma \subseteq \text{conv}(\Omega)$ and, unlike for bounding spheres and AABBs, there is a strong relationship between the enclosed body Ω and the points defining the bounding volume $\text{conv}(\Omega)$. The advantage is that the bounding volumes is automatically updated whenever the enclosed body moves or changes morphology. However, the generation of convex hulls for polytopes is far more complex and expensive than for AABBs (53).

Space-partitioning Finding the closest neighbour to a query point from a pool of m neighbours is a common problem in computational mechanics. A naïve solution would be to compare the outcome of $\mathcal{O}(m^2)$ distance queries; however, for many applications this approach is too expensive. Space-partitioning schemes are a viable solution designed to reduce this cost: their aim is to limit the number of distance queries by subdividing the domain and organising it into data structures.

The literature presents a huge variety of space-partitioning methods. One way in which these could be classified is body-based and space-based, respectively known as trees and grids. On the one hand, the alternating digital tree (ADT) was one of the first and most successful body-based algorithm. ADTs have been extensively used for closet-neighbour searches in applications such mesh generation and FEM simulations (25). On the other hand, the no binary search (NBS) method (159) was designed to provide

DEM/FEM simulations a more efficient solution for dynamic simulations. Modern space-partitioning algorithms are essentially improvements, variations or combinations of either the ADT of the NBS method. The latter was specialised for mesh free methods (12) and improved to deal with polydisperse assemblies (162, 189, 190, 244). Applications of the NBS have been mostly restricted to spherical particles, and the computational complexity may be bounded to $\mathcal{O}(m)$ by using multi-level grids (125). A different approach, involving AABBs, for dense assemblies of polydisperse particles was presented in (234). If one wishes to deal with more complex geometries, compounds of spheres provide a cost-effective solution. An example was recently presented in (67) which, however, shows results for monodisperse spheres only. FEM simulations are commonly handled with methods inspired to the ADT, such as the augmented spatial digital tree (ASDT) (71), or to the ray tracing methods called b-tree (34).

Figure 2.3 presents a graphical comparison of grids, trees and the binary space partition (BSP) methods. BSPs are widely used for computer graphics applications, see (64) and references therein. NBS, ADT, ASDT and other space-based methods were compared in (93), however that study considered only static scenarios. In dynamic simulations, when position or morphology change, these data structures are likely to become unbalanced and their performance suboptimal (22, 23, 28). An updating step, and therefore extra computing effort, is needed to retain the high performance of the broad search.

The biggest problem when using bounding volumes, space-partitioning or a combination of the two, is to decide when to update their data-structure. This decision must be made at runtime for all simulations in which bodies change morphology and/or position. For space-based method, a common solution is to update the partitioning at

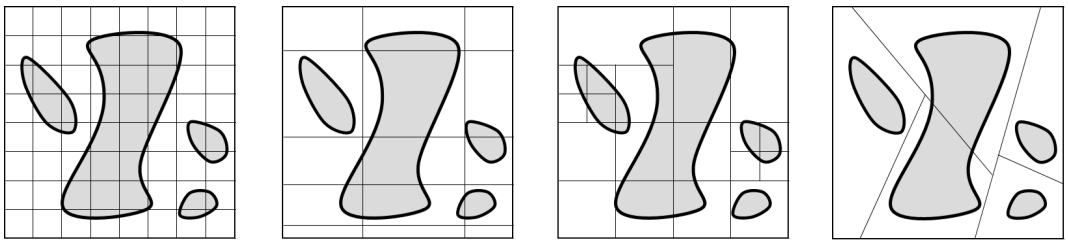


Figure 2.3: Examples of spatial data structures: grid cells (a), adaptive grid (b), quadtree (c) and binary space partition (d).

discrete intervals (15, 21), whereas for bounding volumes this is commonly addressed by creating a *buffer zone* around each body (42, 71, 159, 163). Both solutions are simple to implement and aim to reduce the number of updates to the hierarchical representation of the computational domain; however, they dramatically increase the amount of information passed to subsequent (finer) search levels. For example, in a closest neighbour search problem, these information could be a list of potentially close bodies.

The buffer zone and the cell size are two parameters that impact the performance. The former when rebalancing the tree, the latter when recomputing the boxes. On the one hand, it is important to keep both parameters as large as possible to reduce the number of data-structure updates. On the other hand, however, reducing the size of buffer zone and the cell size can reduce the amount of information passed to the subsequent (and more expensive) search levels. Therefore, defining these values is a difficult choice that users have to make, and unsuitable values may increase significantly the computing cost.

2.3.2 Spatial coherence

A further improvement to the solution of distance queries comes from the observation that, in iterative and time-marching simulations, there is a high level of *spatial coherence*, that is: subsequent solution steps differ only marginally from each other.

Distance algorithms exploiting spatial coherence require extra design and coding effort, but these are usually rewarded in computing time reduction (60, 186, 225, 250). The idea is to cache useful information and to reuse it as "good guess" for the next solution step. For example, the closest neighbour to a query point is inevitably a good guess for a new neighbour-search conducted for the same query point at the next solution step.

Finally, spatial coherence can accelerate any level of the hierarchical framework. This paradigm is used extensively in real-time simulations, robotics and computer graphics application to accelerate both the evaluation of support mappings and finer tests between primitives (105, 122, 135, 165).

2.3.3 Gaps in the literature

A number of gaps emerged from the review of the literature on distance algorithms for arbitrary bodies. Firstly, there is a lack of versatile algorithms: for each numerical method, e.g. FEM or DEM, the literature presents many ad-hoc distance algorithms. These methods are designed and benchmarked only for specific applications, whereas Section 2.1 has shown that the solution of distance queries is crucial for a large variety of problems in computational mechanics. The development of a general framework which deals with all representations of solid bodies (Section 2.2) would allow to couple, very easily, any numerical method.

Secondly, despite the nearly 40 years of development, modern formulations of bounding volumes and space-partitioning schemes are fundamentally unable to adapt as bodies displace and deform. This is a consequence of the extrinsic formulations of their data-structures and the artefacts that these require to cope with morphology updates, namely buffer zone and cell size. The former aims to compensate the lack of relationship between enclosed and bounding volume; the latter compromises the overall size of the data structure with the crucial, but expensive, task of keeping a tree well-balanced.

Finally, the potential of spatial coherence has been often neglected in computational mechanics. Only (71) and (127) used it to reduce the updating costs of binary trees and AABBs. Unlike other fields, in computational mechanics spatial coherence has been restricted to global contact search algorithm only. The few studies which take advantage of this technique are limited to the solution of closest-neighbour search, whereas in robotics and computer graphs is used to accelerate all levels of hierarchical methods.

2.4 Distance methods for convex bodies

A distance query may be formulated as an optimisation problem, and it is well-known that, for convex functions, a local minimum coincides with a global minimum (30, 172). From this analogy between geometry and optimisation theory, it is easy to see the advantage of solving distance queries on convex bodies rather than concave. These can be seen as the domain of the distance function, for which its global minimum is sough numerically. The solution of such problem is mathematically, and geometrically,

easier when carried out on convex domains. For this reason, in practise, all distance queries in computational mechanics are formulated only between convex shapes. This section surveys the literature to identify versatile, robust and fast distant algorithms for convex bodies.

2.4.1 Existing methods

The aim of this section is to identify existing algorithms that (i) are in pursuit of well-posed solutions to distance queries, and (ii) fulfil the requirements listed in Section 2.2.

The importance of defining a mathematically sound problem before solving a distance query is often overlooked by the methods published in the literature. Ill-condition problems can ruin the robustness of distance algorithms as finite-precision arithmetic comes into play. For this reason, defining a problem that admits a unique solution is the key for designing a robust method.

A well-posed problem is derived by recasting the original distance query from physical to *configuration space*. This is done by taking the Minkowski difference between two convex bodies Ω_P and Ω_Q to obtain the *configuration space obstacle* (CSO):

$$\text{CSO} = \Omega_P - \Omega_Q = \{x_P - x_Q \ : \ x_P \in \Omega_P, \ x_Q \in \Omega_Q\} \tag{2.11}$$

which is a set of vectors embedded in an affine space with a fixed frame and origin O (140). It has been shown that a distance query translates into finding the *point of minimum norm* $\nu\,(\text{CSO})$ (39, 89); moreover, since the CSO is convex, it exists a unique vector $\boldsymbol{\nu}\,(\text{CSO})$ which is optimal in the sense that:

$$\|\boldsymbol{\nu}\,(\text{CSO})\,\| = d(\text{CSO}, O). \tag{2.12}$$

The newly published Minimum Norm Duality theorem (52) proves geometrically that

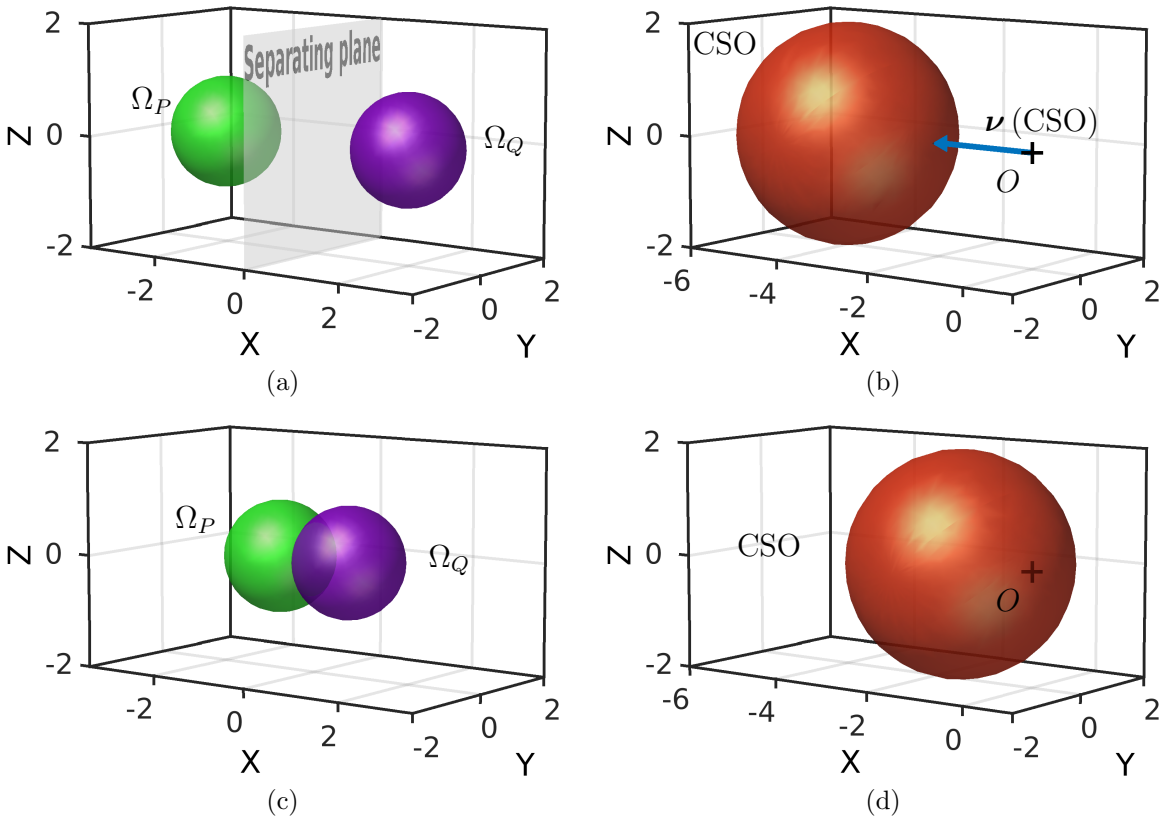


Figure 2.4: Physical (a) and configuration space (b) for separated spheres Ω_P and Ω_Q. The configuration space obstacle (CSO) has a unique point of minimum norm $\boldsymbol{\nu}$ (CSO). Physical (c) and configuration space (d) for colliding spheres P and Q, in this case the CSO includes the origin.

to the minimum distance $d(\text{CSO}, O)$ corresponds to a *separating hyperplane* between the CSO and the origin O. Together with Eq. (2.11), this theorem establishes a link between configuration and physical spaces with the following equality:

$$\boldsymbol{\nu}\,(\text{CSO}) = \mathbf{z_Q z_P} = z_P - z_Q, \tag{2.13}$$

where z_P and z_Q are a pair of witness points in Ω_P and Ω_Q , respectively. Therefore, a well-posed problems equivalent to Eq. (2.10) is defined. Notice that this literature review ignores linear programming algorithms since unable to return a pair of witness points.

Intuitively, the most important consequence of the Minkowski operator is that if the bodies are apart from each other, the minimum norm vector has norm $\|\boldsymbol{\nu}\,(\text{CSO})\| =$

$d(\Omega_P, \Omega_Q)$. Consider for example the two disjoint spheres Ω_P and Ω_Q in Figure 2.4(a). The CSO resulting from the Minkowski difference $\Omega_P - \Omega_Q$, depicted in Figure 2.4(b), has a unique vector $\boldsymbol{\nu}(\text{CSO})$ whose norm corresponds to the distance between the spheres in physical space. For contacting bodies $d(\Omega_P, \Omega_Q) = 0$ and naturally $\boldsymbol{\nu}(\text{CSO}) = \mathbf{0}$. The example in Figure 2.4(c) shows two overlapping spheres, and the CSO in Figure 2.4(d) includes the null vector.

A number of studies observed that the distance algorithms exploiting the Minkowski difference are particularly efficient. Originated from the early works of Lozano-Perez and collaborators (33, 140, 141), this branch of algorithms is called *simplex-based* and is widely adopted for motion path planning in robotics and other fields (see (1) and references therein). Renowned simplex-based distance algorithm are: the Gilbert–Johnson–Keerthi (GJK) (89), Gilbert–Foo (88), Rabbitz's method (197), Chung–Wang (CW) (45), ContactScope (204), enhanced GJK (38), RGJK (178) and EPA-GJK (225). Another alternative family of fast algorithms, which does not exploit the Minkowski difference, is called *feature-based*; this includes: Common-Plane (CP) (50), Lin–Canny (LC) (132), I-Collide (46), Voronoi-Clip (V-Clip) (154) and SWIFT++ (62). These methods work well in practise, however, not all of them solve well-posed problems.

The high speed of simplex-based and feature-based algorithms is due to the low computational complexity of the evaluation of support mapping. Its cost, in fact, scales linearly with the number of features (e.g. vertices or facets) of the bodies. This is a consequence of the Minkowski operator, which never computer the CSO explicitly. This cost may be reduced even further with incremental algorithms that cache data between subsequent solutions steps. All feature-based algorithms can naturally exploit spatial coherence by tracking the closest features (usually Voronoi region) between two

bodies, but their applicability is strictly limited to bodies discretised by polytopes.

The only method which had its applicability extended beyond distance queries and beyond polytopes is the GJK algorithm. This procedure may be applied to quadrics (178), AABBs (225) and NURBS (221); moreover, unlike often reported in the literature, for overlapping bodies this procedure can compute the penetration vector (38, 225).

When compared to other methods, the GJK algorithm has been proven to be one of the fastest procedures. Cameron (37) compared his enhanced GJK with the Lin and Canny (LC) algorithm (132), showing superiority of the former. More recently, two studies (154) and (45) concluded that GJK algorithm has comparable computational costs to the Voronoi-clip (V-Clip) and Chung-Wang (CW) algorithms. In fact, incremental versions of GJK have shown almost-constant time complexity. The Fast Common plane (FCP) (168) exploits space coherence but at the same time introduces approximations to the distance computation. More importantly, both CP and FCP store the common plane in a relatively complex data-structure that modern computer architectures would struggle to cache efficiently. The GJK algorithm does not require any of that.

Because of its extreme versatility and high performance, the GJK has been adopted in a wide range of applications and patents. These include, but are not limited to: robotics (58, 169, 257, 258), real-time rendering (105, 122, 214), rigid-body dynamics (90, 199, 217), medical surgery (135), computer graphics (164, 206), CAD (124), physics (91, 152, 157) and, rarely, in computational mechanics (232).

To understand the reasons why the GJK algorithm has not been extensively applied in computational mechanics, the next section introduces its theoretical framework.

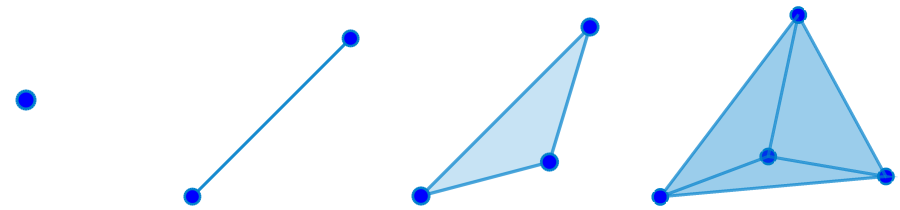


Figure 2.5: In $\mathbb{R}^3$ a simplex can either be a point (a), a line segment (b), a triangle (c) or a tetrahedron (d).

2.4.2 The Gilbert-Johnson-Keerthi (GJK) algorithm

The GJK algorithm computes a pair of witness points by solving the equivalent problem of finding the minimum norm vector of the CSO. This *unique* vector was introduced in Eqs. (2.12) and is denoted by $\boldsymbol{\nu}(\text{CSO})$.

The explicit computation of the entire CSO at runtime is computationally prohibitive but, alike other simplex-based method, the GJK algorithm seeks the solution $\boldsymbol{\nu}(\text{CSO})$ by means of a *simplex*. Let us recall that a m-simplex τ is the convex hull of a set of $m+1$ affinely independent points $\{s_1, ..., s_{m+1}\}$ in $\mathbb{R}^n$, for $m \leq n$. The simplices for three-dimensional problems are: vertices, line segments, triangles and tetrahedrons. Figure 2.5 shows examples of 3D simplices.

For this particular application, the simplex is described by a set of points on the frontier of CSO. This guarantees that the simplex is a subset of CSO. In fact, since the CSO is convex, every convex combination of points of CSO belongs to CSO and therefore $\tau \subset \text{CSO}$.

The GJK procedure seeks $\boldsymbol{\nu}(\text{CSO})$ iteratively. At each k-th iteration, the simplex τ_k is updated to move as close as possible to the origin of the configuration space. When $\boldsymbol{\nu}(\tau_k)$ is sufficiently close to $\boldsymbol{\nu}(\text{CSO})$, the algorithm terminates as the solution to the equivalent problem in Eq. (2.12) is found.

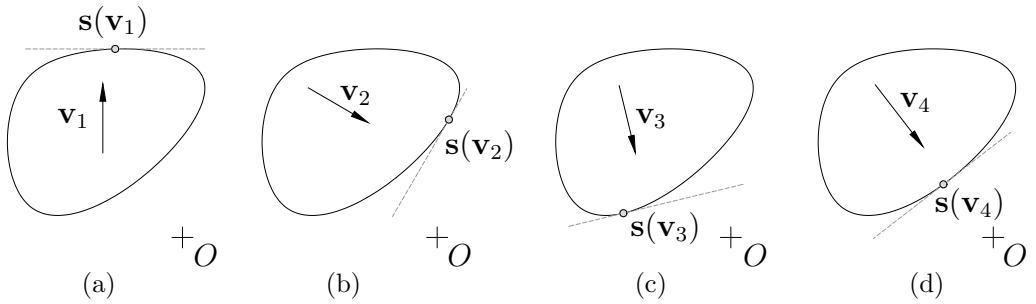


Figure 2.6: Illustration of iterations undertaken by the GJK algorithm for an arbitrary CSO. The first iteration is initialised by the user (a), the second (b) and the third (c) iterations converge toward the point of minimum norm which is reached at the fourth iteration (d).

The GJK algorithm may be interpreted as the following conditional loop:

$$\begin{aligned} &\texttt{while Not close\_enough } (\mathbf{v}_k) \\ &\qquad \mathbf{v}_{k+1} \leftarrow \texttt{refine\_direction } (\mathbf{v}_k) \\ &\texttt{end} \end{aligned} \tag{2.14}$$

where $\mathbf{v}_k = \boldsymbol{\nu}(\tau_k)$ is introduced to emphasise the direction toward which the simplex is moved at the k-th iteration. An illustrative sequence of steps is illustrated in Figure 2.6.

The iterative search descends in the sense that the simplex $\nu(\tau_{k+1})$ offers a better approximation to $\nu(\text{CSO})$ than $\nu(\tau_k)$. This is achieved by removing from τ_k a point that is "far from O", and adding a closer point $w_k \in \text{CSO}$ to form τ_{k+1} such that:

$$\|\boldsymbol{\nu}(\tau_{k+1})\| = \|\boldsymbol{\nu}(\text{conv}(\{\tau_k, w_k\}))\| \leq \|\boldsymbol{\nu}(\tau_k)\|. \tag{2.15}$$

The preceding equation states that a sequence of simplices indeed converges monotonically to $\nu(\text{CSO})$. Furthermore, convergence is achieved in a finite number of steps for polytopes (89).

The point w_k is of vital importance for the GJK algorithm. This is highlighted in Eq. (2.15); in fact, from w_k is defined the new search direction $\mathbf{v}_{k+1}$ and in turn the outcome of the exit `close_enough` condition in Eq. (2.14). The point w_k may be geometrically interpreted as the furthest point from the hyperplane defined by $\mathbf{v}_k$ along the direction $\mathbf{v}_k$, see Figure 2.6. This point, not necessarily unique, lays on the

outer boundary of the CSO and is such that $h_{\text{CSO}}(\mathbf{v}_k) = (w_k - O) \cdot \mathbf{v}_k$. Next, let us present how the point $w_k \in \text{CSO}$ may be computed *without* evaluating the entire CSO explicitly.

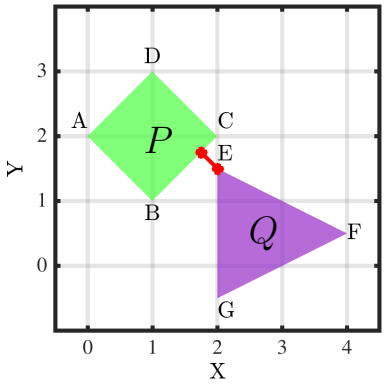
To find the point $w_k \in \text{CSO}$, the GJK algorithm computes the support function on each body independently rather than on the CSO. This allows to avoid the evaluation of the entire CSO, which indeed would be too expensive. Instead, the point w_k on the frontier of the CSO is computed as the support mapping of a compound, see Eq. (2.9).

The literature does not present a demonstration of the fact that $\nu(\text{CSO})$ can effectively be represented by caching few solutions of the support function; the following demonstration fills this gap. The GJK algorithm terminates when $\nu(\tau_k) = \nu(\text{CSO})$. Consider a pair of witness points z_P and z_Q and a m-simplex being expressed by the vertices $\tau = \{s_1, ..., s_{m+1}\}$, where each vertex is given by $s_i = p_i - q_i$, with $p_i \in P, q_i \in Q$. For a set of positive scalars λ_i, the solution $\nu(\tau_k)$ writes:

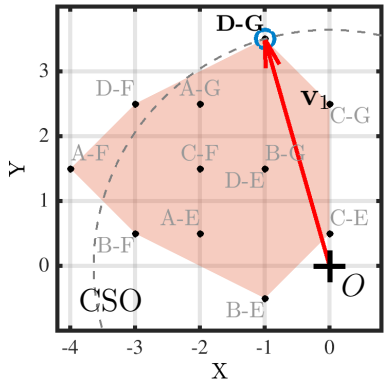
$$\begin{aligned}
\nu(\tau_k) &= \sum_{i=1}^{m+1} \lambda_i s_i = \sum_{i=1}^{m+1} \lambda_i (O + \mathbf{s_i}) = \sum_{i=1}^{m+1} \lambda_i (O + \mathbf{q_i p_i}) \\
&= \sum_{i=1}^{m+1} \lambda_i (O + \mathbf{O p_i} + \mathbf{q_i O}) = \sum_{i=1}^{m+1} \lambda_i p_i + \sum_{i=1}^{m+1} \lambda_i \mathbf{q_i O} \\
&= z_P + \sum_{i=1}^{m+1} \lambda_i \mathbf{q_i O} = \mathbf{z_Q z_P} + z_Q + \sum_{i=1}^{m+1} \lambda_i \mathbf{q_i O} \\
&= \mathbf{z_Q z_P} + \sum_{i=1}^{m+1} \lambda_i (q_i + \mathbf{q_i O}) = \mathbf{z_Q z_P} + O. \\
&= \sum_{i=1}^{m+1} \lambda_i (p_i - q_i) + O. \qquad \square
\end{aligned} \tag{2.16}$$

Therefore, at the most $m + 1$ solutions of the support function need to be stored in memory to describe $\nu(\tau_k)$.

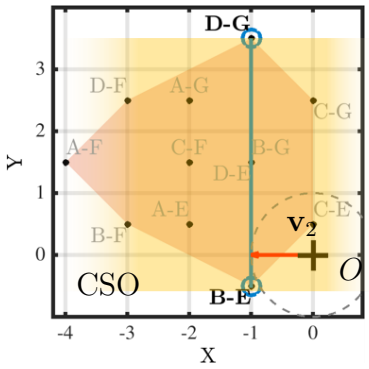
The rest of this section illustrates geometrically how the GJK algorithm solves a distance query. Given the two polygons P and Q in Figure 2.7(a), we wish to compute



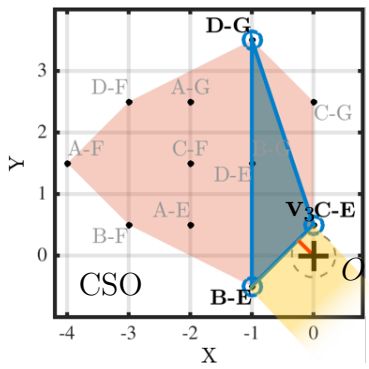
(a) Polygons P, Q and separating axis in red.



(b) Iteration $k = 1$: $\tau_1 = \{\text{D-G}\}$, $W_1 = \tau_1$ and $\nu(\tau_1) = D - G$.



(c) Iteration $k = 2$: $\tau_2 = \{\text{B-E,D-G}\}$, $W_2 = \tau_2$ and $\nu(\tau_2) = \lambda_1 \text{B-E} + \lambda_2 \text{D-G}$.



(d) Iteration $k = 3$: $\tau_3 = \{\text{C-E, B-E, D-G}\}$, $W_3 = \{\text{C-E, B-E}\}$, $\nu(\tau_3) = \lambda_1 \text{C-E} + \lambda_2 \text{B-E}$.

Figure 2.7: Iterative GJK procedure for the pair of distant polygons in (a). At the first iteration (b) an arbitrary search direction $\mathbf{v}_0$ is set downward. At each further k-th iteration (c)-(d) this is recomputed as the point of a simplex τ_k closest to origin lays in the (orange) Voronoi region containing the origin.

the minimum distance $d(P, Q)$, which is the norm of the separating vector depicted in red. Since the bodies are distant, a pair of witness points defining the separating vector exists. The GJK algorithm begins by initialising $W = \tau_0 = \emptyset$ and an arbitrary search direction $\mathbf{v}_0$. In our example, $\mathbf{v}_0$ is set downward, such that the solution to the support mapping along the direction $-\mathbf{v_0}$ is D-G. This identifies a solution vector for the support function, in fact $h_{\text{CSO}}(\mathbf{v}_0) = \mathbf{D\text{-}G} \cdot \mathbf{v}_0$. For the first iteration $k = 1$, the simplex is $\tau_1 = \{\text{D-G}\}$ and $\mathbf{v}_1 = \mathbf{D\text{-}G}$. All quantities involved are illustrated in Figures 2.7(b) (notice that the CSO is shown for illustration purpose only). In

the second iteration B-E is found to be the farthest point of the CSO along $-\mathbf{v}_1$, and is listed first in τ_2. Figure 2.7(c) shows that both vertices support the point of minimum norm $\nu(\tau_2)$, which is then used to update the search direction $\mathbf{v}_2$. A similar situation occurs in the third iteration. As shown in Figure 2.7(d), the subset supporting $\nu(\tau_3)$ is $W = \{$ C-E , B-E $\} \subset \tau_3$. Moreover, because $\nu(\tau_3) = \nu(\text{CSO})$, the GJK algorithm terminates. The reader can observe that the radius of the dashed circles in Figures 2.7(b)–2.7(d), centred at the origin and passing by the point of minimum norm of each simplex, decreases monotonically at every iteration. This is a geometrical interpretation of the fact that the GJK algorithm is a descending method.

2.4.3 Gaps in the literature

A number of studies have reported that the GJK algorithm lacks robustness, and for this reason it is unsuitable for engineering simulations. This is due to the numerical error which can propagate to a new iteration and lead to infinite loops. The erroneous results thus generated can, for example, ruin the user experience in a computer game, produce instabilities in finite element simulations, or lead a real-time collision avoidance systems to failure.

The numerical error is mostly due to the `refine_direction` function in Eq. (2.14), which implements the so-called *distance sub-algorithm* due to Johnson's (106). Let us recall that this function is vital to the GJK algorithm since it defines the search direction, updates the simplex and passes the input for the exit tests. For these reasons, the *Backup procedure* was originally added to handle pathological cases at the expense of CPU time and implementation effort (89).

To improve the performance of the GJK algorithm, Van den Bergen (224) replaced

the Backup procedure with a set of more robust exit conditions. These can establish whether: (a) the two bodies are in contact, or (b) the simplex τ_k cannot move closer to the origin. The former is true if the simplex contains the origin O, or if:

$$\|\mathbf{v}_k\|^2 \leq \varepsilon_{tol} \max\{\|y - O\|^2 : y \in W\} \tag{2.17}$$

where W is the smallest subset of vertices in τ_k that supports $\nu(\tau_k)$. The latter is true if $w_k \in \tau_k$, or if:

$$\|\mathbf{v}_k\|^2 - \mathbf{v}_k \cdot \mathbf{w}_k \leq \varepsilon_{rel}^2 \|\mathbf{v}_k\|^2. \tag{2.18}$$

The values ε_{tol} and ε_{rel} set the accuracy of the GJK algorithm. For engineering applications, these can be set in the order of the machine precision.

However, the same author reported in (225) that the main source of numerical instability is the cancellation error within the Johnson's algorithm. The exit conditions introduced in Eqs. (2.17) and (2.18) are indeed better than the original ones, but cannot counterbalance the lack of robustness of Johnson's algorithm. This issue, not addressed by Van der Bergen, can only be resolved by replacing the original sub-algorithm.

The only attempt to replace Johnson's algorithm was formalised, certainly not for the first time, by Ericson (64) and is particularly popular within the computer graphics community (218). Despite being mathematically equivalent and simpler to grasp, the implementation of this method is particularly involved as it requires a cascade of conditional statements difficult to debug. More importantly, this method is computationally more expensive.

To conclude, as recently reported in (97, 173), the GJK algorithm still lacks robustness, an essential requirement without which it cannot be applied in computational mechanics.

2.5 Distance methods for primitives

A primitive is the simplest geometrical entity contributing to the description of a whole solid body. For the particular applications in computational mechanics, examples of primitives are: spheres, ellipses, line segments, NURBS curves, triangles, and tetrahedrons. Computing the distance between primitives is extremely common in all numerical methods used in engineering and, to facilitate the coupling between different methods, it is desirable to design a versatile algorithm that handles those queries which do not allow a closed-form solution. Furthermore, accuracy and robustness are major concerns of such algorithms. This section reviews existing methods and provides background notions for understanding those issues that prevent from solving distance queries accurately.

2.5.1 Direct methods

Apart from few particular cases, such as sphere–sphere queries, computing the distance between primitives is difficult and expensive. The coupling of DEM and FEM is, from this point of view, notoriously challenging. Triangles, squares, tetrahedrons and hexahedrons are the most common shapes in computational mechanics, but also at the most difficult ones to deal with.

The algorithms designed to solve distance queries between primitives in a single shot are known as *direct methods*; that is, a method that attempts to find a solution without iterations. These methods combine a series of conditional statements to solve those distance queries that do not admit closed-form solution.

The literature presents a large number of direct methods to carry out tests between

specific pairs of primitives. Triangle–triangle queries may be computed by means of potential method (160), an approach that has been successfully applied to 2D and 3D polydisperse triangular elements (249). Similar problems have been addressed by formulating the distance query as an optimisation problem (26, 27, 41). Another specific type of query, driven by the DEM in particular, is the triangle–sphere test. This may be solved using a simple geometric method (64), but higher performances may be achieved with a recently proposed method based on barycentric coordinates (100). Similarly, the distance between quadrilateral facets and spheres may be computed with specific methods described in (43, 114).

Having procedures that are specialised for only one type of distance query increases the implementation effort, and do not guarantee an improvement of performance. Because of the high volume of calls to functions which test primitives, these routines need to be highly optimised, but the optimisation of many routines is a tedious and time-consuming work. Also, the data must be stored in such a way to exploit the caching of modern processor architectures. From a software development point of view, designing a single function that deals with more than a single query would make the optimisation easier and more effective.

Consequently, a number of algorithms for solving distance queries between different primitives have been developed. For instance, the literature presents many algorithms that solve both node–edge or node–facet interaction. These belong to the family of algorithms that solve *point–simplex* problems.

The general case of point–simplex problem is considered, that is: a distance query between a point and any of the simplices defined in 3D space. This choice is motivated by the number of applications in which a particular case of the point–simplex problem

plays an important role. Some of these applications are:

- Contact problems based on the FEM master-slave formulation (246).
- Specific distance queries of edge–sphere, triangle–sphere and tetrahedron–sphere which are commonly encountered in DEM/FEM problems.
- Distance queries such as quadrilateral elements and spherical particles by subdividing the former in triangular facets, therefore expanding the capability of other existing methods, such as (117, 147, 235, 236).

Formally, such problem consists of calculating the minimum distance between a query point and all faces of a simplex. An m-simplex has $(2^{m+1} - 1)$ *faces* (e.g. vertices, edges, facets, volume) and the solution essentially is, in the most general case, the point of a tetrahedron that is closer to the query point than any other point of the tetrahedron.

Because of its wide applicably, the point-simplex problem is extensively studied in the literature. One of the most renown procedure in the computational mechanics community is the *inside–outside* algorithm (238). Its original version has shown very low computing cost, but it failed to detect some intersection and has been consequently fixed in (202). Other methods in the literature include (26, 63, 104, 156, 168, 173, 256); neither of these studies, however, provides arguments supporting the claims on the robustness of the methods. In particular, when computing the normal vector for projecting a point on a plane, the cancellation error leads to degenerate cases which are not addressed in none of these studies. Another example is the solution of point–projection problems. For all these cases, degenerate geometries cause failures unless handled with care.

General algorithms that handle degenerate cases exist (61, 96, 155, 215), but none of

these exploits spatial coherence to accelerate distance queries. Essentially, for a general point–tetrahedron query, these methods examine all 15 subsets (4 vertices, 6 edges, 4 faces and the interior) of the tetrahedron and test which one of these contains the closest point to the query point. There are cases where, by virtue of spatial coherence, cached data can be reused to eliminate *a priori* some of these subsets.

As highlighted in Section 2.3, spatial coherence allows to save computing time for problems which are solved incrementally; *recursive methods* make use of coherence to accelerate point–simplex distance queries. One of these methods was presented in (138), another one is Johnson's algorithm (106). The former was designed for geodesic applications and implements a cascade of conditional statements which is unlike to give high performance on dynamic simulations. The latter, instead, seems suitable and will therefore be examined in the next section.

2.5.2 Johnson's recursive algorithm

A recursive algorithm allows to reuse cached data at subsequent calls to accelerate the solution of distance queries. The difference with respect to iterative procedures is that recursive methods can invoke themselves to solve a distance query, and indeed they may be used to speed-up iterative procedures.

Johnson's algorithm (106) solves point–simplex distance queries recursively and is particularly efficient because it exploits spatial coherence. Another advantage is that its implementation is concise and straightforward.

Let us consider the problem of computing the minimum distance $d(O, \tau)$ between the origin O and a simplex τ. Johnson's algorithm seeks the solution by computing the (unique) point $\nu(\tau)$ closest to the origin than any other point of τ.

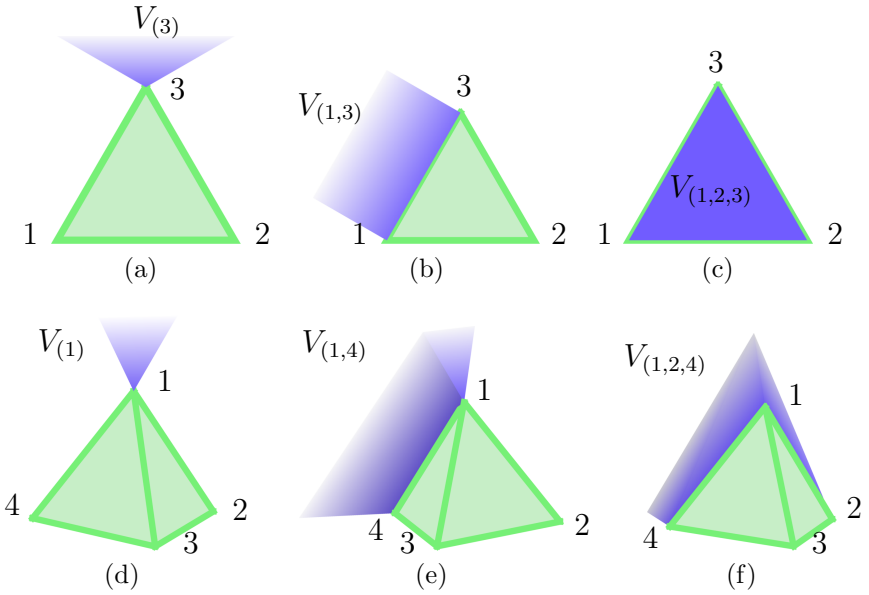


Figure 2.8: For each m-simplex, $(2^{m+1}-1)$ Voronoi regions may be defined. The following regions of a 2-simplex are highlighted: (a) $V_{(3)}$ of the vertex 3, (b) $V_{(1,3)}$ of the edge $\{1,3\}$ and, (c) $V_{(1,2,3)}$ of the facet $\{1,2,3\}$. For a 3-simplex are highlighted: (d) $V_{(1)}$ of the vertex 1, (e) $V_{(1,4)}$ of the edge $\{1,4\}$ and, (f) $V_{(1,2,4)}$ of the facet $\{1,2,4\}$.

The idea is to look for $\nu(\tau)$ in all *Voronoi regions* associated to τ. Recall that a Voronoi region is defined as the set of points which are at least as close to a point of the face as to any other point of τ not in the face. For example, the region associated to one of the vertices of a two-dimensional simplex is represented in Figure 2.8(a). Other examples of regions are shown in Figures 2.8(b) and 2.8(c) for a planar simplex, whereas Figures 2.8(d)-2.8(f) show examples in $\mathbb{R}^3$. The Voronoi region associated to a face of τ is denoted by V_κ, where κ is an ordered and non-empty tuple which lists the indices i of the set $\{s_i\}$ defining the face V_κ is associated to.

An intuitive illustration of the way Johnson's algorithm inspects the Voronoi regions is shown in Figure 2.9 for a 2-simplex. The bottom-up arrows indicate that the method begins the search by inspecting the Voronoi regions associated to the vertices of the simplex. If $\nu(\tau)$ cannot be found in either of these three, the search recursively passes to the next levels, where edges and facet are inspected until the solution is found.

The rest of this section presents technical details about the original Johnson's al-

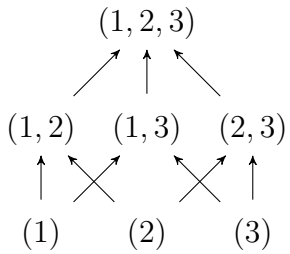


Figure 2.9: Johnson's algorithm conducts a bottom-up search across all Voronoi regions of a 2-simplex. This representation is also known as Hasse diagram.

gorithm as formulated in (89). The algorithm expresses the point $\nu(\tau)$ as a convex combination of the smallest set of vertices defining τ; this requires to compute barycentric coordinates and the smallest subset $W \subseteq \tau$. The subset W identifies, with a tuple κ, the face of an m-simplex supporting $\nu(\tau)$. It has been shown in (89) that a point corresponds to $\nu(\tau)$ if and only if its barycentric coordinates λ_j satisfy:

$$\lambda_j > 0 \;\text{ and }\; \lambda_i \leq 0 \;\; \forall \;\; j \;\in\; \kappa \;:\; i = 1, ..., m+1,\; i \neq j. \tag{2.19}$$

In the preceding equation λ_j and λ_i correspond to the barycentric coordinates for the simple τ and its subsets, respectively. Effectively the solution of a projection problem, Eq. (2.19) guarantees that $\nu(\text{aff}(W)) = \nu(\text{conv}(\tau))$ (224), which geometrically means that the vector $\boldsymbol{\nu}(\tau)$ is perpendicular to a face of the simplex. This face is "optimal" in the sense that it minimises, by virtue of the perpendicular condition, the distance between the origin O and the simplex. Johnson's algorithm recursively inspects all the Voronoi regions of a m-simplex until the signs of the barycentric coordinates comply with Eq. (2.19).

Behind the computational efficiency of Johnson's algorithm stands a recursive solution to the algebraic system that evaluates the barycentric coordinates in Eq. (2.19). Let a simplex $\tau = \{s_i\}$, with $i \in I = \{1, ..., m+1\}$, have its point of minimum norm $\nu(\tau)$ laying on a r-face defined by the points in $\{s_j\}$, with $j \in \kappa$. This face is per-

pendicular to a vector $\boldsymbol{\nu}(\tau)$ or, equivalently, $(\mathbf{s}_j - \mathbf{s}_l) \cdot \boldsymbol{\nu}(\tau) = 0$, for all $j \neq l$, where l is an arbitrary element of κ. The vector $\boldsymbol{\nu}(\tau) = \nu(\tau) - O$ is written in terms of $r+1$ barycentric coordinates λ_j to assemble an algebraic system $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$ which writes (89):

$$\begin{bmatrix} 1 & ... & 1 \\ (\mathbf{s}_2 - \mathbf{s}_l) \cdot \mathbf{s}_1 & ... & (\mathbf{s}_2 - \mathbf{s}_l) \cdot \mathbf{s}_r \\ ... & ... & ... \\ (\mathbf{s}_j - \mathbf{s}_l) \cdot \mathbf{s}_1 & ... & (\mathbf{s}_j - \mathbf{s}_l) \cdot \mathbf{s}_r \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ ... \\ \lambda_r \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ ... \\ 0 \end{bmatrix} \tag{2.20}$$

The index $l \in \kappa$ must be kept constant, usually $l = 1$, and j takes the remaining r values in κ. Johnson's algorithm recursively solves Eq. (2.20) for all subsets of τ. Because κ has cardinality at the most 4, the system $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$ can be efficiently solved by using Cramer's rule.

A solution of the algebraic system in Eq. (2.20) can be written by combining Cramer's rule with a cofactor expansion as follows:

$$\lambda_j = \frac{-1^{1+j} \det \mathbf{A}_{1j}}{\det \mathbf{A}} \tag{2.21}$$

where $\mathbf{A}_{1j}$ are minors of $\mathbf{A}$ obtained by removing the first row and the j-th column from $\mathbf{A}$. To understand the combinatorial logic of Johnson's algorithm, let us look at $\mathbf{A}$ as the minor of another matrix built in the same fashion but for a larger subset. This larger subset is obtained by adding to W a vertex of the simplex s_i not yet included in W, namely $\{s_j : j \in \kappa\} \cup s_i$ for an $i \in I - \kappa$. This leads to a recursive solution for $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$:

$$\lambda_j = \frac{\Delta_j(W)}{\sum_{j \in \kappa} \Delta_j(W)}. \tag{2.22}$$

Where $\Delta_j(W)$ is a cofactor of $\mathbf{A}$ for one of the $(2^{m+1} - 1)$ subsets W of τ. Since the

vertices of τ are linearly independent:

$$\Delta_j(W) = -1^{1+j} \det \mathbf{A}_{1j}. \tag{2.23}$$

Johnson's algorithm computes the values $\Delta_j(W)$ in order of increasing cardinality of W. For the first $m+1$ singletons, the solution is trivial:

$$\Delta_j(\{s_j\}) = 1 \quad j \in I, \tag{2.24}$$

whereas for the remaining subsets:

$$\Delta_i(W \cup s_i) = \sum_{j \in \kappa} \Delta_j(W)(\mathbf{s}_j \cdot \mathbf{s}_l - \mathbf{s}_j \cdot \mathbf{s}_i) \quad \text{for} \quad i \in I - \kappa. \tag{2.25}$$

The equation above is tested for all subsets of τ, keeping $l \in \kappa$ constant, until Eq. (2.19) is verified. By transferring the constraints of Eq. (2.19) on Δ_j, it can be shown (89) that the barycentric coordinates for the vertices of a subset $W = \{s_j : j \in \kappa\}$ comply with Eq. (2.19) if: (i) $\Delta_j(W) > 0$ and (ii) $\Delta_i(W \cup s_i) \leq 0$ for all i in the complement of κ in $I = \{1, ..., m+1\}$. If all barycentric coordinates of the vertices in W are strictly positive, then all the subsets of cardinality $|W|+1$ are inspected. If these have non-positive barycentric coordinates, the algorithm terminates (89).

A speed-up can be achieved for those problem in which the simplex is built vertex-by-vertex by caching the values of $\Delta_i(W)$. These values may be reused at a subsequent call of Johnson's algorithm and thus reduce the number of operations required to evaluate Eq. (2.22).

2.5.3 Gaps in the literature

The cancellation error can induce distance algorithms to fail, but the computational mechanics community overlooks this aspect to a great degree. Particularly when it comes to point projection and point inversion problems, imposing the orthogonal con-

dition by means of a normal vector is an operation particularly prune to rounding error (112, 113, 176).

A close inspection shows that the majority of existing methods does not handle naturally degenerate geometries. Alike many other, Johnson's algorithm is prune to cancellation error and it therefore lacks robustness. If the set of vertices in Eq. (2.20) is affinely dependent, or nearly so, the algebraic system is ill-condition and cannot be solve accurately on finite-precision machines. This shortcoming was observed in various studies (89, 225), but it has never been addressed. Overall, despite its numerical instabilities, the Johnson's algorithm is still an extremely efficient method. Addressing its lack of robustness would improve point–simplex distance queries in the fields of robotics, computer graphics and engineering.

A brute-force search on all Voronoi regions can be robustly implemented and can even exploit spatial coherence, but its performance is poor. One can conclude that the procedure for closest point to tetrahedron presented in (64) is slow by the cascade of conditional statements required, but also speculating on the amount of advanced algorithms published to solve the same problem. However, the literature does not present a study that compares the performance of this brute-force search with other methods. It is therefore not known what gain advanced procedures could bring to the solution of point–simplex distance queries.

2.6 Concluding remarks

This chapter has highlighted the importance of distance algorithms and has reviewed the literature beyond the field of computational mechanics.

The following needs and open questions have been identified:

1. Need for versatile algorithms that handle different representations of solid bodies (e.g. quadrics, NURBS, polytopes),
2. Limited understanding on the robustness of certain procedures published to date and how this affects the overall performance of the algorithm,
3. Spatial coherence is rarely exploited in computational mechanics,
4. Arbitrary bodies (i.e. non-convex and with irregular morphology) are best treated by hierarchical algorithms, but existing methods rely entirely on extrinsic space-partitioning schemes and the majority treats only spheres,
5. The GJK algorithm has been rarely applied to computational mechanics because of its lack of robustness,
6. The lack of robustness of the GJK algorithms has been reported but never addressed,
7. Need for a robust point–simplex distance algorithm that exploits spatial coherence,
8. Few studies support claims on the robustness with in-depth analysis of degenerate simplices arising, for example, in point projection problems.

Altogether, these gaps have identified the limitations that make distance queries one of the major computational bottleneck in engineering simulations.

Chapter 3

Novel hierarchical framework

The literature review concluded that existing distance algorithms are a major computational bottleneck and that further development is required to address the need for more versatile and faster distance algorithms. This chapter introduces three new procedures for the solution of distance queries: the first one is a robust and recursive method for point–simplex tests, the second solves distance queries between arbitrary convex objects accurately to machine precision, and the third algorithm is specifically designed for large simulations involving concave bodies. These are presented independently because they can work independently, however they can also be combined into a hierarchical framework to solve complex distance queries.

3.1 New recursive method for basic primitives

The first novelty presented in this chapter is the *Signed Volumes method*: a procedure for computing the minimum distance between a point and a simplex. The aim is to supply a method which results more robust, and as fast as Johnson's algorithm. This section presents the formulation and implementation of this method.

3.1.1 Overview of the Signed Volumes method

To solve a point–simplex distance query, one needs to find the *unique* point of the simplex that is closer to the query point than any other point of the simplex. The solution is expressed, for the practical reasons described in Section 2.5.2, as a convex combination of the smallest subset of simplex vertices whose barycentric coordinates are strictly positive. Therefore, the unknowns are the minimum subset of vertices and the corresponding barycentric coordinates that define the closest point, see Eq. (2.16).

Johnson's algorithm (106) is an efficient method for this type of queries but it lacks accuracy and robustness when dealing with degenerate geometries. The reasons for that are rooted in the coefficient matrix $\mathbf{A}$ of the algebraic system in Eq. (2.20):

1. The entries of $\mathbf{A}$ involve arithmetic operations that cannot be computed accurately due to cancellation error; and
2. $\mathbf{A}$ is rank deficient, therefore the algebraic system does not admit unique solution.

The novel Signed Volumes method addresses these issues by: moving any arithmetic operation to the right-hand side of the algebraic system, and by removing affinely dependent vertices before assembling the coefficient matrix. The idea is therefore to design a new method that assembles a coefficient matrix whose entries are as simple

as possible and does not involve arithmetic operations, whereas degenerate simplices are handled by removing those affinely dependent points. Altogether this eliminates inaccuracies due to cancellation error and guarantees that the algebraic system is solved accurately. Further details on the formulation of the Signed Volumes method and its implementation are presented in the following sections.

3.1.2 Theoretical algorithm

The newly proposed procedure is named the Signed Volumes method because it evaluates the volume form $\mu(\tau)$ of a m-simplex τ and of other $m+1$ *fictitious simplices* associated to it. Let us first recall the notion of volume form, introduce the fictitious simplices and finally present the method itself.

The volume form is more commonly known as mixed product, and is defined as the dot product of a vector with the result of a cross product between two other vectors. The terminology volume form is here preferred to emphasise the geometrical interpretation of $\mu(\tau)$, which is the volume of some geometric objects (82). More specifically, $\mu(\tau)$ is proportional to the signed measure of length, area or volume of the simplices in Figure 2.5. The most relevant properties of the volume form are: (i) it changes sign if the orientation of a vector is changed, and (ii) it is null for linearly dependent vectors (82).

A fictitious simplex is obtained by substituting the origin O to a vertex of τ. Since τ has $m+1$ vertices, $m+1$ fictitious simplices are defined. Notice that, because of property (i) of the volume form, these must be built with care to guarantee consistency in the sign of the volume forms. Figure 3.1 shows an example how to correctly operate on a 2-simplex in $\mathbb{R}^2$. If the triangle includes the origin O, its three fictitious simplices

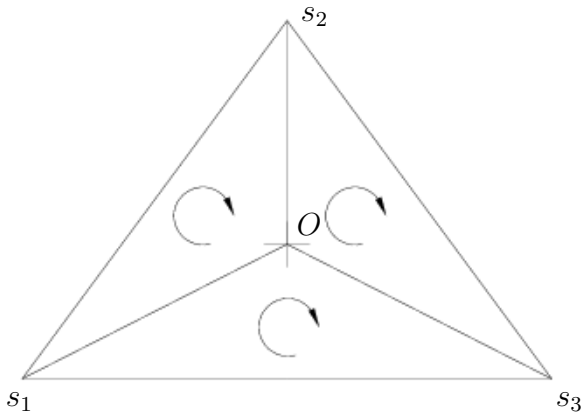


Figure 3.1: From a simplex $\tau = \{s_1, s_2, s_3\}$, three fictitious simplices $\{O, s_2, s_3\}$, $\{s_1, O, s_3\}$ and $\{s_1, s_2, O\}$ are defined. Notice that, for this example, their volume forms have the same signs as illustrated by the three arrows.

must be a permutation following the right-hand rule. This is illustrated by the arrows inside each fictitious simplex in Figure 3.1.

The Signed Volumes method aims to solve a distance query between the origin O and a simplex τ by computing the point of minimum norm $\nu(\tau)$. This is expressed as a convex combination of the smallest subset $W \subseteq \tau$ of vertices supporting $\nu(\tau)$ and a set of barycentric coordinates associated to each vertex.

The theoretical Signed Volumes method takes three steps:

1. Project, if possible, the vertices of τ into a lower dimensional space, thus descending from $\mathbb{R}^n$ to a reduced space $\mathbb{R}^r$, with $r < n$.
2. Discard the vertices, if any, not supporting $\nu(\tau)$ to identify the smallest subset W. This obviously includes discarding affinely dependent vertices.
3. Solve an algebraic system of equations $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$ to compute the barycentric coordinates $\boldsymbol{\lambda}$.

The aim of Step 1 is to lead the search of W (carried out in Step 2) toward a space in which is "safer" to compute the barycentric coordinates (carried out in Step 3). Effectively a preprocessing effort, Step 1 enables the Signed Volumes method

to execute subsequent steps accurately. The projection relies on the fact that the barycentric coordinates are invariant to affine transformations (48), thus allowing to compute $\boldsymbol{\lambda}$ in a lower-dimensional space $\mathbb{R}^r$, and to use these values in the space $\mathbb{R}^n$ where the simplex reside.

The smallest dimension r of the reduced space $\mathbb{R}^r$ is sought recursively though one or more projection steps. For a m-simplex τ, the algorithm takes $r = m$ at first and tests, in Step 2, whether $W \equiv \tau$. If this is not the case, a $(m-1)$-simplex τ^r is projected onto a lower dimension. Since the barycentric coordinates are invariant to projection, $\nu(\tau^r) = \nu(\tau)$, and a theorem due to Carathéodory (240) establishes that $\nu(\tau^r)$ can be expressed as a convex combination of $r+1$ or fewer points of τ^r (240). Therefore, the point $\nu(\tau)$, laying on a r-face of τ, may be expressed as:

$$\nu(\tau) = \sum_{i \in I} \lambda_i s_i \;:\; \lambda_i \geq 0,\; \sum_{i=1}^{n+1} \lambda_i = 1 \tag{3.1a}$$

$$\nu(\tau) = \sum_{j \in \kappa} \lambda_j s_j \;:\; \lambda_j > 0,\; \sum_{i=1}^{r+1} \lambda_j = 1 \tag{3.1b}$$

where $I = \{1, ..., n+1\}$ and $\kappa \subseteq I$, for $r \leq n$.

Unlike Johnson's algorithm, the Signed Volumes method verifies that all barycentric coordinates are strictly positive by means of Eq (3.1b). Let us recall from Chapter 2 that Johnson's algorithm, instead, solves a system of equations for each subset of τ, and only then it tests whether Eq. (2.19) is verified to ensure that all λ_j's are greater than zero. The Signed Volumes automatically guarantees that λ_j will be positive since it first finds the minimum subset W expressed by κ in Eq. (3.1b), and afterwards it solves $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$ in Step 3.

Step 2 is based on the observation that it exists a relationship between the sign of the volume forms of the simplex τ, of its $m+1$ fictitious simplices and of the unknown

subset W supporting $\nu(\tau)$. For example, all these coincide in the simplex in Figure 3.1 because $W \equiv \tau$. The Signed Volume method compares the sign of $\nu(\tau)$ and the signs of its fictitious simplices, if these coincide ($W \equiv \tau$), otherwise a vertex is discarded. A vertex s_j can be discarded from a m-simplex if it resides in the opposite side of the hyperplane defined by the other s_k points, with $k \in \{1, ..., m+1\} - \{j\}$. This step also allows to remove affinely depend points from the simplex.

Step 3 computes the barycentric coordinates and is executed only when the smallest subset W of τ supporting $\nu(\tau)$ has been projected onto the lowest-dimensional space $\mathbb{R}^r$. This is achieved by a recursive execution of Steps 1 and 2.

The barycentric coordinates are computed from an algebraic system whose coefficient matrix does not involve arithmetic operations. The Signed Volumes method assembles an elementary system of $r+1$ equations based on Eq. (3.1b) only: $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$. In extended form this writes:

$$\begin{bmatrix} s_1^1 & ... & s_{r+1}^1 \\ ... & & ... \\ s_1^l & ... & s_{r+1}^l \\ 1 & ... & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ ... \\ \lambda_{r+1} \end{bmatrix} = \begin{bmatrix} p^1 \\ ... \\ p^l \\ 1 \end{bmatrix} \tag{3.2}$$

The first rows of $\mathbf{M}$ contain the l-th coordinate of s_j and the last one consists of the $(r+1)$-dimensional unit vector. The column vector $\mathbf{p}$ contains the coordinates of the point obtained from the projection of the origin O onto the affine-hull of the points s_j, followed by 1. Since $r \leq 3$, the system involves no more than four equations and a solution can be efficiently computed using Cramer's rule.

The simple construction of the matrix $\mathbf{M}$ in Eq. (3.2) plays a pivotal role in the development of a robust procedure: this only enforces the partition of unity on the values λ_j's and not the (orthogonal) condition of minimum distance between simplex

and origin. The Signed Volumes method is robust because it transfers the condition of minimum distance to a place that does not compromise the solution of the equation system, namely: the right-hand side of Eq. (3.2).

The increased robustness provided by the Signed Volumes method over the Johnson's algorithm is due to the different enforcement of the condition of minimum distance (Eq. (2.12)). The former embeds it directly into the coefficient matrix (Eq. (2.20)), whilst the latter relies on the simple matrix $\mathbf{M}$, whose determinant is proportional to the volume form of the subset W:

$$\det \mathbf{M} = r!\,\mu(W). \tag{3.3}$$

This can only be null if the points in W are affinely independent, but this is not possible because of Steps 1 and 2. Therefore, the system $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$ cannot be ill-conditioned.

Finally, Figure 3.2 illustrates how the Signed Volumes method seeks $\nu(\tau)$ for a 1-simplex, a 2-simplex and a 3-simplex. The top-down arrows indicate that the method begins the search by looking inside the simplex. If $\nu(\tau)$ is not supported by all vertices, a vertex is discarded and the search descends down to a tuple of lower cardinality. Notice that Figure 3.2(b) is opposite to what Johnson's algorithm does (see Figure 2.9).

3.1.3 Implementation

The Signed Volumes method has a simple geometrical interpretation that makes its implementation straightforward.

Algorithm 1 shows the main program from which three functions `S3D`, `S2D` and `S1D` may be called. Apart from the trivial case that consists of a 0-simplex, each function is specific to a m-simplex and follows the three steps described in the previous section.

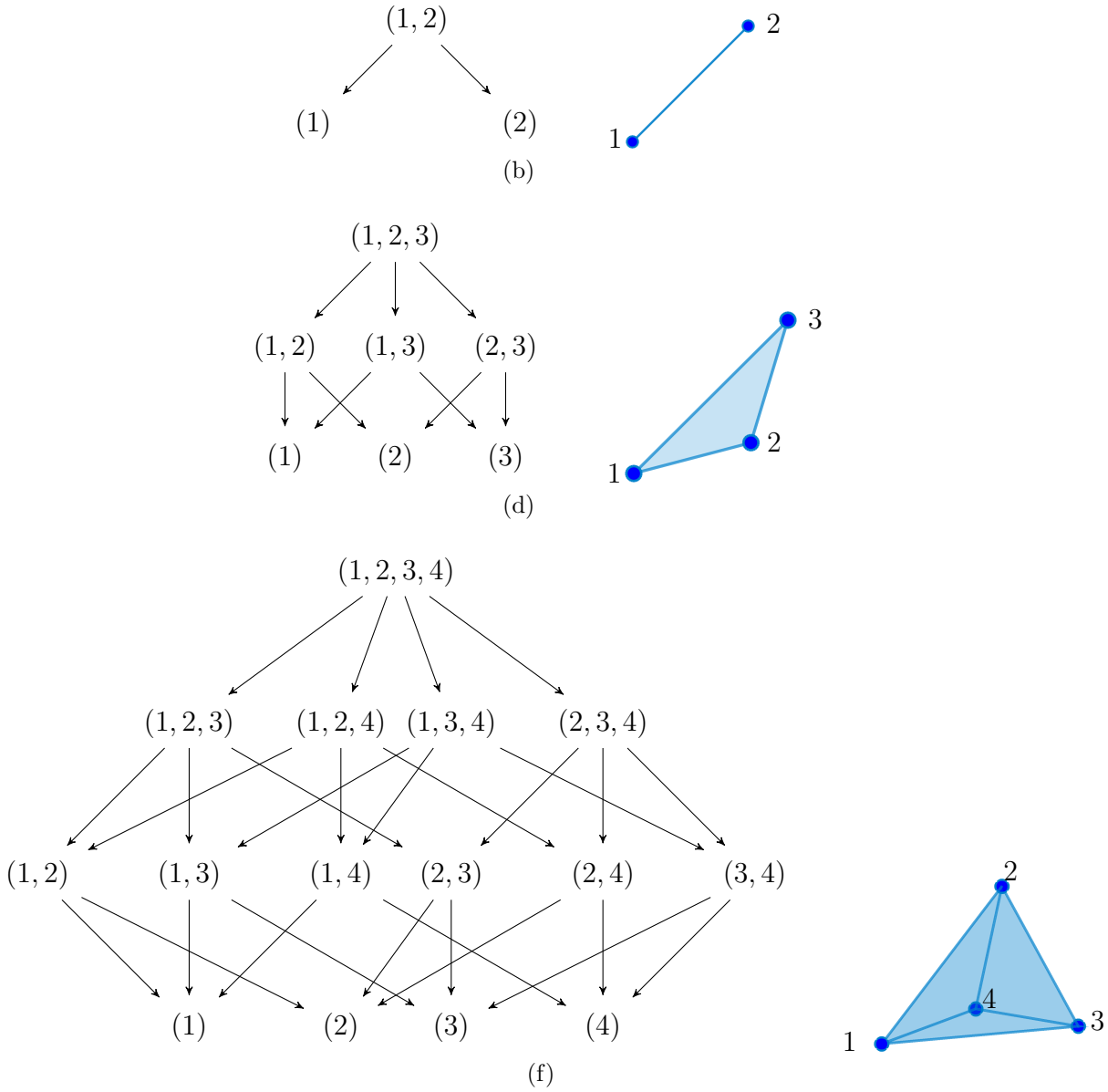


Figure 3.2: Illustration of how the Signed Volumes method conducts a recursive search on a line segment or 1-simplices (a), triangle or 2-simplices (b) and tetrahedron or 3-simplices (c). The numbers specify the tuples that identify all faces of a simplex.

Algorithm 1 Signed Volumes distance sub-algorithm

```
1: procedure SignedVolumes ( τ )
2:     τ has r + 1 vertices
3:     if r = 3 then
4:         [W, λ] = S3D( τ )
5:     else if r = 2 then
6:         [W, λ] = S2D( τ )
7:     else if r = 1 then
8:         [W, λ] = S1D( τ )
9:     else
10:        λ = 1 , W = τ
       return W, λ
```

Recall that Step 2 discards a vertex not supporting the point of minimum norm; from a numerical point of view, this is the most challenging steps since it requires an accurate computation of Eq.(3.3).

A vertex not supporting $\nu(\tau)$ is found by comparing the signs of $\mu(\tau)$ and of other $m+1$ fictitious simplices. The comparison is carried out by the following function:

$$\texttt{CompareSigns}(\, a\,, b\,) = \begin{cases} 1, & \text{if } a > 0, b > 0 \\ 1, & \text{if } a < 0, b < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

It should be highlighted that Eq. (3.4) takes into accounts null and NaN values, arising either from degenerate simplices or rounding error.

The functions `S3D`, `S2D` and `S1D` invoked by Algorithm 1 are described below.

Function `S3D` This function computes the point of minimum norm $\nu(\tau)$ of a tetrahedron, i.e. a 3-simplex, by taking only two of the three steps described in Section 3.1.2. In the first one, Step 2, the procedure tries to discard a vertex from τ, afterwards it solves $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$. `S3D` receives in input an (ordered) list of vertices $\{s_i\}$,

for $i \in I = \{1, ..., m+1\}$, and outputs the subset of points W with the barycentric coordinates that express $\nu(\tau)$ as in Eq. (3.1b).

Step 2 decides whether or not to discard a vertex by comparing the signs of the volume form $\mu(\tau)$ and the other $m+1$ simplices. However, a number of operations may be saved by observing, from Eq. (3.3), that $\mu(\tau)$ and $\det \mathbf{M}$ have the same sign: rather than computing explicitly the volume form of the simplices, the sign of $\det \mathbf{M}$ may be used. From Eq. (3.2) $\mathbf{M}$, writes:

$$\mathbf{M} = \begin{bmatrix} s_1^x & s_2^x & s_3^x & s_4^x \\ s_1^y & s_2^y & s_3^y & s_4^y \\ s_1^z & s_2^z & s_3^z & s_4^z \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.5}$$

and its determinant is given by the following cofactor expansion:

$$\det \mathbf{M} = \sum_{\substack{i=4 \\ j=1,...,4}} (-1)^{i+j} M_{i,j} = C_{4,1} + C_{4,2} + C_{4,3} + C_{4,4} \tag{3.6}$$

In the preceding equation, $C_{i,j}$ is a *cofactor* and $M_{i,j}$ is a *first minor* of $\mathbf{M}$: the determinant of matrix obtained by removing the i-th row and j-th column from $\mathbf{M}$. It is easy to show that the volume form of all fictitious simplices is proportional to the cofactors $C_{4,j}$ and that, from Eq. (3.3), their signs coincide.

The pseudo-code in Algorithm 2 shows an implementation of `S3D` that solves the system in Eq. (3.2) with Cramer's rule. Firstly, all terms of Eq. (3.6) are computed, and their signs compared with `CompareSigns` to test whether a vertex can be discarded or not. If all signs are equal, the origin lays inside the simplex and thus all four vertices are supporting the point of minimum norm. In this case, the barycentric coordinates are given by: $\lambda_j = C_{4,j} / \det \mathbf{M}$ for all j. Otherwise, the function `CompareSigns` has to be called for each $C_{4,j}$, with $j = \{2, 3, 4\}$, and the j-th vertex can be discarded if

the output is 0. The computation of the barycentric coordinates is then carried out by the function `S2D`. However, if two or more fictitious simplices have sign different to the sign of $\det \mathbf{M}$, `S2D` is invoked more than once and its outcomes must be compared to find which is the smallest subset W for which $\|\boldsymbol{\nu}(W)\|$ is minimum.

Algorithm 2 Sub-routine for 3-simplex

procedure `S3D`($\{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4\}$)
  for $j = 1:4$ **do**
    $C_{4,j} = -1^{j+4} M_{4,j}$
    $\det(\mathbf{M}) = \det(\mathbf{M}) + C_{4,j}$
  if `CompareSigns`($\det(\mathbf{M})$, $C_{4,j}$) for all j **then**
    $\lambda_j = C_{4,j} \,/\, \det(\mathbf{M})$
    $W = \{\mathbf{s_1},\ \mathbf{s_2},\ \mathbf{s_3},\ \mathbf{s_4}\}$
  else
    for $j = 1:4$ **do**
      if `CompareSigns`($\det(\mathbf{M})$, $-C_{4,j}$) **then**
        `[`W^*`,` $\boldsymbol{\lambda}^*$`] = S2D(`$\{\mathbf{s}_i : i \in \{1,2,3,4\} - j\}$`)`
        $d^* = \| \sum_{i \in W^*} \lambda_i^* \mathbf{s}_i \|$
        if $d^* < d$ **then**
          $W = W^*$
          $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$
          $d = d^*$
  return W, $\boldsymbol{\lambda}$

Function `S2D` This function computes the point of minimum norm $\nu(\tau)$ of a 2-simplex by taking all steps described in Section 3.1.2. It receives in input three vertices $\{s_j\}$, with $j = \{1, 2, 3\}$, to express $\nu(\tau)$ as convex combination of the smallest subset $W \subset \tau$ as in Eq. (3.1b).

First, `S2D` projects the origin O onto the affine hull of the vertices in $\{s_j\}$ to obtain the point p_O. To do so, it computes a vector $\mathbf{p}_O$:

$$\mathbf{p}_O = \frac{\mathbf{s}_1 \cdot \mathbf{n}}{\|\mathbf{n}\|^2}\ \mathbf{n} \tag{3.7}$$

where $\mathbf{n}$ is the normal of the triangle: $\mathbf{n} = (s_2 - s_1) \times (s_3 - s_1)$.

It should be noted that the denominator of Eq. (3.7) would become null for degenerate simplices, however `S2D` handles this case automatically. If the points $\{s_j\}$ are (almost) affinely dependent, the triangle τ is a needle, and the cancellation error makes all components of $\mathbf{p}_O$ a NaN value. When passed to `CompareSigns`, the NaN value triggers independent searches on selected subsets of τ and the correct answer is robustly computed.

The vector $\mathbf{p}_O$ has direction $\mathbf{n}$ and length equal to the distance between the origin O and the affine hull of $\{s_j\}$; let us demonstrate that from the point p_O, whose coordinates feed into Eq. (3.2), can be computed. Consider the vector $\mathbf{h} = \mathbf{p}_O - \mathbf{s}_1$ laying on the affine hull of $\{s_j\}$. The projection point is given by: $s_1 + \mathbf{h} = s_1 + \mathbf{p}_O - \mathbf{s}_1 = s_1 + \mathbf{p}_O - s_1 + O = \mathbf{p}_O + O$. In fact, the projected point is $p_O = \mathbf{p}_O + O$.

In order to descend from $\mathbb{R}^3$ to $\mathbb{R}^2$, the point p_O and all the vertices of τ are projected onto the "safest" Cartesian plane. This is identified as the plane on which the simplex shades the largest area.

The procedure projects and computes the areas of the fictitious simplices all at once by evaluating the minors $M_{1,4}, M_{2,4}, M_{3,4}$ of the matrix in Eq. (3.5). The pseudocode in Algorithm 3 shows an implementation of `S2D`. A for loop computes $\mu_{max} = \max\{|M_{1,4}|, |M_{2,4}|, |M_{3,4}|\}$ and the coordinate J that will be discarded to project the face and p_O onto a Cartesian plane. Afterwards, the signs of fictitious simplices and μ_{max} are compared using the usual `CompareSigns` function. For a 2-simplex, the system $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$ is obtained by discarding the J-th coordinate from the vertices of τ and p_O. This is readily done by removing the J-th row from $\mathbf{M}$, see Eq. (3.5), and using its first and second minors to compute the barycentric coordinates. For cases in which

Algorithm 3 Sub-routine for 2-simplex

procedure S2D($\{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}$)
  $\mathbf{n} = (\mathbf{s_2} - \mathbf{s_1}) \times (\mathbf{s_3} - \mathbf{s_1})$
  $\mathbf{p_o} = (\mathbf{s}_1 \cdot \mathbf{n})\ \mathbf{n} / \|\mathbf{n}\|^2$
  $\mu_{\max} = 0,\ k = 2,\ l = 3$
  for $i = 1 : 3$ **do**
    $\mu = s_2^k s_3^l + s_1^k s_2^l + s_3^k s_1^l - s_2^k s_1^l - s_3^k s_2^l - s_1^k s_3^l$
    if $|\,\mu\,| > |\,\mu_{\max}\,|$ **then**
      $\mu_{\max} = \mu,\ J = i$
    $k = l,\ l = i$
  Discard the J-th coordinate from $\{\mathbf{s}_i\}$ so that $\mathbf{s}_i = [s_i^x, s_i^y]$
  $k = 2,\ l = 3$
  for $j = 1 : 3$ **do**
    $C_j = (-1)^j (p_o^x s_k^y + p_o^y s_l^x + s_k^x s_l^y - p_o^x s_l^y - p_o^y s_k^x - s_l^x s_k^y)$
    $k = l,\ l = i$
  if CompareSigns($\mu_{\max}$, C_j) for all j **then**
    $\lambda_j = C_j\ / \mu_{\max}$
    $W = \{\mathbf{s_1},\ \mathbf{s_2},\ \mathbf{s_3}\}$
  else
    for $j = 2 : 3$ **do**
      if CompareSigns($\mu_{\max}$, $-C_j$) **then**
        [W^*, $\boldsymbol{\lambda}^*$] = S1D($\{\mathbf{s}_i : i \in \{1, 2, 3\} - j\}$)
        $d^* = \| \sum_{i \in W^*} \lambda_i^* \mathbf{s}_i \|$
        if $d^* < d$ **then**
          $W = W^*$
          $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$
          $d = d^*$
  return W, $\boldsymbol{\lambda}$

p_O lays outside the projected triangle, the function `CompareSigns` is invoked for each second minor, and if the output is 0, the vertex associated to it is discarded. Similarly to `S3D`, if one or more fictitious simplices have sign different to $\det \mathbf{M}$, the outcomes must be compared to find which subset W returns the minimum norm $\|\boldsymbol{\nu}(W)\|$.

Function `S1D` This function computes the point of minimum norm $\nu(\tau)$ of a 1-simplex by taking all steps described in Section 3.1.2. Two vertices s_1 and s_2 in $\mathbb{R}^3$ are the input to this function. As for the other functions, the output is the smallest subset W of vertices which support $\nu(\tau)$ and the associated barycentric coordinates. `S1D` begins by projecting the origin O onto the affine hull of $\{s_1, s_2\}$ to obtain the coordinates of the vector $\mathbf{p}_O$:

$$\mathbf{p}_O = \mathbf{s}_2 + \frac{\mathbf{s}_2 \cdot \mathbf{t}}{\mathbf{t} \cdot \mathbf{t}} \mathbf{t} \tag{3.8}$$

Where $\mathbf{t} = (s_2 - s_1)$. Alike `S2D`, degenerate simplices passed to Eq. (3.8) cannot compromise the robustness of the Signed Volumes and it can be shown, following the demonstration for Eq. (3.7), that the projection point is derived from Eq. (3.8).
In order to descend from $\mathbb{R}^3$ to $\mathbb{R}^1$, the point p_O and all the vertices s_j are then projected onto the "safest" axis of the Cartesian coordinate system. This is identified as the axis on which the simplex shades the largest length. Again, this projection is based on the comparison between the signs of the volume forms of the simplex and the fictitious simplices associated to it.
The pseudo-code in Algorithm 4 shows an implementation of `S1D`. The logic for obtaining the subset W and the barycentric coordinates is identical to the one described for `S2D`.

To conclude, by virtue of Carathéodory's theorem, the Signed Volumes method

Algorithm 4 Sub-routine for 1-simplex

procedure `S1D`($\{\mathbf{s}_1, \mathbf{s}_2\}$)
  $\mathbf{t} = \mathbf{s_2} - \mathbf{s_1}$
  $\mathbf{p_o} = (\mathbf{s}_2 \cdot \mathbf{t})/(\mathbf{t} \cdot \mathbf{t})\ \mathbf{t} + \mathbf{s}_2$
  $\mu_{\max} = 0$
  for $i = 1 : 3$ **do**
    $\mu = s_1^i - s_2^i$
    if $|\,\mu\,| > |\,\mu_{\max}\,|$ **then**
      $\mu_{\max} = \mu,\ I = i$
  Keep only the I-th coordinate from $\{\mathbf{s}_1, \mathbf{s}_2\}$
  $k = 2$
  for $j = 1 : 2$ **do**
    $C_j = (-1)^j (s_k^I - p_O^I)$
    $k = j$
  if `CompareSigns(`$\mu_{\max}$`,`C_j`)` for all j **then**
    $\lambda_j = C_j\ / \mu_{\max}$
    $W = \{\mathbf{s_1},\ \mathbf{s_2}\}$
  else
    $\lambda_1 = 1\ ,\ W = \mathbf{s}_1$
  return W, $\boldsymbol{\lambda}$

solves small systems of only $r+1$ equations: these correspond to the number of vertices needed to express the projection of $\nu(\tau)$. This method recursively seeks the minimum set of vertices $W \subset \tau$ and, at the same time, gets rid of affinely dependent vertices. Whilst Johnson's algorithm solves a system of equations for each subset $W \subset \tau$ and then tests if Eq. (2.19) is verified or not, the Signed Volumes method identifies the unique subset W for which Eq. (3.1b) holds true, and only then it solves a system of equations to compute the barycentric coordinates.

3.2 Improved distance method for convex bodies

The second novelty presented in this thesis is a procedure for the solution of distance queries between convex bodies. It is required that any body representation (e.g. polytopes, quadrics, splines) may be treated and that the minimum distance is computed

accurately to machine precision. Moreover, this chapter presents, for the first time, a study on the impact that the distance sub-algorithm has on the GJK algorithm. This provides a better understanding about the numerical instability that were observed, but not addressed, in the literature.

3.2.1 On the instability of the GJK algorithm

This section investigates the numerical instability of the GJK algorithm. From the literature review in Section 2.3.3, it was concluded that the lack of accuracy and robustness prevents this fast procedure from being adopted in computational mechanics. The literature states that the major source of numerical error is the distance sub-algorithm, however it does not explain in what capacity and to what extent. Therefore, more the information are needed before improving for the GJK algorithm.

Let us first notice that the numerical instability manifests itself in infinite loops (173, 225). This is readily explained by looking at the simplistic interpretation in Eqs. (2.14)-(2.15): if the new search direction, $\mathbf{v}_{k+1}$, is not better than the previous one, $\mathbf{v}_k$, the procedure does not descent monotonically and the logic breaks.

This observation suggests that the numerical instabilities affect both the final result and the convergence rate. The fact that Johnson's algorithm assembles an algebraic system of equations whose coefficient are affected by rounding error, Eq. (2.20), introduces inaccuracy to the computation of the search direction. Unless $\mathbf{v}_k$ is computed accurately to machine precision, the search direction for a subsequent GJK-iteration cannot lead to the shortest solution-path. This has never been observed in previous studies, and it will be demonstrated by the numerical experiments in Chapters 4 and 5.

A large number of tests have been carried out to better understand under what

circumstances the GJK algorithm loops infinitely. It appears that scenarios in which bodies placed at a distance larger than an certain characteristic length are not critical. In particular, a series of spherical and polyhedral particles was tested using as characteristic length the diameter of the minimum bounding sphere. However, for bodies that are very close or touching, about 10% of distance queries fails. For these configurations simplices develop into slender and degenerate geometries.

In some sense, degenerate simplices are intrinsic to the way the GJK algorithm solves distance queries. One can observe that flat bodies, and bodies with large curvatures, are more likely to originate extremely deformed simplices. The reason is because a simplex takes the shape of the configuration space obstacle CSO: if the faces of the CSO closer to the origin are nearly flat, so will be the last simplex of the GJK procedure. An example in 2D was shown in Figure 2.7, in which the last simplex indeed adapts to the "flat" portion of the CSO. For 3D cases this problem is even more severe, however it is difficult to visualise and it will not be detailed further.

Surprisingly, by tracking the simplex evolution during the iterative process, it can be observed that not all degenerate simplices imply failure. Let us consider a particular test of two configurations for a pair of polygons: overlapping (Figure 3.3(a)) and distant (Figure 3.3(c)). The GJK algorithm follows the same procedure illustrated in Figure 2.7 to compute the minimum distance. At the k-th iteration, $\nu(\tau_k) = \nu(\text{CSO})$, and hence the GJK algorithm terminates. The particular orientation of the polygons, however, is such that the simplices τ_k result nearly-degenerate for both configurations. Figure 3.3(b) and Figure 3.3(d) present the simplex for the overlapping and distant polygons, respectively.

The simplices in Figures 3.3(b) and 3.3(d) have the same nearly-infinitesimal area

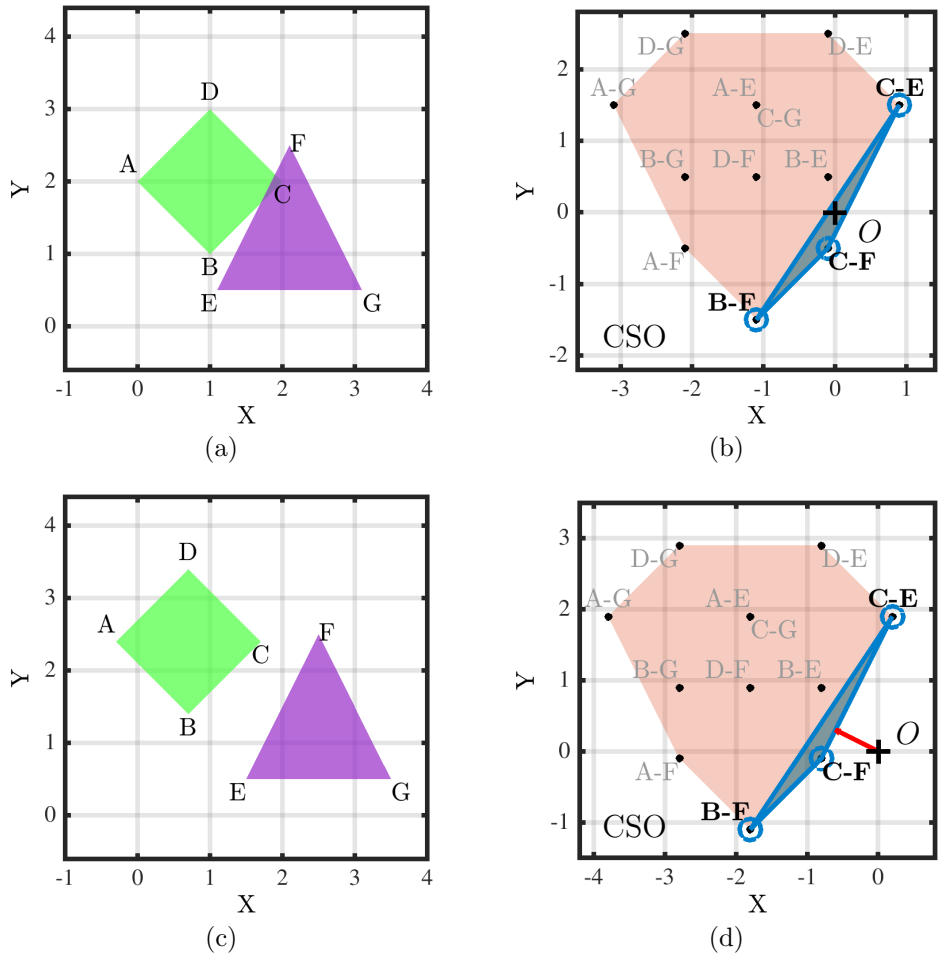


Figure 3.3: Overlapping polygons (a) and distant polygons (c) with the highly-deformed simplices generated upon termination of the GJK procedure, respectively (b) and (d).

but, since they have different sets of points supporting $\nu(\tau_k)$, Johnson's algorithm fails only for the overlapping configuration. The reason is that the sub-algorithm converges when testing different Voronoi regions: $V_{(1,2,3)}$ for the overlapping polygons, and $V_{(1,2)}$ for the distant polygons. The difference between these Voronoi regions is that the first one is nearly-degenerate, the second one is not.

The new information obtained from this study is that numerical instabilities affecting the GJK algorithm are not due to degenerate simplices themselves, but rather to the region of simplex supporting the point of minimum norm. Whenever this is infinitesimal, the orthogonal condition embedded into Eq. (2.20) yields to numerical

instabilities. Furthermore, it has been found that the convergence rate decays if the search direction is not computed accurately at each GJK-iteration.

3.2.2 A new robust sub-algorithm

The Signed Volume method can replace Johnson's algorithm since it is capable of computing the search direction $\mathbf{v}_k$ accurately to machine precision. This method, introduced in Section 3.1, is designed to be robust and to handle rounding error. For this reason the Backup procedure can be withdrawn from the GJK algorithm.

It should be noted that, since the GJK algorithm updates the simplices by adding only one vertex at a time, only 2^m (rather than $2^{m+1}-1$) Voronoi regions can possibly support the point $\nu(\tau_k)$. For the example shown in Figure 2.7(d), the point $C-E$ is added last and, for the descending nature of the GJK algorithm, it must support the simplex point of minimum norm. Whereas, it is unlike that the point $D-G$ can support the solution. Taking this into account can reduce the number of operations, but some adjustments have to be made to the Signed Volumes method.

The Signed Volumes method can be tailored to exclude those Voronoi regions that cannot support the point of minimum norm of a simplex. As already mentioned, since only one vertex is added per GJK-iteration, the new vertex must support the solution. This means that any Voronoi region defined by a tuple not including the new vertex cannot support the point of minimum norm of a simplex. These regions can therefore be excluded a priori. Figure 3.4 illustrates an example in which the vertex labelled as 1 is the latest added to the simplex. In this case the sub-algorithm should ignore the regions $(2,3)$, (2) and (3). This requires little coding effort because the Signed Volumes method is recursive from top to bottom by definition, whereas Johnson's

Algorithm 5 GJK algorithm

procedure GILBERT-JOHNSON-KEERTHI $(P, Q, \mathbf{v}_0)$
  $k = 0$
  $\mathbf{v}_1 = \mathbf{v}_0$
  $\tau_k = \emptyset,\ W_k = \emptyset$
  repeat
    $k = k + 1$
    $\mathbf{w}_k = \mathbf{s}_P(-\mathbf{v}_k) - \mathbf{s}_Q(\mathbf{v}_k)$;
    if $\|\mathbf{v}_k\|^2 - \mathbf{v}_k \cdot \mathbf{w}_k \leq \varepsilon_{rel}^2 \|\mathbf{v}_k\|^2$ **then**
      continue
    $\tau_k = \{\mathbf{w_k}\} \cup W_{k-1}$;
    $[W_k,\ \lambda] =$ `SignedVolumes`(τ_k);
    $\mathbf{v}_{k+1} = \sum \lambda_i \mathbf{y}_i : \mathbf{y}_i \in W_k,\ i = 1, \ldots, |W_k|$;
  until $|W_k| = 4$ or $\|\mathbf{v}_k\|^2 \leq \varepsilon_{tol} \max\{\|\mathbf{y}_k\|^2 : \mathbf{y}_k \in W_k\}$
  return $\|\mathbf{v}_k\|$

algorithm needs data caching to exclude these Voronoi regions.

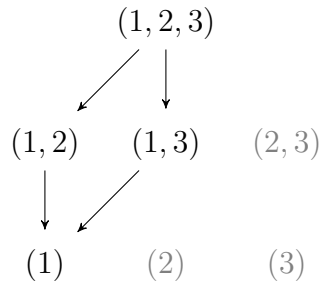


Figure 3.4: The Signed Volumes method can be tailored to exclude from the recursive search the tuples disconnected and in grey colour to improve the performance.

Indeed, given an m-simplex, the Signed Volumes method allows to inspect only 2^m Voronoi regions, rather than $2^{m+1} - 1$, without any additional cached data. The little coding effort required is outlined in the next section.

3.2.3 Implementation

The improved implementation of the GJK procedure proposed in this thesis is shown in Algorithm 5. This receives in input two arbitrary, and possibly different, representations of body P and Q, and an initial guess vector $\mathbf{v}_0$. The latter is usually obtained

from a previous call to the GJK algorithm. The procedure begins by initialising the iteration counter k, the simple τ_k and the solution of the support functions $\mathbf{w}_k$. The conditional loop first evaluates the support mapping on the two bodies and stores the Minkowski difference in $\mathbf{w}_k$ (line 7). If this is close to the separating vector computed at the iteration $k-1$, the GJK algorithm terminates and returns $\mathbf{v}_k$. Notice that this test is based on a relative tolerance ε_{rel}. If the first test is negative, a new vertex is added to the simplex τ_k which is then passed to `SignedVolumes`, the function implementing the Signed Volumes method. This computes $\mathbf{v}_{k+1}$ as convex combination of barycentric coordinates λ_i of the smallest subset W_k. For 3D (2D) problems, the last test verifies if W_k has cardinality equal to four (three), and if the simplex has moved toward the origin or less than an absolute tolerance ε_{tol}.

Algorithm 5 differs from the original implementation and from other published to date. It omits various pre-processing operations outlined in the original paper by Gilbert et al (89) because, despite the claim of enhance robustness, those operations have been found to have little relevance in practise. The implementation presented here also omits the termination condition $\mathbf{w}_k \in Y$ introduced in (224). This is not particularly efficient on finely meshed bodies, such as those encountered in computational mechanics, and is essentially applicable to polytopes only. When compared to SOLID3, the renown open-source collision detection library (226), Algorithm 5 does not invoke the Backup procedure and it does not attempt to terminate once this is be triggered. In fact, SOLID3 (version 3.5.8) terminates once the Backup procedure is invoked. Although there might be practical reasons behind this choice, the risk is that the final solution to the distance is coarsely approximated. Finally, Algorithm 5 is simpler than other implementations published to date and yet it aims to provide fast

and accurate solutions to distance queries.

If implemented in ANSI C, Matlab or similar languages, the only data structure needed in Algorithm 5 concerns the simplex. To account for the various configurations of a m-simplex in $\mathbb{R}^n$, a structure `simplex` is implemented. This includes: the number of vertices currently in the simplex, $n+1$ or fewer barycentric coordinates, the coordinates of $n+1$ or fewer vertices of of the first body, the coordinates of $n+1$ or fewer vertices of of the second body, and a list of $n+1$ or fewer points supporting the point of minimum norm of the simplex. These variables are stored as follows:

```
struct {
        int             nvrtx;
        int             Wids[n+1];
        float           lambdas[n+1];
        float           YAcrd[n+1][n];
        float           YBcrd[n+1][n];
} simplex;
```
(3.9)

The way the indexes `Wids` are stored in Eq. (3.9) is particularly important and it is convenient to place the latest vertex added to the subset at the beginning. Keeping track of the order in which vertices are added to the simplex makes the implementation simple and efficient. In fact, this trivial bookkeeping allows to easily exclude those Voronoi regions that cannot support the point of minimum norm of a simplex.

Tailoring the Signed Volumes method for the GJK algorithm requires basically no coding effort. The three lines which need editing are: line 9 in Algorithm 2, line 12 in Algorithm 3 and line 11 in Algorithm 4. For these the loops need to run from $j = 2$, instead of $j = 1$; as a result, given an m-simplex, the $(m-1)$-facet which does not include the latest vertex and its child faces are excluded.

This concludes the description of an efficient and robust implementation of the GJK algorithm, however further optimisation may be implemented.

Optimisation Whilst the implementation presented thus far excludes certain Voronoi regions, it may repeat the inspection of some regions. For instance, by looking at the diagram in Figure 3.4 there are two arrows pointing at the region $V_{(1)}$ indicating that this region may be inspected twice. This could add a little computational effort, more pronounced for 3-simplices: as shown in Figure 3.2, the regions $(1,2)$, $(1,3)$, $(1,4)$ and (1) may be all inspected twice.

These regions are not always inspected twice, but only when two faces of the same simplex face the origin. This scenario is usually encountered when the bodies are very close to each other since the simplex adjusts to the boundary of the CSO. For 3D applications however, repeated inspections to the same Voronoi region are not uncommon and add useless computing operations.

To optimise the implementation, one needs to enrich the data structure in Eq. (3.9) by addingan `int` variable `notWids[n+1]`. This may be seen as a list of fictitious vertices that cannot support the point of minimum norm, and therefore will be ignored by the recursive search.

3.3 Intrinsic hierarchy for arbitrary bodies

Lastly, the third novelty introduced in this thesis is an hierarchical search for the minimum distance between arbitrary bodies. This is specifically designed to limit the computing effort associated to spatial data structures, and yet to work on all representations of solid body. The idea is to carry out the coarse phase without assigning a

frame of reference, thus using intrinsic variables only, and build spatial data structures exclusively if the bodies are sufficiently close, i.e. at an arbitrary distance.

3.3.1 Overview

The solution of distance queries between arbitrary bodies presents a number of challenges: a body may be concave, its morphology can change as the simulation progresses, and it may be represented as a compound of various primitives. The literature review presented in Section 2.3 has highlighted that such general scenarios are best handled by hierarchies of convex bounding volumes.

All bounding volumes employed to date assign a frame of reference, and this is a mayor limitation that breaks the link between body and space-partitioning scheme. Basically, even for bodies that are always far apart, their spatial data structure has to be updated as bodies displace and deform. Moreover, the criteria that trigger the update of space-partitioning schemes have a significant impact on the overall computing time. This makes the solution of distance queries complex and expensive, particularly in dynamic simulations and optimisation problems which require time-marching schemes or iterative methods for the solution (247).

The method introduced in this section resists the temptation of assigning a frame of reference and is thought to be the first of its kind in computational mechanics. During the first (coarse) phase of the hierarchical search, the bounding volumes are defined by means of intrinsic quantities only. These create a strong coupling between body and space-partitioning, in fact, the bounding volume automatically updates as a body moves and deforms. Extrinsic binary trees are used only in a second (fine) phase when two bodies are found "close enough".

The novelty is to build a tree for each couple of bodies only once these are found at an arbitrary distance. For example, in the specific application of contact search, the value for this distance is null, so that spatial data structure are only build once collision is imminent.

Overall, it is expected that this method will reduce the number of operations for the broad contact search. Further details on the formulation and the implementation of this method are presented in the following sections.

3.3.2 Theoretical algorithm

The new hierarchical algorithm comprises three distinct phases. It is required that each phase exploits spatial coherence and operates on any representation of solid bodies, thus aiming for high performance and versatility. Notice that the term "search" is here used in its broadest sense, as this method may be used for problems like contact search or closest neighbour search.

The three major phases are: (i) approximate convex decomposition of bodies, (ii) broad intrinsic search and (iii) narrow extrinsic search. Their sequence is outlined in Figure 3.5. After the narrow phase a number of *update criteria* are tested to check whether the body decomposition has to be repeated or not.

The decomposition begins by importing the bodies and subdividing them into smaller quasi-convex parts called *sub-bodies*. These are then tested at the broad and the narrow phases; the former uses the convex hulls to estimate the minimum distance between a pair of sub-bodies, the latter builds a binary tree for each pair of bodies to find a list of close neighbours. These steps solve the distance query for a single solution step. Before proceeding to the next step, a number of update tests are executed. If

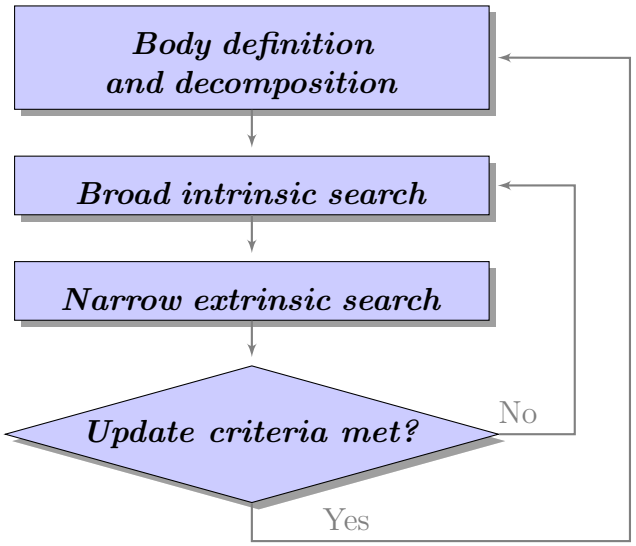


Figure 3.5: Hierarchical contact search

one of these criteria is met, one or more sub-bodies have to be recomputed; if not, the solution continues to the next step and the distance query is repeated. Notice that the body decomposition should not be confused with domain decomposition techniques, such as (133), herein only the outer surfaces of a body is considered, not its interior.

The convex hull and its support function are evaluated without assigning a frame of reference, and this is a key advantage of the new broad search over common approaches. It is expected that this will reduce the overall operations for the broad search. The validation of this hypothesis is provided in Chapter 4.

Figure 3.6 illustrates with an example the various phases of the hierarchical search. The example considers two quasi-convex bodies that move toward each other, see Figure 3.6(a). At each solution step, the minimum distance between their convex hulls is computed without having to introduce a frame of reference, see Figure 3.6(b). Once this distance is less than an arbitrary value, a frame of reference is assigned to build an AABB around each facet of each body. At subsequent solution steps, a binary tree tests the AABBs for collision. Figure 3.6(d) highlights two faces whose AABBs are

found in contact and are therefore tested using suitable method for primitive testing. Notice that all phases can be carried out with this method, regardless of the description of the bodies.

More details on each phase are given below, whilst Section 3.3.3 presents an implementation which seamlessly combines all phases.

Approximate convex decomposition The aim of this preliminary phase is to identify all bodies and to decompose them in approximatively convex sub-bodies. Distinct bodies are readily identified from the input, e.g. from the connectivity matrix of a FEM simulation; however, the decomposition is not a trivial operation. Let us consider a body $\Omega \subset \mathbb{R}^n$ and an arbitrary decomposition of its boundary Γ:

$$\Gamma = \bigcup_{i \in I} \Gamma_i \qquad \text{for} \qquad I = \{1,\ ...,\ N\} \tag{3.10}$$

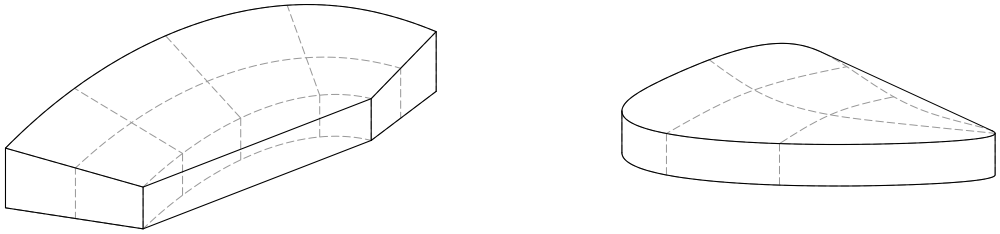
Although not uniquely defined, this decomposition is exact in the sense that it provides a finite set of sub-boundaries whose union is Γ itself.

In order to uniquely specify the decomposition in Eq. (3.10) a criterion must be defined. Ideally, this should enable a quick decomposition but also a quick evaluation of $d(\Gamma_i, \Gamma_j)$, for all $i,\ j \in I$. Furthermore, if each sub-body is nearly convex, the evaluation of this distance for $i = j$ may be avoided altogether. This would simplify tremendously the broad search, however a measure of the *convexity* of a body is necessary. The convexity $C(\Gamma)$ of a body with boundary Γ may be measured as follows:

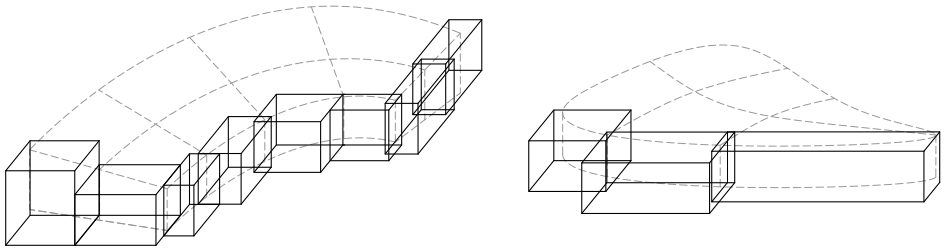
$$C(\Gamma) = \min_{x \in \Gamma} \ \|x - P(x)\| \tag{3.11}$$

where $P(x)$ is the projection of a point x on the boundary of $\text{conv}(\Omega)$ with respect to the half-line with origin x and directional normal to Γ at x. The decomposition is

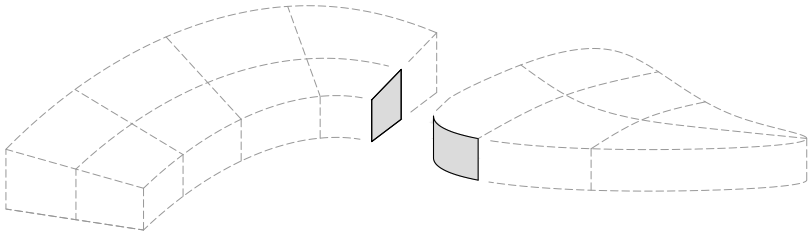
(a)



(b)



(c)



(d)

Figure 3.6: Distance queries between two arbitrary bodies (a) are computed hierarchically in different phases. Firstly, the distance between their convex hulls is computed (b). The second phase involves the generation of bounding volumes around the faces (c), for the sake of clarity not all AABBs are illustrated. Finally, the distance between faces is computed (d).

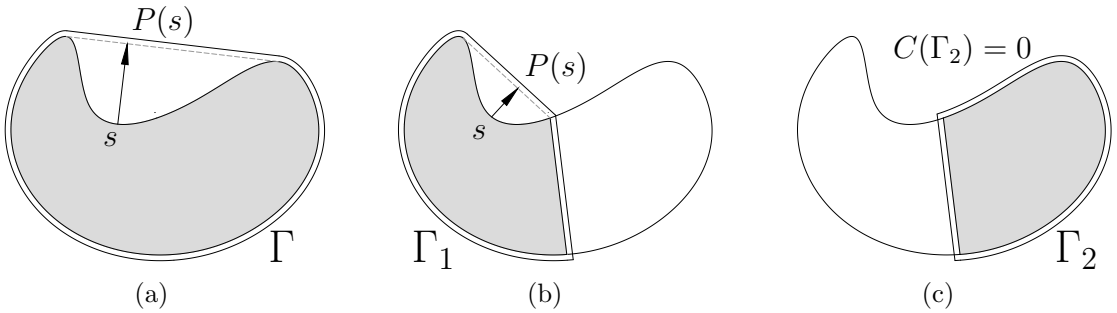


Figure 3.7: Given a concave body Ω with frontier Γ (a), this may be decomposed into two sub-bodies, whose boundaries have convexity $C(\Gamma_1)$ and $C(\Gamma_2)$ less or equal than an arbitrary positive value. The approximatively convex sub-bodies resulting from the decomposition are shown in (b) and (c), and their convexity is represented as the magnitude of a vector $\|s - P(s)\|$, where s lays on the frontier of a sub-body and $P(s)$ is its projection on the convex-hull of the sub-body.

thus required to provide, for an arbitrary body Ω, a set of sub-bodies with convexity at the most ε_c; namely,

$$\Gamma \approx \bigcup_{j=1}^{N} \operatorname{conv} \Gamma_i \quad \text{with} \quad C(\Gamma_i) \leq \varepsilon_c. \tag{3.12}$$

The convex decomposition is illustrated with an example in Figure 3.7. A concave body Ω has convexity $C(\Gamma)$, whilst for a certain decomposition each sub-boundary has a convexity much smaller: $C(\Gamma_1) = \|s_1 - P(s_1)\|$ and $C(\Gamma_2) = 0$.

The method described in (148) provides a decomposition of the kind illustrated in Figure 3.7, albeit this has never been applied in computational mechanics before. The method was originally designed for triangular meshes only, but it can be readily extended to other representations of solids (see Section 4.7 for the particular case of NURBS surfaces). An example of decomposed triangular mesh surface is shown in Figure 3.8; this includes the original model, its convex hull, and the convex hulls of the resulting sub-bodies. An example of decomposed trivariate solid NURBS is shown in Figure 3.9, this includes the original model, its convex hull, and the convex hulls of the resulting sub-bodies.

As shown earlier in Figure 3.5, this phase is invoked once at pre-processing and is carried out without assigning a frame of reference. Only occasionally, at runtime, a sub-body that deforms significantly, or changes its morphology, has to be re-decomposed.

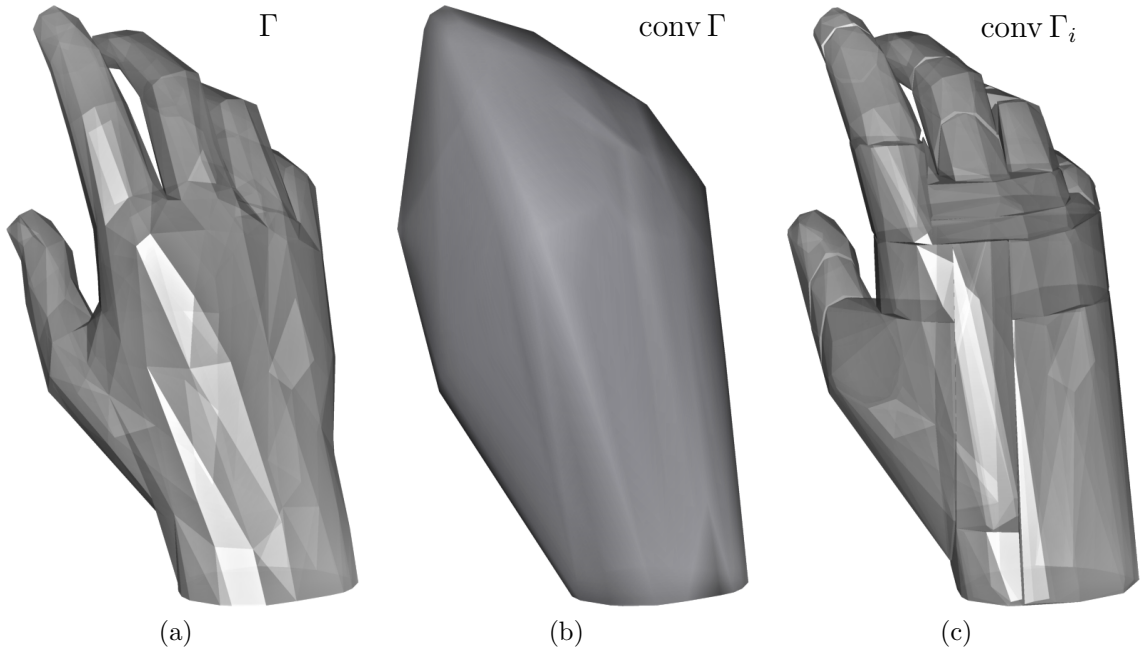


Figure 3.8: Hand model described by polygonal mesh (a), its convex hull (b), and an approximated convex decomposition represented by the convex hulls of the resulting sub-bodies (c).

Intrinsic broad search This phase endeavours to provide a list of sub-bodies that are within a distance not greater than ε_d. Since this is just a coarse (broad) level of search an approximate, but conservative, answer is sought. The level of approximation depends on the choice of bounding volume for the sub-body Γ_i and on the specific application. For instance, in contact search $\varepsilon_d = 0$ and any bounding volume reviewed in Chapter 2 may be used.

The new method uses, as shown in Figure 3.6(b), the convex hulls $\operatorname{conv}\Gamma_i$, for $i \in I = \{1, ..., N\}$, as bounding volume of all sub-bodies. This choice brings several advantages:

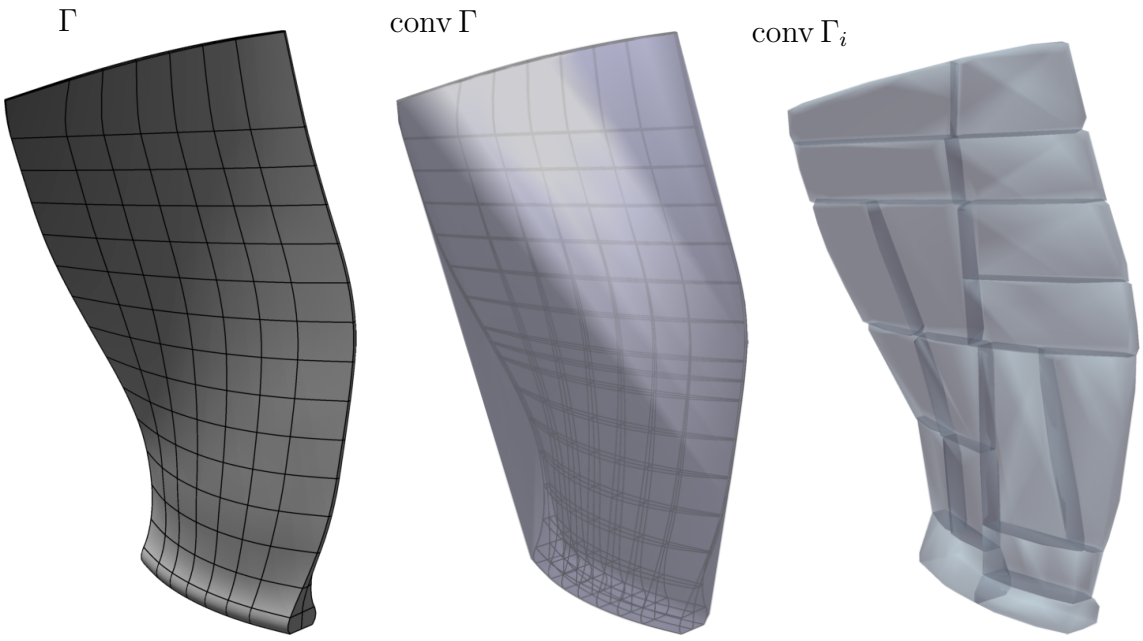


Figure 3.9: Turbine fan blade described by trivariate NURBS (a), the convex hull of its control points (b), and an approximated convex decomposition represented by the convex hulls of the resulting sub-bodies (c).

1. There is no need to assign a frame of reference to compute the minimum distance $d(\operatorname{conv}\Gamma_i, \operatorname{conv}\Gamma_j)$ for $i, j \in I$.
2. Convex hulls are, by definition, the tightest bounding volumes. Namely, for an arbitrary boundary Γ, with convexity $C(\Gamma) \leq \varepsilon_c$, the convex hull $\operatorname{conv}\Gamma$ encloses the smallest volume than any other bounding volume.
3. The support function on $\operatorname{conv}\Gamma$ may be evaluated for any representation of Γ and without computing explicitly its convex hull. See Chapter 2 for meshes, quadrics or compounds of these, and Chapter 4 for the particular case of NURBS.
4. The design of multi-level broad searches is straightforward since $\operatorname{conv}(\Gamma_i) \subset \operatorname{conv}(\Gamma)$ for $i \in I$. An illustration of this hierarchical subdivision is shown by the examples in Figures 3.8 and 3.9, where the body, its convex hull and an approximate decomposition for polygons and NURBS are depicted.

The Euclidean distance between convex hulls of sub-bodies is the intrinsic variable that governs the novel broad search. This is a good, and conservative, approximation because the maximal separation between convex sets is equal to the smallest distance between them (52).

However, computing the distance for all sub-body pairs is computationally expensive. These are $(N^2-N)/2$ distance queries which populate the upper triangle of the matrix:

$$D = \begin{bmatrix} 0 & d_{1,2} & \cdots & d_{1,N} \\ & 0 & \ddots & \vdots \\ & & 0 & d_{N-1,N} \\ 0 & & & 0 \end{bmatrix} \tag{3.13}$$

where $d_{i,j}$ is short for $d(\operatorname{conv}\Gamma_i, \operatorname{conv}\Gamma_j)$.

The new method exploits spatial coherence to avoid recomputing at each solution steps all values $d_{i,j}$: it is assumed that the distance between two bodies varies only marginally in subsequent time-steps. This suggests that $d_{i,j}$ may be approximated by a function $\tilde{d}_{i,j}$ whose evaluation is sufficiently cheap to be evaluated at each solution steps for all values of matrix D.

The function $\tilde{d}_{i,j}$ is defined so that: (i) $\tilde{d}^t_{i,j} \leq d^t_{i,j}$ holds for all time-steps t, and (ii) it can be updated incrementally with little computational effort. The following definition meets these requirements:

$$\tilde{d}^{t+1}_{i,j} = \tilde{d}^t_{i,j} - \left(\max_{\mathbf{s}_i \in \Gamma_i} \|\mathbf{s}_i\| + \max_{\mathbf{s}_j \in \Gamma_j} \|\mathbf{s}_j\| \right) \tag{3.14}$$

where $\mathbf{s}_i$ and $\mathbf{s}_j$ are the displacements of two points, respectively on Γ_i and Γ_j, accumulated in a solution step. Effectively an approximation of the accumulated displacement, $\tilde{d}_{i,j}$ may evaluated very efficiently. At time $t = 0$ is taken $\tilde{d}^0_{i,j} = d^0_{i,j}$. Other definitions different from Eq. (3.14) may be used, however this is preferred for simplicity and suit-

ability to dynamic simulations.

It is assumed that this logic will reduce significantly the of number calls to distance queries, and Chapter 4 presents a series of validation tests for this hypothesis.

Once $\tilde{d}_{i,j} \leq \varepsilon_d$, a distance algorithm is invoked, to recompute $d_{i,j}$, for the single pair comprising Γ_i and Γ_j only. If $d_{i,j} \leq \varepsilon_d$, the pair is passed to the narrow search that is described below.

Extrinsic narrow search In this phase, the distance queries are carried out at the facet level. Given a list of convex or quasi-convex *body-pairs* close to each other, the narrow search creates a new list of close *facet-pairs*. Recall that a facet is, for all body representations, the image of an outer portion of parametric domain. The procedure described here is rooted in this definition and it is designed to equally process meshes, NURBS, quadrics and compounds of these.

Unlike other methods, herein each body-pair is processed individually. For the example illustrated in Figure 3.6(c), a number of AABBs is built and sorted in a binary tree; these are built, traversed and tested individually for each pair passed from broad to narrow search. Conventional methods do not consider a single pair of sub-bodies, but build a single tree for all bodies in the domain. Instead, the new approach generates a larger number of trees, but of smaller size.

The choice of using many binary trees is based on the assumption that it is more efficient to process smaller trees than a large one. Multiple binary trees require more memory, but they are more likely to remain well-balanced as bodies move and deform. Furthermore, they are only built and traversed on demand from the broad search. This method needs more operations at runtime for building the trees, but less for traversing

and, more importantly, rebalancing them. The idea is not to reinvent bounding volumes and space-partitioning schemes, but to make an innovative use of them.

The underlying assumption made about the performance of the multiple trees may not hold for computer graphics and other applications, but will be verified for computational mechanics in Chapter 4 and Chapter 5. The verification tests will concern dynamic simulations, in which distance queries are used to perform contact search.

Let us now present how each facet may be bounded. This presentation employs AABBs and ADTs because widely employed, but the methodology suits any other data structure reviewed from the literature in Chapter 2, e.g. object-oriented bounding-boxes and BSP tree. The particular context of dynamic simulations is considered, in which bodies move and each AABB encloses a facet and its buffer zone. The buffer zone, or *Verlet distance*, is a fictitious layer that often takes into account: the size, the velocity and possibly the convexity of a facet. Herein, the buffer zone z_{b} is defined as follows:

$$z_{\mathrm{b}} = \varepsilon_{\mathrm{buffer}} + v_{\max}\, \Delta t \tag{3.15}$$

where $\varepsilon_{\mathrm{buffer}}$ is an arbitrary coefficient in the order of the squared-root of the machine precision, $v_{\max}$ is the maximum nodal velocity of a facet and Δt is time-step increment. Different constructions of AABBs associated to z_{b} are illustrated in Figure 3.10. The simplest is in Figure 3.10(a), which also results in the largest size. The smallest AABB is built by taking a normal offset to the facet, as shown in Figure 3.10(b). The configuration in Figure 3.10(c), which takes into account the orientation of the facet, is used throughout this work.

The rest of this section presents flowcharts that describe the processes of: approximate decomposition, broad and narrow searches. These are the major modules

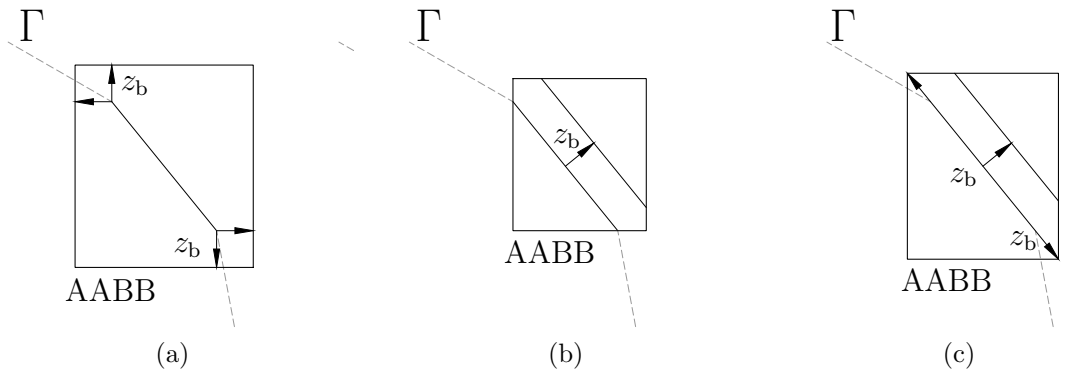


Figure 3.10: Different types of AABB constructions around a 2D flat facet based on the buffer zone value z_b: (a) expands the box along the axis directions, (b) projects the buffer zone outside the facet, and (c) accounts for normal and tangential direction.

in Figure 3.5 and will be here closely inspected. Firstly, the decomposition of sub-bodies is shown in Figure 3.11. This is a linear sequence of steps that starts by reading the input to define the bodies, decomposes them, and measures the distance $d(\text{conv}\,\Gamma_i, \text{conv}\,\Gamma_j) = d_{i,j}$ between a pair of resulting sub-bodies. The exact value $d_{i,j}$ is then used to initialise $\tilde{d}_{i,j}$, which will be used at runtime to estimate the distance between the convex hulls of the sub-bodies. Secondly, the intrinsic broad search is shown in Figure 3.12. This is invoked at runtime and begins by computing a new value for $\tilde{d}_{i,j}$. If this is greater than ε_d, the pair Γ_i, Γ_j is discarded from the list of potential contact pairs; whilst if equal or smaller than ε_d, the exact value for $d_{i,j}$ is recomputed and tested to establish whether the sub-bodies should be passed to the narrow phase or not. Finally, the extrinsic narrow search is shown in Figure 3.13. This is essentially a standard search based on AABBs and a binary tree. After these phases are completed, one or more criteria are tested to check if one of the sub-bodies has deformed to the point there its convexity exceeded the threshold valued ε_c, and if yes, the body will be re-decomposed. Notice that at runtime this decomposition is triggered only occasionally for a specific body, and not for all $(N^2 - N)/2$ pairs.

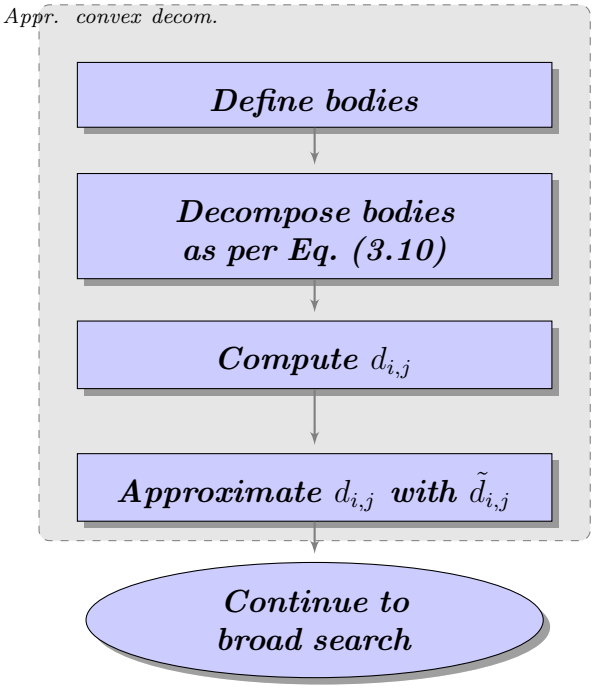


Figure 3.11: Workflow for the approximate convex decomposition module. A body Ω is decomposed into sub-bodies. The minimum distance between each sub-body is stored in the upper triangle of a matrix whose elements are $d_{i,j}$. Afterwards, to save computing time, the distance is approximated by $\tilde{d}_{i,j}$.

As mentioned earlier, this novel method makes use of existing routines in such a way to reduce the computing effort of space-partitioning. This is only possible because the broad search is conducted intrinsically, that is, without assigning a frame of reference to solve distance queries. The next section presents how to implement and combine the three major phases described thus far into a hierarchical framework.

3.3.3 Implementation

This section describes the data structures, algorithms and implementation details necessary to adopt the novel hierarchical search. For the sake of clarity, the presentation is tailored for solving contact detection problems using particular algorithms selected by the author. The methodology, however, is general and applicable to other algorithms.

The most important data structures to be stored concerns the sub-bodies and pairs

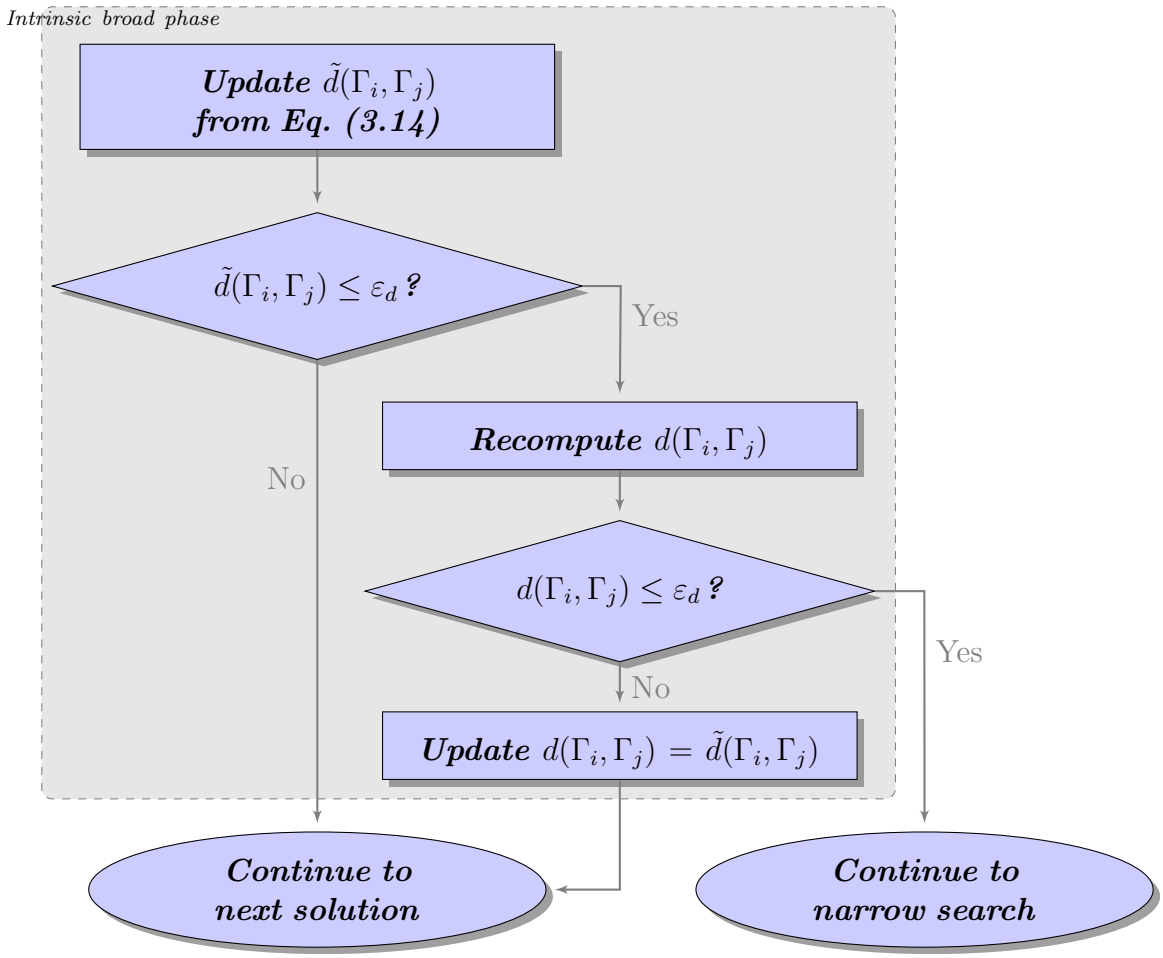


Figure 3.12: Workflow for the intrinsic broad search module. The approximate distance $\tilde{d}(\Gamma_i, \Gamma_j)$ between the i-th and j-th body is tested at each iteration against an arbitrary value ε_d. Only if smaller the exact distance $d(\Gamma_i, \Gamma_j)$ is recomputed.

of these. A sub-body is stored in the following structure:

```
struct {
        int          bodyid;
        int          parent;
        int          type;
        int          updatef;
        int          contactf;
        double       zbuffer;
        double       convexity;
        double       mdisp;
} subody;
```
(3.16)

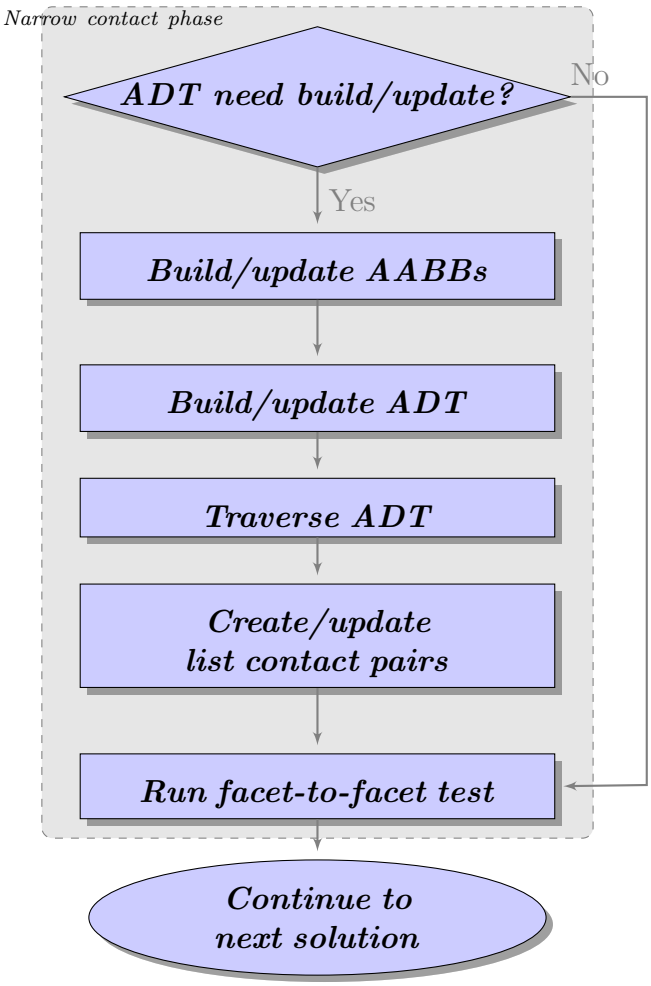


Figure 3.13: The workflow for the extrinsic narrow search module is linear and very similar to the one of existing methods. The test is repeated each time-step for all facets selected during the broad search.

This includes the unique ID, or label, of the sub-body and of its parent body. It is also necessary to store the type of description of the body; notice that each type presented in Chapter 2, needs more specific fields. For example, polygonal meshes are described by means of point coordinates. Two flags, `updatef` and `contactf`, are used to trigger the space-partitioning update and to track whether the sub-body is in contact or not with the other bodies. Finally, the values of z_b, ε_c and $\max_{\mathbf{s}_i \in \Gamma} \|\mathbf{s}_i\|$ are stored.

The information about a pair of sub-bodies are stored in the following structure:

```
struct {
        int             masterid;
        int              slaveid;
        int                  dd;
        int             ncntacts;
        int             mcntacts;
        int     lcntacts[mcntacts];
} bdpair;
```
(3.17)

This includes the ID's of the sub-bodies in the pair and the approximated squared distance between them (`dd`). The number and the list of overlapping AABBs, or equivalent bounding volumes, must be tracked for the narrow search; these are stored in `ncntacts` and `lcntacts` respectively. The maximum number of overlapping AABBs `ncntacts` is stored for memory management purpose.

The algorithms used herein for the hierarchical search have been taken from the literature, and most of them did not require any modification. For the approximated convex decomposition and the binary tree functionality, this work relies on third-party implementations. Recent publications in the field of computer graphics provide efficient algorithms for decomposition of polygonal models (62, 86, 136, 148). The algorithm described in (148) is efficient and simple to implement and is therefore used throughout this section for decomposing bodies. This is coupled with the ADT (25) for the space-partitioning of AABBs, the GJK algorithm presented in Section 3.2 to compute the distance $d_{i,j}$, and the Signed Volumes method.

The pseudo-code for the contact search is presented in Algorithm 6. This begins by searching for the maximum displacement experienced, in a single solution step, by

all sub-bodies. This value is then subtracted to the previously computed value of $\tilde{d}_{i,j}$. If this is greater than zero, the algorithm proceeds to the next pair of sub-bodies, alternatively it carries out the broad and narrow search as detailed in Figures 3.12 and 3.13.

Algorithm 6 Hierarchical contact search

```
1: for i = 1, ..., N do
2:     subbody = Γ_i
3:     for all Nodes j in Γ_i do
4:         if s_max < s_j then
5:             s_max = s_j                          ▷ Find maximum displacement
6:             subbody->mdisp = s_max
7:             subbody->updatef = 1
8: for i = 1, ..., N − 1 do
9:     for j = i + 1, ..., N do
10:        subbodyi = Γ_i , subbodyj = Γ_j
11:        d̃_{i,j} −= (subbodyi->mdisp + subbodyj->mdisp)
12:        if d̃_{i,j} > 0 then
13:            continue                             ▷ Carry on solution
14:        else
15:            if GJK is active then
16:                d̃_{i,j} = GJK( Γ_i, Γ_j )        ▷ Broad search
17:            if d̃_{i,j} ≤ ε_toll then             ▷ Narrow search
18:                Deactivate GJK
19:                if Bodies were in contact before then
20:                    bdpair->updatef = 0
21:                else
22:                    bdpair->updatef = 1
23:                if subbodyi->updatef OR
                      subbodyj->updatef OR
                      bdpair->updatef do
24:                    Invoke UpdateAABBs
25:                    Invoke UpdateADT
26:                    Invoke TraverseADT
27:                if Number of colliding AABB = 0 then
28:                    Activate GJK
29:            else
30:                continue                         ▷ Carry on solution
```

A characteristic aspect of contact detection problems is that bodies may contact and/or overlap for prolonged periods of time. This may be a consequence of the contact

formulation, e.g. penalty method, or the particular application, e.g. large sliding, but the consequence is an increased computational cost of the contact detection. This mechanism would trigger unsustainable computational costs due to the broad search. When two bodies are in contact, following from Eq. (3.14), we have $\tilde{d_{i,j}} \leq d_{i,j} = 0$ and therefore this procedure would trigger the re-computation of $d_{i,j}$ at each time-step for all contacting pairs of sub-bodies.

Algorithm 6 avoids this extra costs by activating or deactivating the broad search. The broad search is deactivated if the distance between the convex hulls of Γ_i and Γ_j is null, and is activated if there are no AABBs in contact. The test on the number of colliding bounding boxes is line 19 does exactly this. As the simulation advances, $d_{i,j}$ may become null and hence trigger the narrow search. At this moment the broad search is deactivated and the facets of Γ_i will be tested for collision against the facets of Γ_j by means of the ADT tree and AABBs. Namely, all subsequent time-steps will assume that $\operatorname{conv}\Gamma_i$ and $\operatorname{conv}\Gamma_j$ collide and the distance algorithm will not be invoked again. Only when the number of overlapping AABBs is zero the broad search will be reactivated.

3.4 Concluding remarks

Three new distance algorithms have been presented in this chapter: (i) the Signed Volumes method: a new recursive procedure for point–simplex distance queries; (ii) an improved version of the Gilbert-Johnson-Keerthi (GJK) algorithm for faster and more accurate distance queries between convex bodies; and, (iii) an innovative hierarchy of bounding volumes for arbitrary representations of concave objects. They are extremely

versatile and make use of spatial coherence to minimise the computing time. All common representations of solid bodies may be treated and these algorithms can be combined into a hierarchical search to tackle complex distance queries.

The Signed Volumes method solves point–simplex distance queries robustly and accurately. The projection of the origin onto the affine hull of a hyperplane is the most critical step, but it is carried out in such a way that the overall robustness cannot be affected.

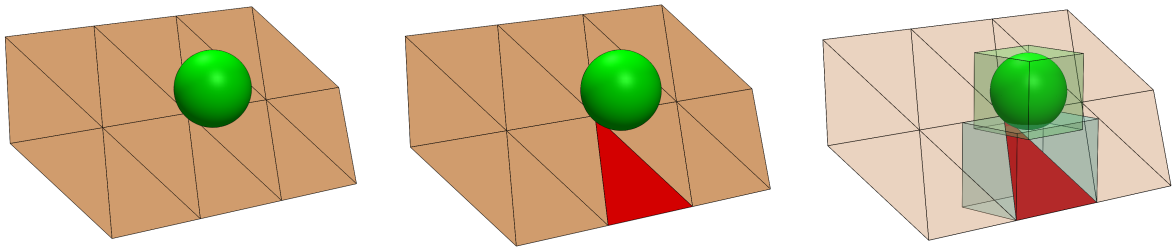
The improved GJK algorithm employs the Signed Volumes method as distance sub-algorithm. Combined together, these provided a robust and efficient procedure for measuring the distance between any pair of convex bodies.

Finally, an innovative hierarchical framework has been formulated on the assumption that a large number of small binary trees may be build and traversed more quickly than a unique tree. Before any space-partitioning data is built, a broad search is carried out without assigning a frame of reference. This is only based on the approximate displacement of each body and on the minimum distance between two bodies. This removes, in the broad phase, the troublesome task of updating binary trees and consequently all the costs associated to it.

Chapter 4

Applicability, verification & performance tests

This chapter verifies robustness and stability of the routines implementing the algorithms presented in Chapter 3. Firstly, the Signed Volumes method is illustrated and subjected to a number of tests for DEM and FEM contact analysis. Afterwards, the GJK algorithm and the novel hierarchical search are combined to solve multi-body problems. The results are compared against reference solutions. Together with an analysis of the theoretical costs, these validate the hypothesis underpinning the hierarchical search. A simple example is also presented to highlight the applicability of the routines to the treatment of (concave) NURBS curves and surfaces. Finally, an extensive comparison between the original and the improved GJK algorithm is presented.



(a) (b) (c)

Figure 4.1: Rigid sphere impacting a flat surface described by elements with triangular facet.

4.1 Example of sphere–triangle contact detection

This section presents with an example how the Signed Volumes method (Section 3.1) solves a distance query. The aim is to give a step-by-step explanation of the recursive logic behind `SignedVolumes`, the routine that implements the Signed Volumes method. Particular attention is paid to geometrical interpretation of the sub-routines `S2D` and `S1D` introduced in Chapter 3.

Let us consider the generic scenario presented in Figure 4.1, where a sphere impacts a flat surface represented by triangles. Figure 4.1(a) illustrates a particular time-step of a dynamic simulation in which the surface could either be rigid or deformable, and the triangles could be shell elements or outer facets of tetrahedral elements. For simplicity, only the primitives highlighted in Figure 4.1(b) are tested for collision. Assuming that the surface is at rest and the sphere moves toward it, there will be a time-step in which the primitives have their AABBs overlapping; this is the first time-step that invokes `SignedVolumes`. The objective is therefore to establish whether the sphere and the triangle in Figure 4.1(c) are in contact or not. In what follows, a pair of primitives tested by a contact search algorithm is simply referred as *contact pair*.

The sphere–triangle test computes distance between the primitives, if this is less

Table 4.1: Coordinates of the points in Figure 4.2(a)

Point	x-coordinate	y-coordinate	z-coordinate
s_1	0.45	0.00	−0.60
s_2	−0.55	−0.75	−1.55
s_3	−0.55	0.75	−1.55

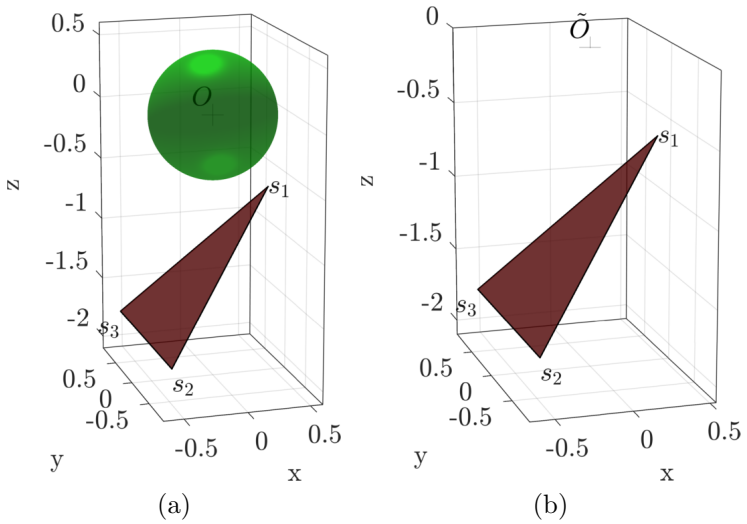


Figure 4.2: Example of sphere-triangle contact test: original (a) and equivalent problem (b).

than zero the contact pair overlaps, otherwise the test is negative. This example considers the primitives illustrated in Figure 4.2(a). The sphere is centred at the origin $\tilde{O}$ and has diameter $\phi = 1$; the triangle has coordinates of s_1, s_2, s_3 given in Table 4.1. The `SignedVolumes` is used to solve the equivalent problem shown in Figure 4.2(b); it computes the minimum distance between $\text{conv}(\{s_1, s_2, s_3\})$ and the origin $\tilde{O}$. Since the sphere is symmetric, it is easy to show that the original contact–pair collides only if this is smaller than ϕ.

The first function invoked within `SignedVolumes` is `S2D`, which begins by projecting the 2-simplex $\tau = \{s_1, s_2, s_3\}$ on the three Cartesian planes to compute the area of

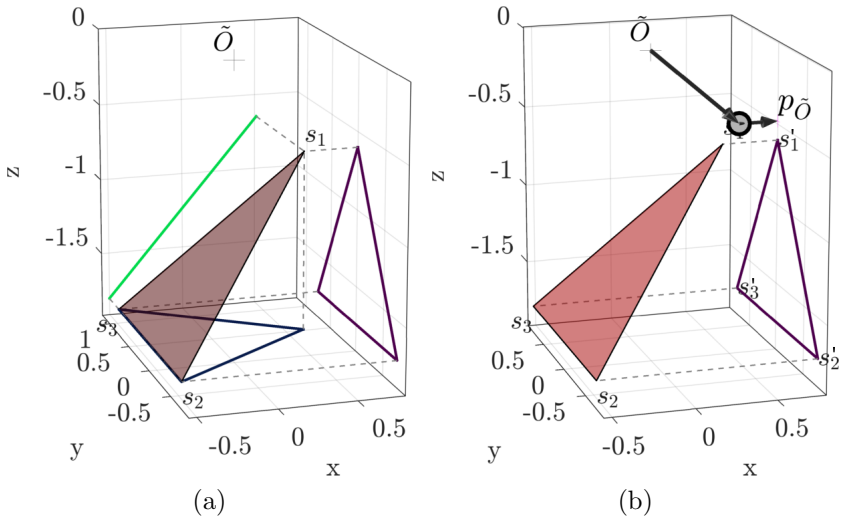


Figure 4.3: Given a point $\tilde{O}$ and a triangle $\{s_1, s_2, s_3\}$ (a), these are projected from 3D to 2D space onto a Cartesian plane.

three triangles. A matrix $\mathbf{M}$ may be defined so that:

$$\det \mathbf{M} = \begin{vmatrix} s_1^x & s_2^x & s_3^x & 0 \\ s_1^y & s_2^y & s_3^y & 0 \\ s_1^z & s_2^z & s_3^z & 0 \\ 1 & 1 & 1 & 1 \end{vmatrix} = C_{1,4} + C_{2,4} + C_{3,4} + C_{4,4}. \tag{4.1}$$

The first three cofactors in Eq. (4.1) correspond to the areas of the triangles projected on yz-plane, xz-plane and xy-plane, respectively. The minor whose absolute value is $\max\{|C_{1,4}|, |C_{2,4}|, |C_{3,4}|\}$ corresponds to the triangle with the largest area.

Figure 4.3(a) illustrates all projections of τ. From the coordinates in Table 4.1 and Eq. (4.1) it can be found that the triangle projected on the yz-plane is the largest one. Furthermore, it should be noted that the area projected onto the xz-plane is a needle of zero area, namely: $|C_{2,4}| = 0$.

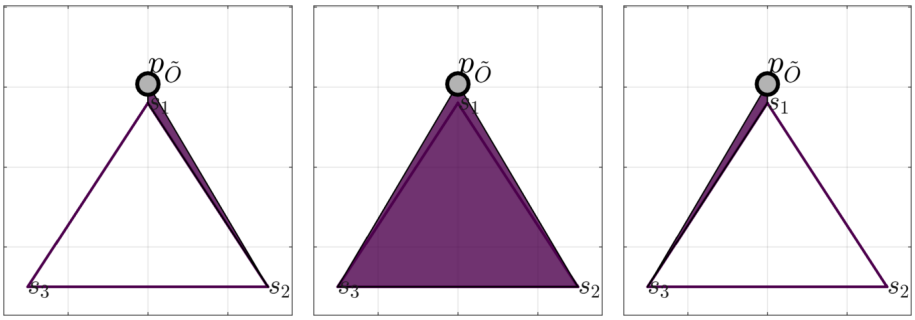
The plane the largest triangle lays on is shown in Figure 4.3(b). The projection of p_O onto the yz-plane is simply obtained by discarding the x-coordinate of p_O and all points in τ. This yields to four new points s'_1, s'_2, s'_3 and $p'_{\tilde{O}}$ illustrated in Figure 4.3(b).

Following the projection step, three 2D fictitious simplices are defined and their

areas compared against $C_{2,4}$. The comparison is carried out with `CompareSigns`, the routine that implements the function defined in Eq. (3.4). Whilst $C_{2,4}$ is known from previous steps, the areas of the fictitious simplices in Figure 4.4 must be computed. From Eq. (3.3) the area of the first, second and third fictitious simplex have sign equal to the following determinants:

$$\begin{vmatrix} s_1^y & s_2^y & p_O^y \\ s_1^z & s_2^z & p_O^z \\ 1 & 1 & 1 \end{vmatrix}, \quad \begin{vmatrix} p_O^y & s_2^y & s_3^y \\ p_O^z & s_2^z & s_3^z \\ 1 & 1 & 1 \end{vmatrix} \quad \text{and} \quad \begin{vmatrix} s_1^y & p_O^y & s_3^y \\ s_1^z & p_O^z & s_3^z \\ 1 & 1 & 1 \end{vmatrix}, \tag{4.2}$$

respectively.



(a) (b) (c)

Figure 4.4: Illustration of the three fictitious simplices obtained by replacing $p_{\tilde{O}}$ to s_3 (a), s_1 (b) and s_2 (c).

For this example, the first and the third determinants in Eq. (4.2) have the sign equal to sign $C_{2,4}$. As can be seen from Figure 4.4, this means that the subset $\{s_2, s_3\}$ does not support the point of τ closest to p_O. The support is sought by invoking `S1D` independently for $\{s_1, s_2\}$ and $\{s_1, s_3\}$.

The routine `S1D` essentially repeats all steps of `S2D` for the 1-simplices $\{s_1, s_2\}$ and $\{s_1, s_3\}$. It begins by projecting O' onto the affine hull of a simplex, and eventually descends from $\mathbb{R}^3$ to $\mathbb{R}^1$ by projecting all points and O' onto the "safest" axis of the Cartesian coordinate system. The result of this procedure is illustrated in Figure 4.5 for $\{s_1, s_2\}$ only, but `SignedVolumes` repeats these steps for $\{s_1, s_3\}$.

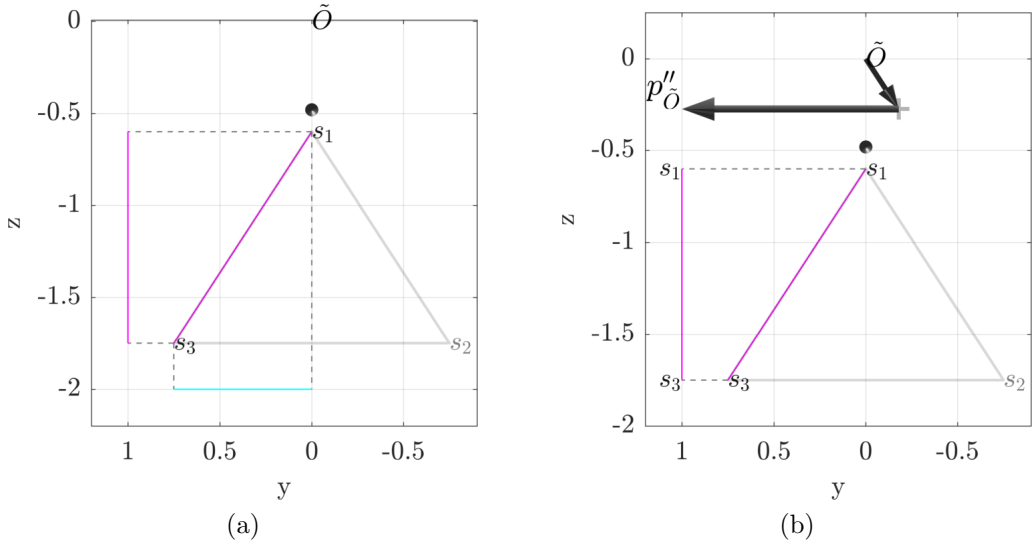


Figure 4.5: Projection of a fictitious simplex from 2D (a) to a 1D Cartesian axis (b).

Finally, the original distance query is solved: both calls to `S1D` return s_1 as the closet point of the simplex to O', and therefore the `SignedVolumes` terminates.

4.2 Remarks on numerical robustness

A crucial part of the Signed Volumes algorithm is the computation of $\det \mathbf{M}$, its sign in particular. This section studies the effect of rounding error for degenerate simplices, i.e. when the points of a simplex are affinely dependent.

The limited accuracy of floating-point arithmetic makes the evaluation of the determinant of small-rank matrices a difficult operation. Modern processor units, CPUs and GPUs, commonly use three levels of precision to represent floating-point numbers: *half-precision* (16 bits), *single precision* (32 bits) and *double precision* (64 bits). These represent a real number by:

$$\pm\, m \times \beta^e \tag{4.3}$$

where the mantissa (or significand) $\pm\, m$ has length p and the signed integer exponent e

is usually represented in binary $\beta = 2$ base by q digits. The widely accepted IEEE 754 standard sets $p = 24$ for single-precision and $p = 53$ for double-precision and defines any exceptions which cannot be represented by the format in Eq. (4.3).

The two types of error encountered in floating-point arithmetic are *absolute* and *relative*. The discrepancy between an exact representation x and its floating-point approximation $\tilde{x}$ is the *machine epsilon* $\varepsilon = \frac{1}{2}\beta^{1-p}$. This is an absolute bound to the accuracy.

The relative error occurs when evaluating expressions, and this can be several order of magnitude larger than ε. Let us consider two arithmetic values $\frac{x_1 - \tilde{x_1}}{x_1} = \eta_1$ and $\frac{x_2 - \tilde{x_2}}{x_2} = \eta_2$. The relative error resulting from the difference $\gamma = \eta_1 - \eta_2$ can be several order of magnitude larger than ε. This is known as *cancellation* and its effect can be catastrophic for projection point problems such as those presented in (26, 63, 104, 156, 168, 173, 256). As already mentioned in Chapter 2, cancellation is overlook in all these previous studies.

Computing the determinant of a matrix is an operation extremely prune to cancellation error. A solution is to evaluate the determinant using a formulation that is less likely to trigger cancellation. For example, the determinant of 3-by-3 matrix may be written as:

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a(ei - fh) - b(di - fg) + c(dh - eg) \tag{4.4a}$$

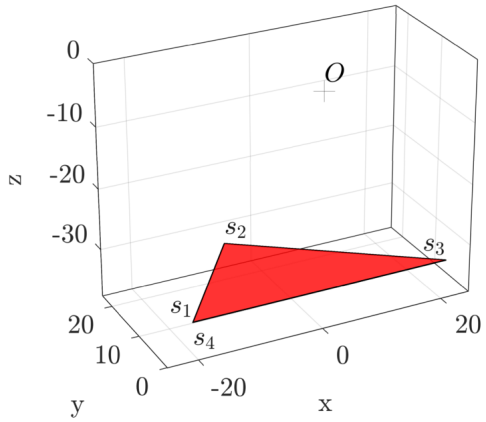
$$= aei + bfg + cdh - ceg - bdi - afh. \tag{4.4b}$$

Although computationally more efficient, Eq. (4.4a) is more prune to cancellation than Eq. (4.4b), thus the latter should be preferred.

Let us investigate the behaviour of `SignedVolumes` when using double-precision when dealing with a degenerate simplex. Consider the 3-simplex $\tau = \{s_1, s_2, s_3, s_4\}$ depicted in Figure 4.6. The aim is to find the set W supporting the point of minimum norm

$$\nu(\tau) = \left\{ \sum_{i=1}^{|W|} \lambda_i s_i : s \in \tau \ , \ i \in W \ , \ \lambda > 0 \ , \ \sum_{i=1}^{|W|} \lambda_i = 1 \right\}. \tag{4.5}$$

The coordinates of all vertices s_i are also included in Figure 4.6.



Point	x-coordinate	y-coordinate	z-coordinate
s_1	-20.713282046795065	0.00	-31.557752817397152
s_2	-6.3505362624669246	19.715748586578577	-31.557752817397152
s_3	20.713255776709875	-0.032989069061845351	-31.557752817397152
s_4	-20.713282046795065	0.00	-31.557752817397152

Figure 4.6: Example of degenerate 3-simplex.

By following the procedure presented in Chapter 3, the reader can verify that by solving Eq. (4.4b) with double-precision, $\mu(\tau) = -3.637\,978\,807\,091\,713 \times 10^{-12}$. This triggers, from `CompareSigns`, the search in three facets of τ with `S2D`, which is invoked independently to test $\{s_1, s_2, s_3\}$, $\{s_1, s_2, s_4\}$ and $\{s_1, s_3, s_4\}$. The three calls to `S2D` guarantee that the correct answer is found, but at an extra computing effort.

The cancellation error yields to extra search operations which may be easily visualised tracking how `SignedVolumes` descends the Hasse diagram of a 3-simplex. Fig-

ure 4.7 shows all Voronoi regions of τ and highlights the solution $V_{1,2,3}$ in green. An arrow is drawn every time one of the functions `S3D`, `S2D` or `S1D` is invoked. Apart from the arrow linking $V_{1,2,3,4}$ with $V_{1,2,3}$, all the others report unnecessary search operations that should be avoided.

This example has shown that computing the determinant with Eq. (4.4b) is not sufficient to prevent cancellation from compromising the Signed Volumes method. This problem is well-known in the literature and is also known to affect other applications in computational mechanics, such as mesh generation (77) and X-FEM (216).

A more robust alternative for computing the determinants is to design a software library that enables arbitrary precision. Exact arithmetic (76), multi-precision arithmetic (75) and adaptive floating-point arithmetic (183, 209) are the most successful solutions published to date. Exact arithmetic libraries can operate on either integers or doubles numbers, but these have high memory and processing costs. Multi-precision libraries achieve arbitrary precision by storing numbers in a *multiple-digit* format (75); the major drawback to this solution is the portability of such libraries. A fast and flexible solution is the adaptive precision floating-point library for geometric operations presented in (209). This substitutes the multiple-digit format with a *multiple-component* format which stores numbers as a sum of ordinary floating-point. Arbitrary precision is achieved by adding more terms a posteriori only, if a particular arithmetic operation requires it.

Adaptive floating-point arithmetic allows to compute $\operatorname{sign} \det \mathbf{M}$ exactly and, consequently, it increases the effectiveness of the search operation dictated by `CompareSignes`. Let us verify this by repeating test in Figure 4.6 with the geometrical predicates presented in (209).

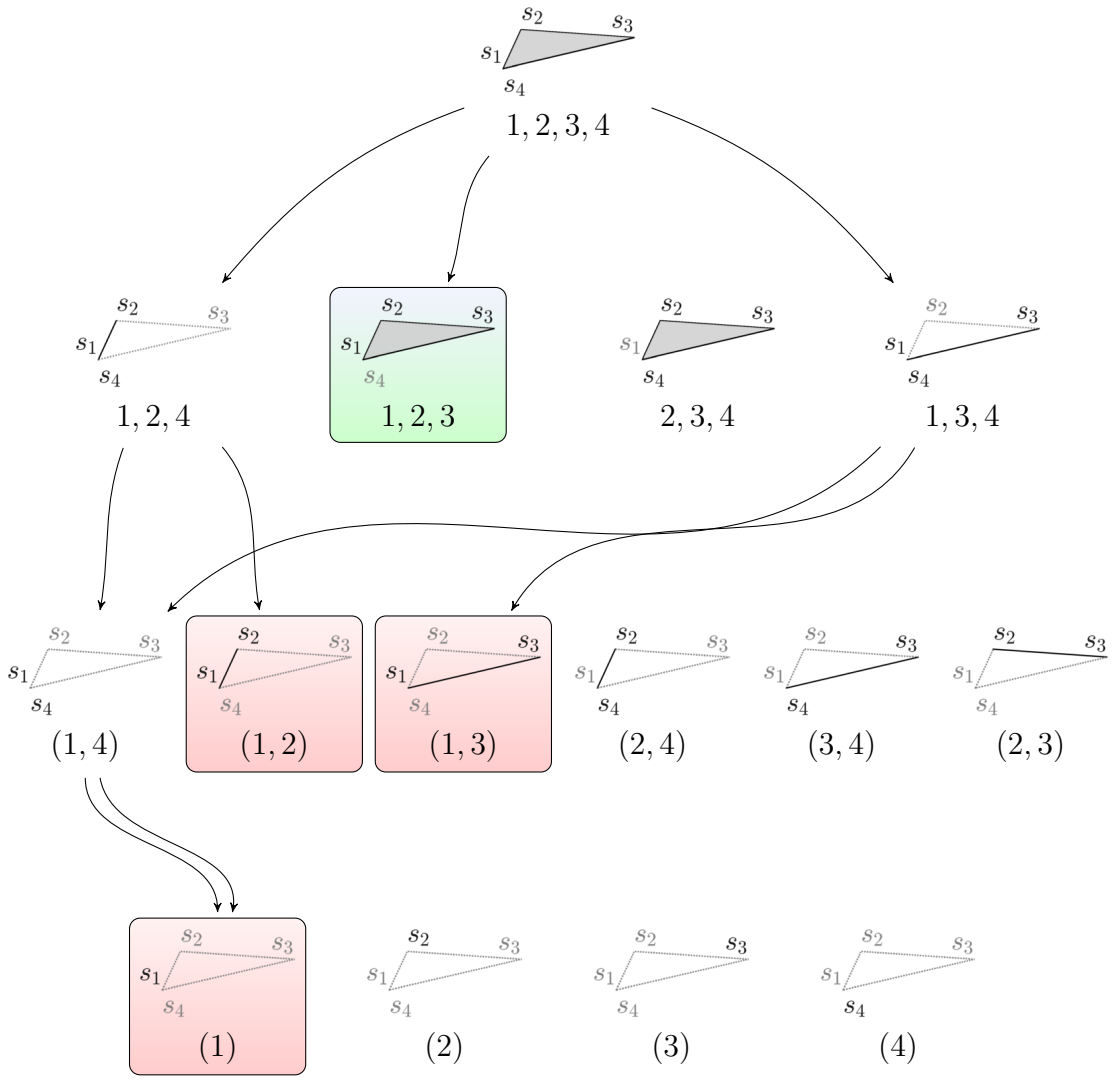


Figure 4.7: Hasse diagram and search path of the `SignedVolumes` algorithm showing the effect of cancellation. Without adaptive-floating point arithmetic the algorithm does not select the shortest path to find the exact solution which is highlighted in green.

Whilst with double-precision the volume of τ was negative, with adaptive arithmetic this is exactly null, i.e. $\mu(\tau) = 0$. Furthermore, only one of the minors computed in `S3D` is non-null. For the current example, the only arrow drawn on the Hasse diagram links $V_{1,2,3,4}$ with $V_{1,2,3}$, demonstrating that adaptive arithmetic drastically reduces the number of operations when `SignedVolumes` deals with degenerate simplices.

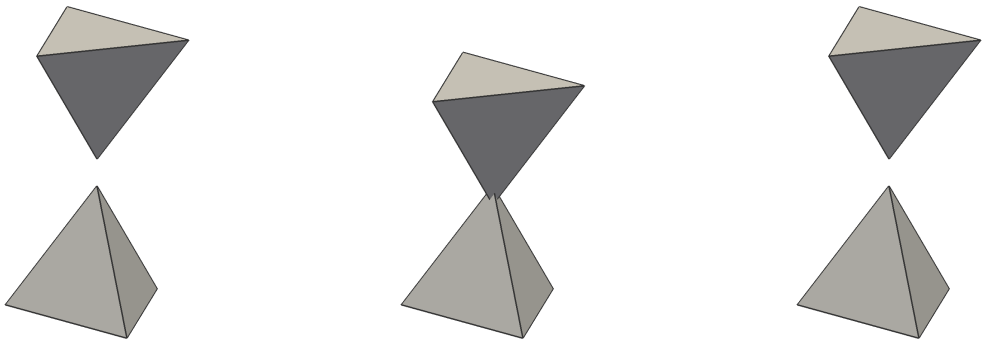
Numerical experiments on degenerate simplices show that the total CPU time of `SignedVolumes` is reduced with adaptive floating-point arithmetic. However, for sim-

plices with affinely independent points, standard double-precision arithmetic is 10% to 15% faster. From this point of view, the contribute of adaptive floating-point arithmetic is therefore negligible. There are applications however in which the higher accuracy of adaptive arithmetic brings tangible benefits, one application is discussed in Chapter 6.

Overall, the test presented in this section investigated the effect of cancellation of the routine which implements the Signed Volumes method. It has been shown that adaptive floating-point arithmetic is beneficial for degenerate simplices; although cancellation on double-precision arithmetic does not affect significantly the final solution, it does make the computation more expensive by increasing the number of operations. As rule of thumb, the use of adaptive arithmetic in `SignedVolumes` is recommend for applications whose results is heavily influenced by the outcome of the distance query, e.g. optimisation problems.

4.3 Simple contact benchmarks for rigid and deformable elements

This section presents three verification tests for discrete element method (DEM) and finite element method (FEM). These are a selection of a larger benchmarking suite created with the aim of verifying the correct implementation of the algorithms described in Chapter 3. The benchmarks focus on narrow contact search and resolution of single-element contact pairs. All elements have the same material properties: Young modulus $E = 2.1 \times 10^8$ MPa and density $\rho = 7.85 \times 10^3$ kg m^{-3}. The material is considered isotropic linear elastic; furthermore, for simplicity, the contact is modelled by penalty formulation and friction is ignored.



(a) $t = 0.0$ s (b) $t = 0.9$ s (c) $t = 1.8$ s

Figure 4.8: Solution of contact benchmark between two deformable tetrahedral finite elements.

4.3.1 Test two deformable tetrahedral elements

This test involves two tetrahedral finite elements under gravitational load. An element has its base supported in the vertical direction, the other is free to fall and initially at rest. This benchmark aims to verify the correct implementation of contact routines for triangular facets.

The solution is shown in Figure 4.8 at three time-steps: $t = 0.0$ s, $t = 0.9$ s and $t = 1.8$ s. Because of the ideal contact conditions, it is expected that the falling element, after collision, bounces back to its original position. This requires a robust implementation of the geometrical routines from which the contact forces is calculated.

The small overlap visible in Figure 4.8(b), due to the penalty formulation, persists for several time-steps; should the resultant contact force be asymmetric for any time-step, the top tetrahedron will not bounce back vertically. Notice that the resultant contact force is the sum of the contributes from six contact pairs.

For verification purpose, three independent quantities are monitored during the simulation: spatial coordinates, kinetic energy and ground reaction force. The vertical coordinate of the vertex at the bottom of the falling tetrahedron has an almost perfect

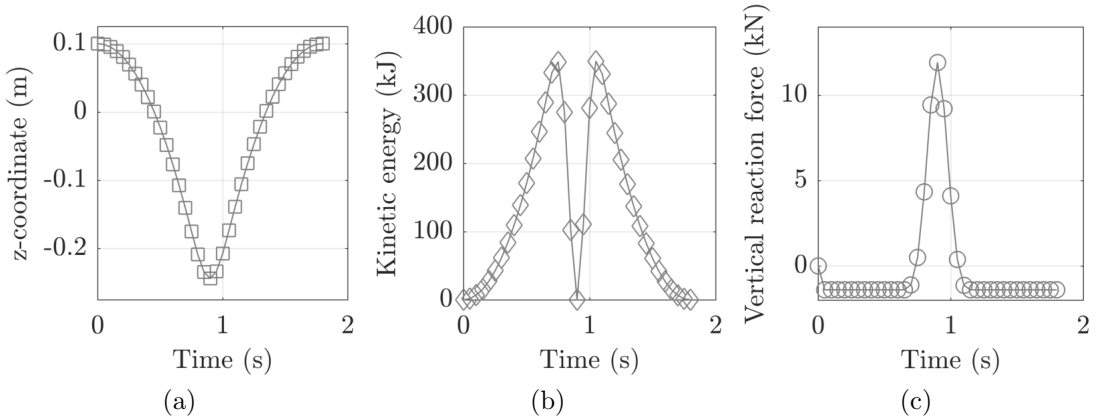


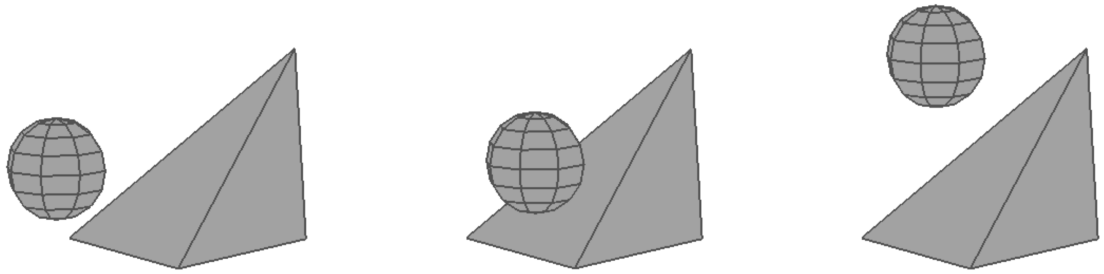
Figure 4.9: History of node coordinate (a), kinetic energy (b) and contact reaction force (c) for contact benchmark between tetrahedrons in Figure 4.8.

symmetrical history, as shown in Figure 4.9(a). The initial and final position along the z-axis of this node are within a small tolerance. Figure 4.9(b) shows the history of the kinetic energy in the system. Like before, the curve exhibits nearly perfect symmetry before, after and during collision. Finally, the resultant of vertical forces acting on fixed tetrahedron are reported in Figure 4.9(c).

The consistency of the results in Figure 4.8 gives some confidence about the contact detection and resolution algorithms for tetrahedral elements. More benchmarks, omitted from this thesis, were carried out to verify other scenarios, such as edge–to–edge, face–to–face and edge–to–face. They have all returned consisted results, and therefore it can be concluded that the implementation is accurate and robust.

4.3.2 Test rigid sphere and deformable tetrahedron

The second test considers a rigid sphere moving toward a grounded tetrahedron with velocity v_0. Gravity is ignored and only the nodes at the bottom of the tetrahedron are constrained. The aim is to validate the implementation of the routines for sphere–triangle tests.



(a) $t = 0.0$ s (b) $t = 5 \times 10^{-1}$ s (c) $t = 1.0$ s

Figure 4.10: Sphere-tetrahedron contact test.

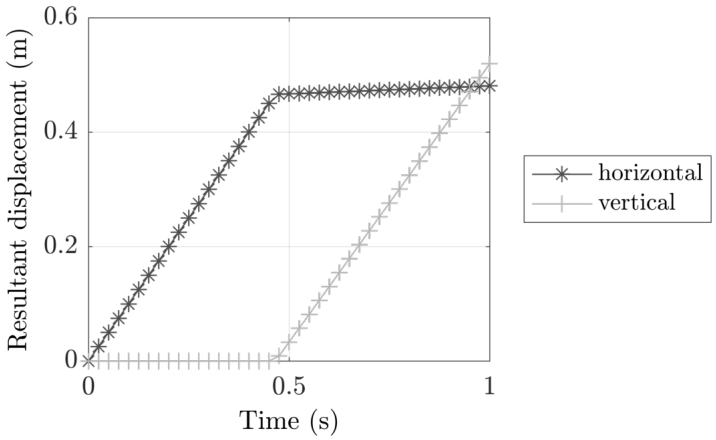


Figure 4.11: Horizontal and vertical displacement of rigid sphere.

The solution is shown in Figure 4.10 at three time-steps: $t = 0.0$ s, $t = 0.5$ s and $t = 1.0$ s. Because the impact occurs on a face inclined at 45°, it is expected that the sphere converts, after collision, its motion from horizontal to vertical. This requires a robust implementation of the geometrical routines from which the contact forces is calculated.

The displacement history of the sphere is shown in Figure 4.11. The graph shows that, after collision, the sphere retains some of its initial horizontal displacement (the horizontal resultant accounts for the contribution from u_x and u_y components), this is because the tetrahedron element is deformable. By repeating this test with spheres of smaller size, it was found that horizontal displacement tends to remain constant after

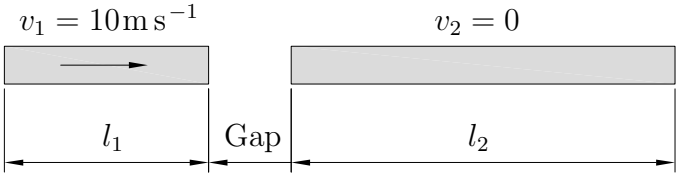


Figure 4.12: Impact of two elastic bars.

collision.

Like in Section 4.3.1, this benchmark is only a particular scenario of the many tested for validating the routines. Other tests included spheres of various size impacting an edge or a vertex of the tetrahedron and, since all tests returned consistent results, it can be concluded that the implementation is accurate and robust.

4.3.3 Test two deformable hexahedral elements

One of the most technically important problems in impact analysis is the longitudinal collision between collinear rods. This is usually studied for analytical and experimental applications, but here is used to investigate the effect of the penalty factor ϵ on numerical analysis of impacts. See (247) for more details on the contact formulation.

Let us consider two rods illustrated in Figure 4.12. Their displacement, respectively u_1 and u_2, may be described by a one-dimensional travelling stress wave:

$$EA\frac{\partial^2 u_1}{\partial x^2} = -\rho A\frac{\partial^2 u_1}{\partial t^2} \quad \text{and} \quad EA\frac{\partial^2 u_2}{\partial x^2} = -\rho A\frac{\partial^2 u_2}{\partial t^2}. \tag{4.6}$$

The rods have identical material (density ρ and Young modulus E) and area A, but one is initially at rest, the other travels at velocity v_1. Eq. (4.6) requires the solution of a boundary value problem. The well-known analytical solution, derived by d'Alembert, superimposes two stress waves in each rod. A wave travels with velocity

$c = \sqrt{\frac{E}{\rho}}$, the other $-c$. Typically this is written as $u_1(x,t) = f_1(x-ct) + g_1(x+ct)$ for the fist rod, and $u_2(x,t) = f_2(x-ct) + g_2(x+ct)$ for the second. The solution is illustrated in Figure 4.13. Initially, both bars are stress free (Figure 4.13(a)). The bars do not separate when $v_1 \neq v_2$ at the interface (Figure 4.13(b)), but when $\sigma > 0$ at the interface (Figure 4.12(c)). This is because the bars remain in equilibrium while colliding. After collision the left-hand bar is at rest, whilst the other departs maintaining some oscillations (Figure 4.12(d)).

From the solution of Eq. (4.6), the time of impact t_{impact} may be computed and is here compared against a FEM simulation. Each bar is modelled by a single hexahedral FEM so that the geometry reflects Figure 4.12. Due to the penalty formulation the bars overlap and the gap is negative. The solver employs a penalty factor ϵ to apply contains and this heavily influences t_{impact}. On the one hand, for low values of ϵ the bars may never separate, on the other hand, for excessive values of ϵ they may bounce before moving apart. Therefore, ϵ needs to be calibrated.

For length $l_1 = 1\,m$, the analytical solution for the impact time is $t_{\text{impact}} = 0.0245$ s, however, only a small range of values of ϵ predicts it accurately. The comparison between numerical and analytical results is shown in Figure 4.14. In this experiment, the value 4.0×10^8 returns an accurate estimate of t_{impact}. While the elements overlap, the contact force preserves equilibrium, afterwards the gap opens and the bars move apart. The final velocity is also predicted accurately by all ϵ within the range in Figure 4.14. Higher values of ϵ caused the rods to bounce repetitively before t_{impact}.

This benchmark provided a mean to calibrate the penalty factor ϵ and showed its influence over t_{impact}. The value of ϵ found is in line with what advised in (247).

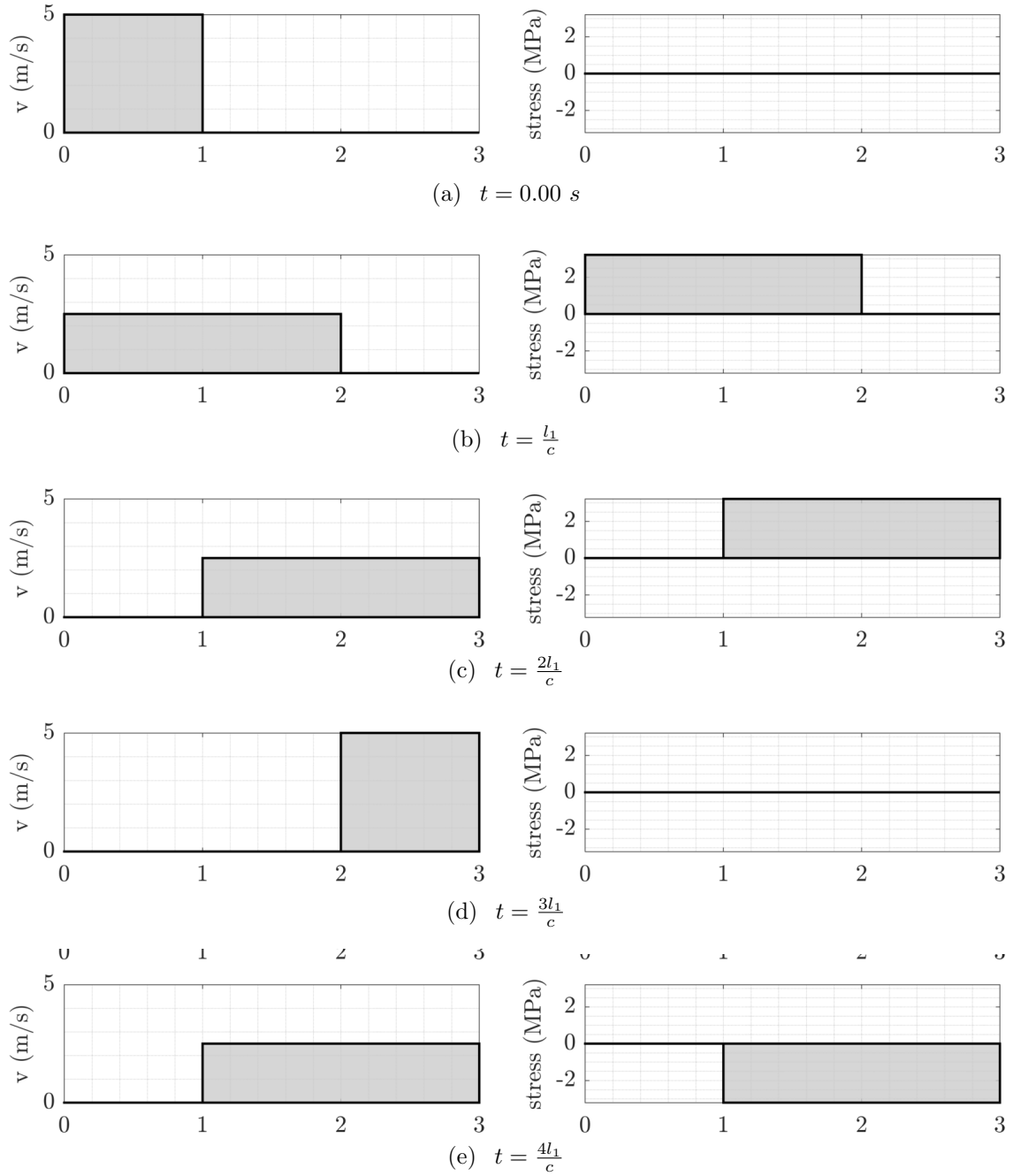


Figure 4.13: Analytical solution of velocity (a) and stress (b) for collinear impact problem.

4.4 Free falling particles benchmark

The aim of this test is to validate the logic of the new hierarchical contact search presented in Chapter 3. To do so, initially the bodies are all separated, and they repeatedly come into contact during the simulation.

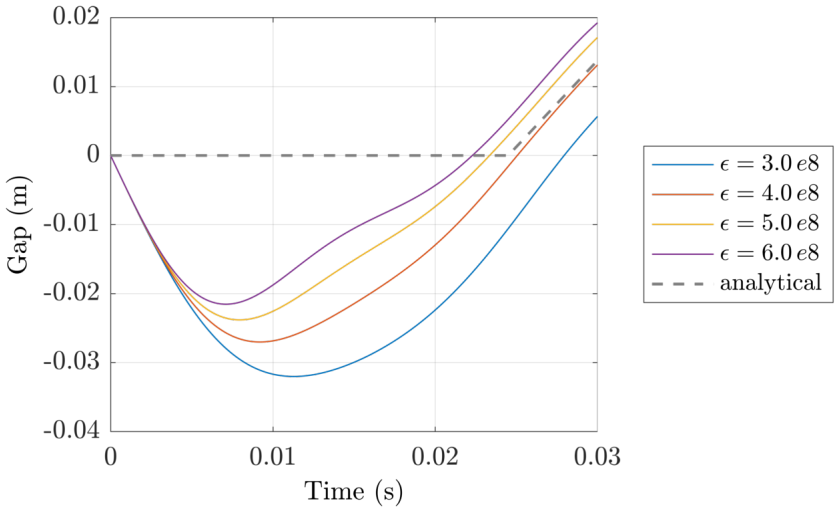
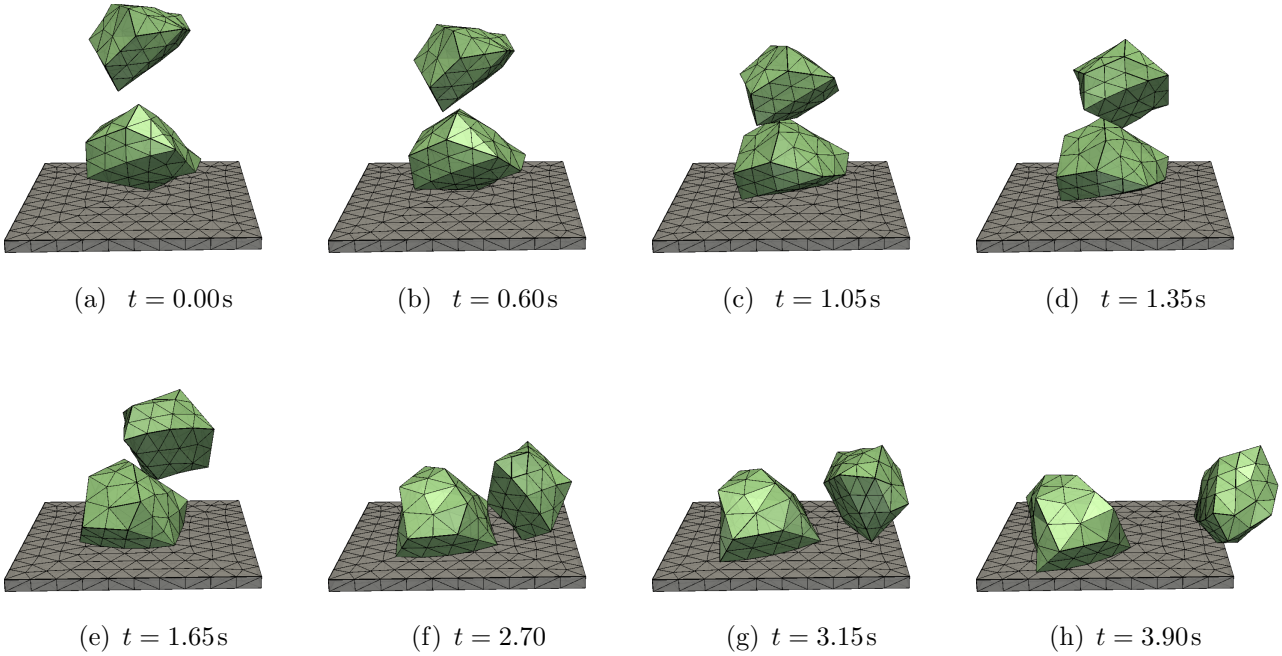


Figure 4.14: Influence of the penalty factor on time of impact.

For simplicity, only three convex bodies are considered: a fixed plane and two sand grains falling under gravitational load. All these are described by tetrahedral finite elements and the contact penalty formulation is used. The grains are rigid, modelled with density 2.65 $\mathrm{kg\,m^{-3}}$ and fall under gravitational force whilst initially at rest. The shape of the grains is reproduced from (222), whilst the plane is a square with edge length of 0.5 mm.

The three bodies are represented in Figure 4.15 at different time-steps. At time $t = 0.00\ s$ all bodies are separated. After few time-steps, the bottom grain collides with the plane. The top grain however comes in contact only at approximately $t = 1.00\ S$, see Figure 4.15(a)-4.15(c). Afterwards, it rolls over the bottom grain, reaches the plane, and falls off it. The simulation terminates at time $t = 4.00\ S$ after about 45000 time-steps.

If self-contact is ignored, the three bodies form three *body-contact-pairs*. The distance between each pair is computed by means of the algorithms presented in Chapter 3 and illustrated in Figure 4.16. Each row pictures a time frame and each column highlights a specific body-contact-pair. When a pair of bodies is separated, the separating



(a) $t = 0.00\text{s}$ (b) $t = 0.60\text{s}$ (c) $t = 1.05\text{s}$ (d) $t = 1.35\text{s}$

(e) $t = 1.65\text{s}$ (f) $t = 2.70$ (g) $t = 3.15\text{s}$ (h) $t = 3.90\text{s}$

Figure 4.15: Two convex grains falling on a plane under gravitational load.

vector is shown. See for example Figures 4.16(a)-4.16(d). When a pair of bodies is in contact, the overlapping facets are highlighted in red and green colour. See for example Figures 4.16(e), 4.16(h).

The GJK algorithm computes, for each body-contact-pair, the separating vector. As described in Section 3.3, to save computing effort, this is not recomputed at each time-step. Instead, the distance d between a pair of bodies is recomputed only when the estimated distance $\tilde{d}$ becomes null or negative.

The result of this benchmark shows that the inequality $d \geq \tilde{d}$ is always verified, hence the logic of the contact search is sound. Figure 4.17(a) shows the history of the squared distance between the top grain and the plane; the distance is a smooth and continuous curve which is never exceeded by $\tilde{d}$. Similarly, Figure 4.17(b) represents the same quantities for the body-contact-pair composed by the two grains. The fact that $\tilde{d}$ has an oscillatory trend does not compromise the accuracy of the solution. To minimise the computing time, however, the number of oscillations of $\tilde{d}$ should be as low

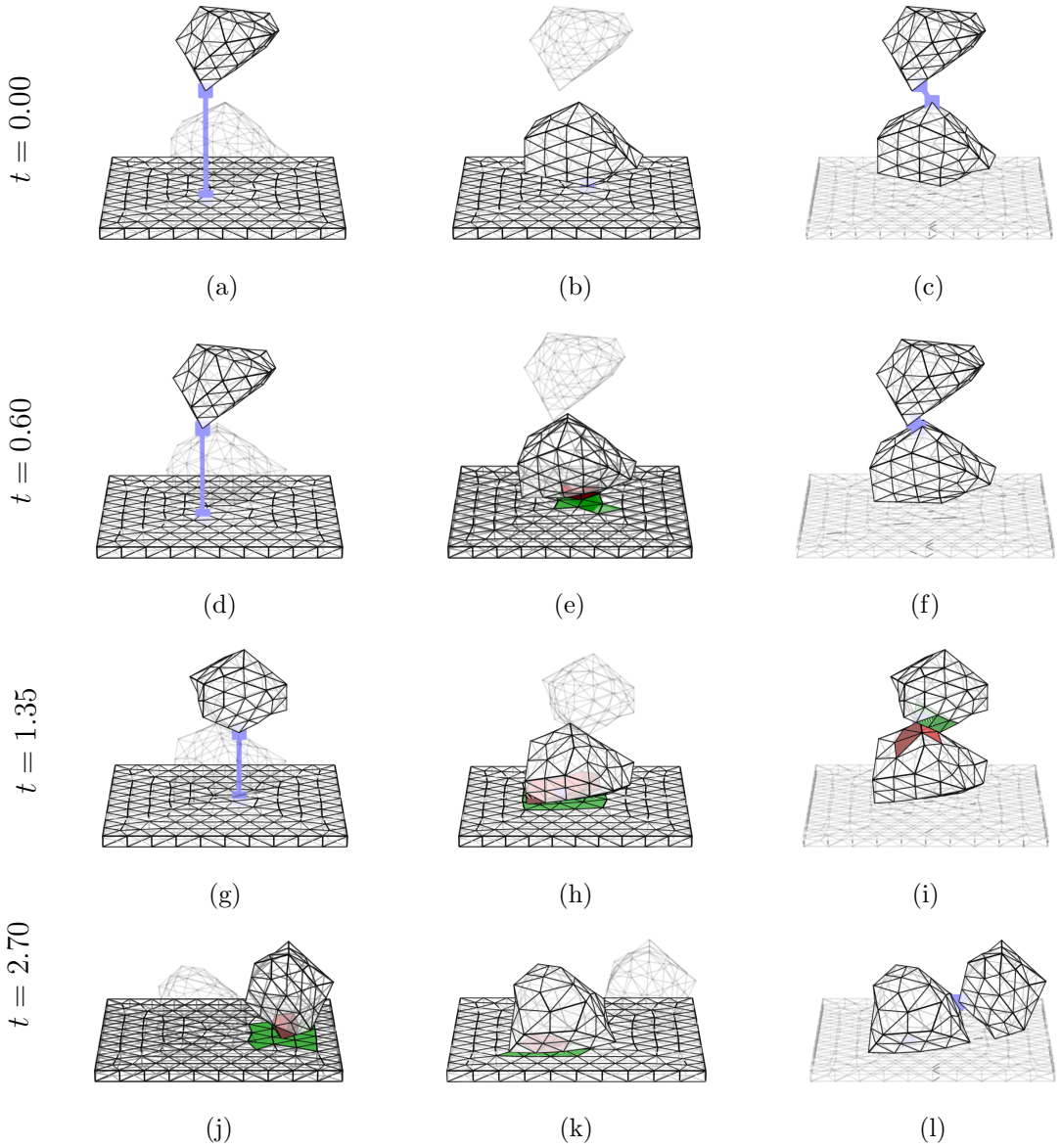
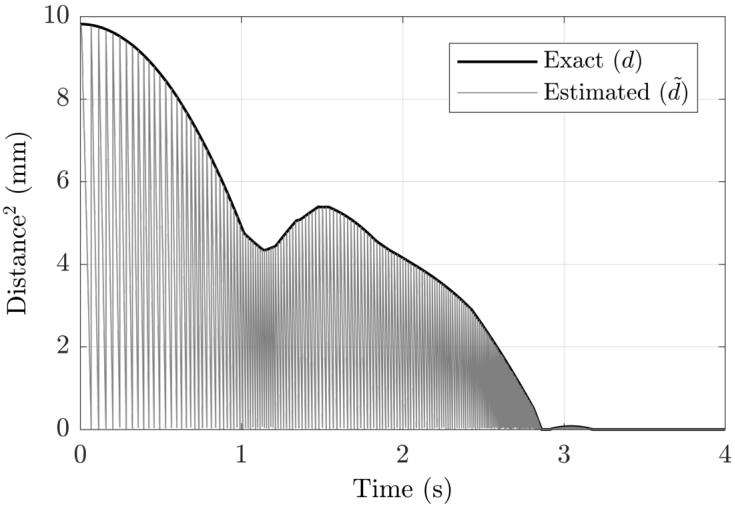


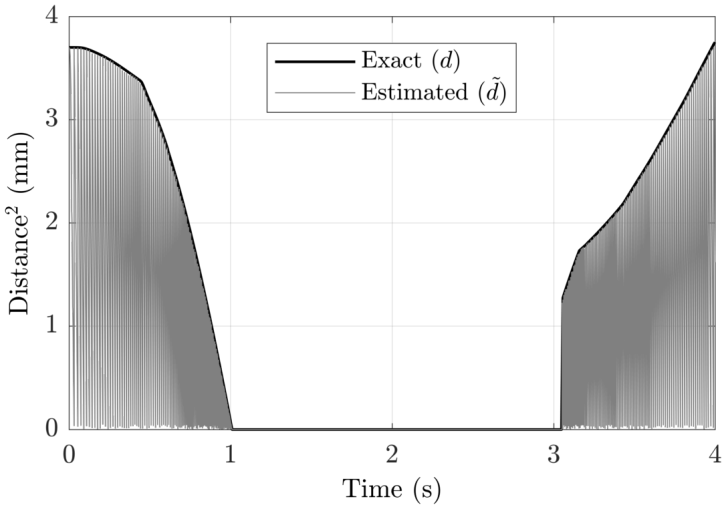
Figure 4.16: Contact search for each body-contact-pair at $t = 0.0$, $t = 0.6$, $t = 1.35$ and $t = 2.70$. The rows represent different body-contact-pairs at a given time: (a)-(c) $t = 0.0$, (d)-(f) $t = 0.6$, (g)-(i) $t = 1.35$ and (j)-(l) $t = 2.70$.

as possible. The definition of $\tilde{d}$ proposed in Section 3.3 allows to reduce the number of oscillations and, consequently, the number of calls to the GJK algorithm.

In this benchmark, to compute d at each time step, a brute-force approach would have called the GJK algorithm 60734 times, however, the proposed approach invokes it only 2769 times, reducing the number of calls by 95%. This proves that, despite Figure 4.17 shows a rather dense pattern for $\tilde{d}$, the problem can be solved accurately in a sustainable number of calls to the GJK algorithm.

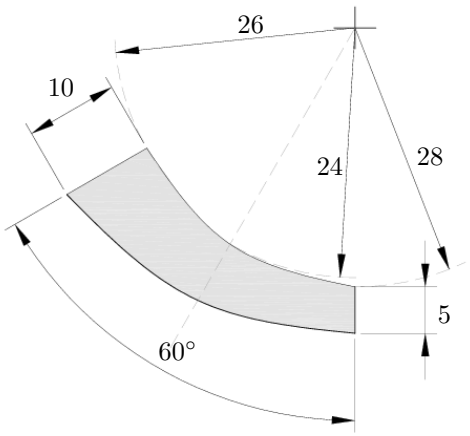


(a) Distance between top grain and plane.



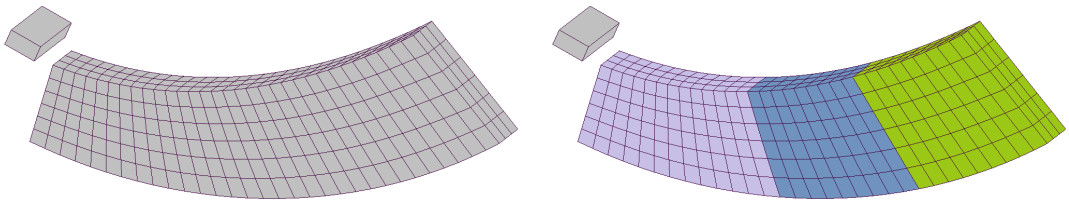
(b) Distance between top and bottom grain.

Figure 4.17: Evolution of exact and estimated distances for the grains in Figure 4.15.



(a)

Figure 4.18: Dimension of the geometry used in the ironing benchmark.



(a) (b)

Figure 4.19: Geometry (a) and approximate convex decomposition (b) of large sliding benchmark.

4.5 Ironing benchmark

This section introduces a new ironing problem, in which a single–element body slides on top and a concave body with dimensions illustrated in Figure 4.18 and mesh depicted in Figure 4.19(a). This benchmark is somehow similar to the ironing problem presented in (139), expect that the concave body makes the contact detection task more difficult.

The problem considers two deformable and unconstrained bodies. An initial angular velocity $1\frac{\mathbf{rad}}{\mathrm{sec}}$ is given to the single-element body, so that this gently slides over the whole surface of the concave body. Both bodies are free to move in space and the material properties are the same as in Section 4.3.1. Because of the large displacement,

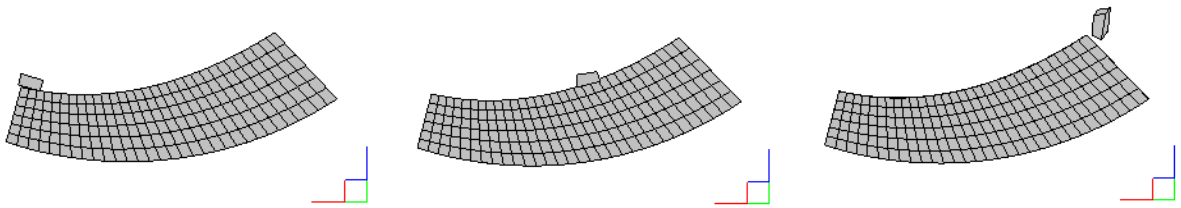
a standard binary tree search may be not efficient to solve this problem and is here replaced by the new algorithmic framework presented in Section 3.3.

Unlike the benchmarks presented thus far, this model involves a concave body which has to be decomposed in approximatively convex sub-bodies. The result of such decomposition is shown in Figure 4.19(b). This generates three approximatively convex sub-bodies that are passed to the hierarchical contact search. If self-contact pairs are ignored, the decomposition forms three body-contact-pairs.

Let us recall from Chapter 3 that the objective of the approximate convex decomposition is to facilitate the narrow contact search by generating smaller well-balanced binary trees. By adopting multiple trees of smaller size, it becomes easier to keep the trees balanced. Furthermore, less memory is needed if the contact regions is a restricted part of the domain.

The solution to this benchmark is shown in Figure 4.20 at different time steps. The moving object approaches the top-surface almost tangentially (Figure 4.20(a)) and a small overlap is maintained by the penalty formulation during contact (Figure 4.20(b)). The sliding object begins to revolve around its centre of mass as it departs from the other object, the revolution however appear to be off-plane (Figure 4.20(c)). After further investigation, it was concluded that this unexpected behaviour is not related to the contact search, but dependent on the penalty factor and on the time-step size. The former affects the contact force and hence the time at which the sliding object is repulsed; the latter influences the accuracy of stress computed within the sliding element as it begins to spin.

The benchmarks is repeated using a classic binary tree search and the hierarchical contact search presented in Chapter 3. It is found that the kinetic energy of the system



(a) (b) (c)

Figure 4.20: Time-frames of the solution of the large sliding benchmark at time 12 ms (a), 20 ms (b) and 24 ms (c).

is identical when computed with the two approaches. Similarly, the barycentre of the sliding object tracks the same trajectory in both cases. The graphs depicted in Figure 4.21 presents these results. The comparison considers the ADT (25), a classic binary tree search for contact mechanics problems, and the newly prosed method involving approximate convex decomposition. Notice that, despite the spinning highlighted in Figure 4.20(c), the horizontal coordinate of the barycentre of the single-element obtained with the two methods is identical. From this, it can be concluded that the results are not affected by the implementation of the contact search algorithms.

The comparison between the two methods match perfectly, demonstrating that the implementation is accurate and correct. However, this benchmark is too small to observe tangible reduction of CPU time; Chapter 5 presents an application in which a significant reduction of computing time is achieved with the new method.

4.6 Scalability of contact algorithms

This section assesses the applicability of the hierarchical contact search presented in Section 3.3 by comparing its theoretical cost against other methods widely used for engineering simulations. The comparison focuses on the narrow search phase, which is

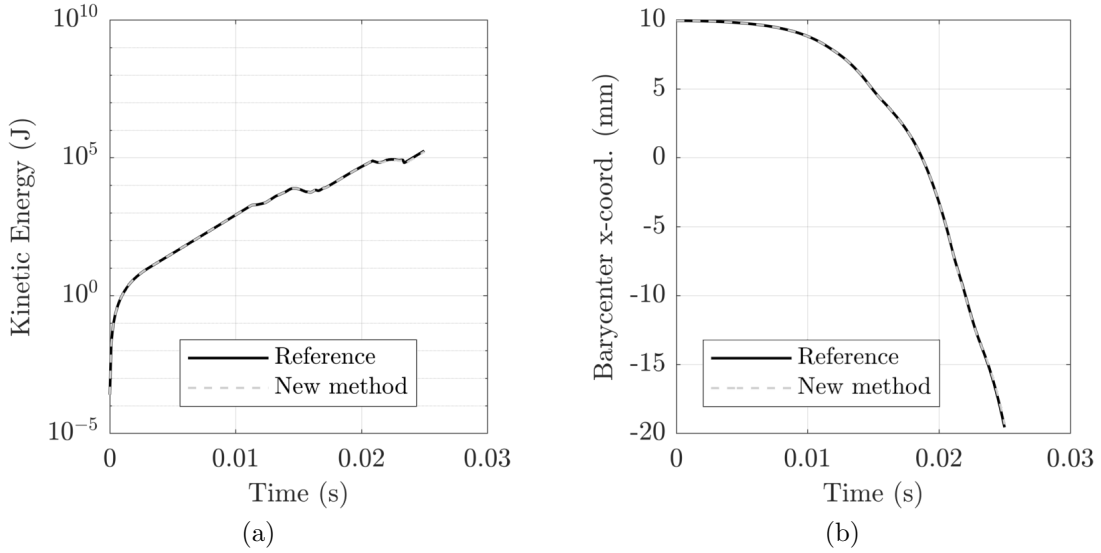


Figure 4.21: Result comparison for sliding benchmark: kinetic energy (a) and single-element coordinate of barycentre (b).

governed by the binary trees.

Of particular interests are the memory and number of operations needed for building and traversing a tree (the traversing operation corresponds to solve the intersection problem). These must be within the hardware capacity to enable the solution of large simulations. Furthermore, it should be recalled that the key difference between common methods and the new one is that the latter requires to build a binary tree for each sub-body resulting from the approximate convex decomposition.

Let us begin by describing the worst-case scenario; that is, the object configurations which require the highest number of operations to solve the intersection problem. Consider a computational domain in $\mathbb{R}^n$ with m approximately convex bodies, each one of these has l outer facets. The worst-case configuration is realised if the m bodies are mutually in contact. This is because the new method would build and traverse $\frac{m(m-1)}{2}$ binary trees, one for each body-contact-pair.

The number of binary trees is therefore proportional to $\mathcal{O}(m^2)$, and significantly

higher than common approaches, but it exists an upper limit to m. A particular case of this worst-case configuration was studied by Descartes (119), who formulated a theorem which states that only $n + 2$ spherical objects can mutually contact each other. That is: only 4 disks in $\mathbb{R}^2$, or 5 spheres in $\mathbb{R}^3$, can be assembled tangent to each other. Although there is no mathematical proof, it is reasonable to assume this limit holds for approximately convex bodies even if non-spherical.

This analysis assumes $m \leq 6$, which is a pessimistic scenario. To account for non-convex bodies and for the overlap due to penalty contact formulations. A further assumption is that the number of operations for building and traversing a single binary tree is in the order of $N \log N$, where $N = m \cdot l$ is the number of items to be listed in the tree. This is in agreement with most of binary trees (64); for example, the ADT presented in (25) has cost $\mathcal{O}(N \log N)$. Based on these assumptions, it is possible to estimate the theoretical cost of the hierarchical contact search and compare it against the standard ADT.

The cost for building and traversing a standard binary tree amounts to $2N \log N$, assuming that m contacting bodies, with l facets each, are listed in a single tree.

The new hierarchical contact search decreases the building costs to $m\, l \log l$ since each tree has only l listed items. The traversing cost however scales quadratically with the number of trees, namely: $\frac{m(m-1)}{2}\, l \log l$. This suggest that the newly proposed algorithm may result not sustainable for large values of m, but it would result cheaper than existing methods if m is sufficiently small.

The analysis that follows aims to quantify the threshold value of m for which the new and other methods have similar costs.

Figure 4.22 illustrates the worst-case scenario costs of a standard ADT and com-

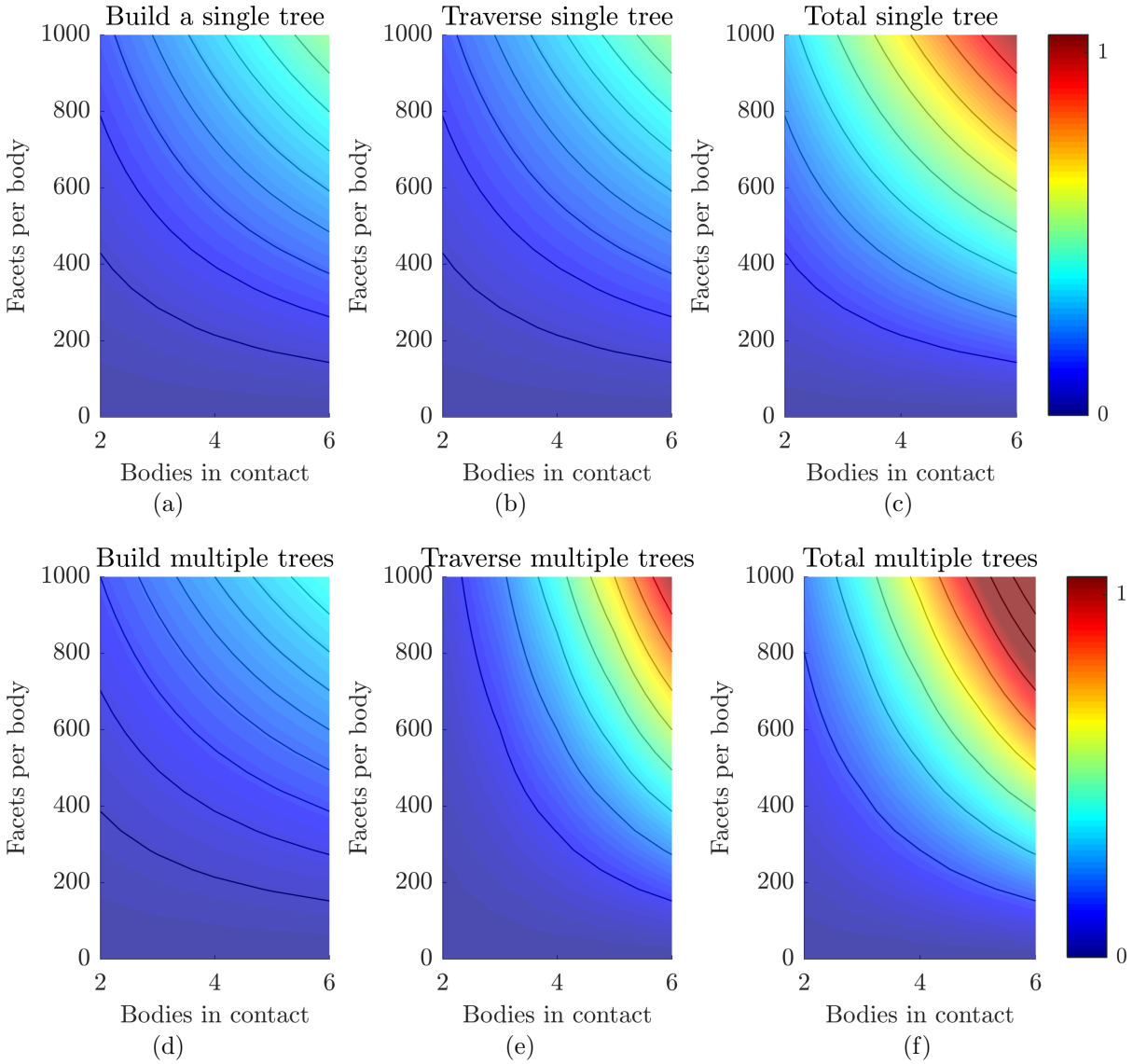


Figure 4.22: Normalised cost functions for building (a), traversing (b) a single binary tree and the sum of these costs (c). Normalised cost functions for building (d), traversing (e) a multiple binary trees and the sum of these costs (f).

pares it with the costs of the new hierarchical contact search. The latter requires fewer operations for building trees, see Figure 4.22(a) and Figure 4.22(d), however the quadratic dependence of the number of bodies governs the traversing operation. The cost of traversing a single binary tree (Figure 4.22(b)) does not show a steep gradient when the number of bodies increases, while traversing multiple ADTs for 4 or more mutually contacting bodies results significantly more expensive (Figure 4.22(e)). Overall, the total costs are summed and shown in Figure 4.22(c) for a single ADT, and in Figure 4.22(f) for the new contact search approach.

From the difference between the total costs associated to a single tree and multiple trees, it is possible to quantify the gain of a method with respect to the other. This is illustrated in Figure 4.23 along with a continuous line that marks the no-gain level-set, i.e. when the two methods have the same theoretical cost. The region to the left of this line is where the new contact search results cheaper, whereas the region to the right indicates that a standard ADT would perform better.

Figure 4.23 shows that when the number of bodies m mutually in contact is smaller than 4, it is advantageous to treat each body-contact-pairs individually. If $m = 4$ there is no theoretical advantage of a method over the other, but for larger values the cheapest approach is to employ a single ADT only. Furthermore, the results show that the gain is nearly independent on the number of outer facets of a body when this is sufficiently large (approximatively 200 or above). This is shown by the isosurfaces which tend to become vertical lines as the number of facets per body increases.

The results presented in this section show that the theoretical cost of the new narrow search approach is sustainable and comparable to the one of existing methodologies. This is concluded from an analysis that looked at dense configurations of polydisperse

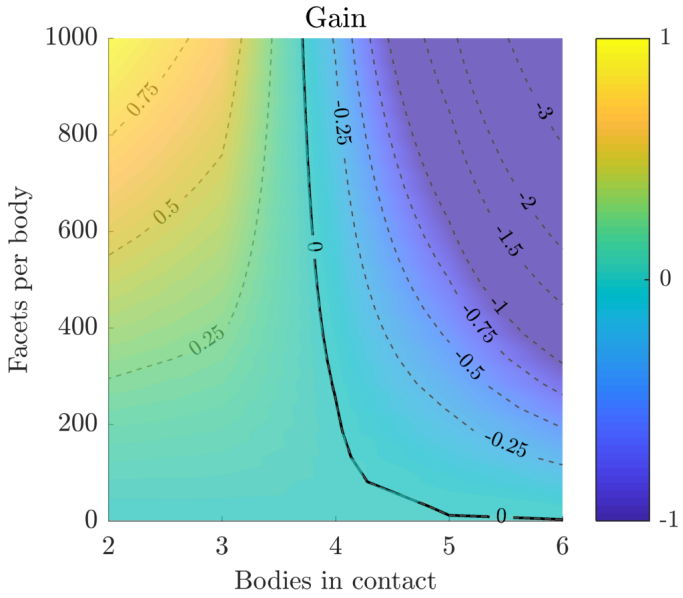


Figure 4.23: Gain of common narrow search methodologies over the new hierarchical contact search presented in Chapter 3. The tick continuous line shows the no-gain level-set, that is, when the gain is null.

nearly-convex bodies. Consequently, for loose assemblies it is expected a reduction in computing cost. Only extremely polydisperse and dense assemblies can cause the theoretical cost to be higher; however, as it shall be presented in Chapter 5, this scenario is not encountered in reality.

4.7 Particular case of concave NURBS bodies

As the literature review in Chapter 2 highlighted, there is a need for contact detection algorithms that can handle NURBS bodies efficiently.

There are essentially two challenges to address while solving distance queries with NURBS in computational mechanics:

1. The cost associated to the evaluation of the basis functions; and
2. The solution to point projection problems depends upon an initial guess difficult to define.

The former makes the construction of usual spatial data structures search more expensive, the latter affects predominantly the narrow phase of contact search, in which an inaccurate initial guess will lead to a slow converge rate or, even worse, to an incorrect solution (local minima).

The methods presented in Chapter 3 have been designed to address these challenges and this section illustrates how to solve, within the novel hierarchical framework, distance queries between a point and concave NURBS. For the sake of completeness, basic notions of B-splines basis and NURBS functions are briefly reviewed below. Sections 4.7.2 and 4.7.3 focus on the two challenges described above, respectively.

4.7.1 B-spline functions and NURBS surfaces at a glance

NURBS have properties that are rooted in the B-spline basis functions. The properties that will be exploited for solving distance queries are briefly reviewed.

The most common formulation of B-spline basis function is due to Cox and de Boor (54): the i-th B-spline of order p is a function $N_{i,p}(u)$ defined over the interval $u \in [u_0, u_m]$ by means of $m = n + p + 1$ *knots*. All knots are listed in non-decreasing and non-uniform[1] knot vector $\Xi = \{u_0, ..., u_m\}$.

It is possible to evaluate efficiently B-splines basis functions with finite-precision arithmetic by means of the Cox-deBoor recursive formulation:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{4.7a}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - ui} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - ui + 1} N_{i+1,p-1}(u) \tag{4.7b}$$

[1]Non-decreasing means that $u_0 \leq .. \leq u_i \leq .. \leq u_m$. Non-uniform means not equally distributed between the domain extrema u_0, u_m.

The evaluation of $N_{i,p}(u)$ for an arbitrary order p from the step function in Eq. (4.7a) and it continues with the linear combination in Eq. (4.7b) for functions of order $p > 0$.

B-splines basis functions have important properties that are exploited by the algorithmic framework developed in this thesis. These are briefly reviewed here:

P. 4.1. *Local support*: $N_{i,p}(u) = 0$ if $u \notin [u_i, u_{i+p+1}]$

P. 4.2. *Non-negative*: $N_{i,p}(u) \geq 0$ for all i, p and u

P. 4.3. *Partition of unity*: $\sum_{j=i-p}^{i} N_{j,p}(u) = 1$ for all $u \in [u_i, u_{i+1}[$

P. 4.4. *Continuity*: $N_{i,p}(u_i)$ has $p - k$ derivatives in u_i, where k if the multiplicity of u_i in the knot vector.

B-splines are used to define NURBS bodies, such as the curve $C(u)$ in $\mathbb{R}^n$:

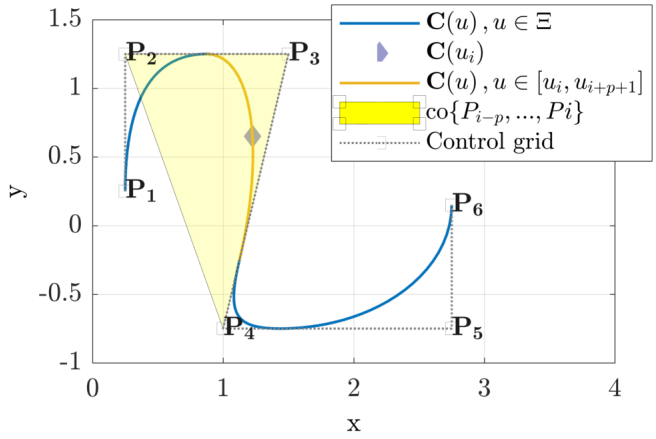
$$C(u) = \frac{\sum_{i=0}^{m} N_{i,p}(u)\, P_i\, w_i}{\sum_{i=0}^{m} N_{i,p}(u)\, w_i} = \sum_{i=0}^{n} R_{i,p}(u)\, P_i \tag{4.8}$$

where P_i is a list of control points, each one associated to a weight w_i, and $N_{i,p}$ are the B-spline basis functions defined in Eq. (4.7). In what follows, the extrema of the knot vector Ξ have multiplicity $p+1$ and are normalised between 0 and 1 for practical reasons. Furthermore, the rational function $R_{i,p}(u)$ is defined over the knots of Ξ which may be non-uniformly distributed; hence the name "NURBS" (non-uniform rational B-splines).

From the contact search point of view, the two most important properties that NURBS inherit from B-splines are:

5. *Interpolatory at the extrema*: $C(0) = P_0$ and $C(1) = P_m$;
6. *Convex hull*: For $u \in [u_i, u_{i+1}]$, it follows from P. 4.1-P. 4.3, that $C(u)$ lies within $\text{conv}\{P_{i-p}, ..., P_i\}$.

For further details on B-splines and NURBS, refer to (191, 205).



Point	x	y	w
P_1	0.25	0.25	1
P_2	0.20	1.25	1
P_3	1.50	1.25	1
P_4	1.00	−0.75	3
P_5	2.75	−0.75	1
P_6	2.75	0.15	1

Figure 4.24: Illustration of convex hull property for a quadratic NURBS curve defined on $\Xi = [0, 0, 0, 0.25, 0.5, 0.75, 1.0, 1.0, 1.0]$.

4.7.2 Bounding volumes

Aided by the improvements made to the GJK algorithm, the novel methods presented in Chapter 3 make use, to the best of the author's knowledge, for the first time of the convex hull property 6 to solve distance queries between NURBS.

It follows from 6 that control grids offer a natural bounding volume to hierarchical contact searches. Furthermore, since the property holds true of an arbitrary interval $[u_i, u_{i+1}] \subset \Xi$, NURBS may be decomposed in approximatively convex shapes at no extra cost. Formally: the set of points $\{C(u^\star) : u^\star \in [u_i, u_{i+1}\}$ is contained in conv $\{P_{i-p}, ..., P_i\}$. This property is illustrated in Figure 4.24, in which a knot u_j is taken in the interval $[u_i, u_{i+1}]$ and the convex hull conv $\{P_i, ..., P_{i+p+1}\}$ containing $C(u_j)$ is highlighted.

With no extra coding, the GJK algorithm can be used to measure the distance between a point Q and the convex hull conv $\{P_i, ..., P_{i+p+1}\}$, or a portion of it. If $d(\text{conv}\,\{P_i, ..., P_{i+p+1}\}, Q) > 0$, as described in Chapter 3, the hierarchical contact search continues without building bisection trees. If $d(\text{conv}\,\{P_i, ..., P_{i+p+1}\}, Q) = 0$ the contact search triggers the narrow phase and the GJK returns a set of points, not

necessarily unique, that supports Q in $\text{conv}\{P_i, ..., P_{i+p+1}\}$. These are of course the support points in Eq. (2.16). For the example in Figure 4.24 a set of support points returned is $\{P_2, P_3, P_4\}$.

The following section illustrates how the information about the support of Q is used to initiate the narrow contact search.

4.7.3 Initial guess for projection problems

A projection problem consists of finding the point on a NURBS which is closest to a query point than any other NURBS point. Projection problems are of utmost importance for many applications that involve NURBS; for example, in engineering these are the starting point for solving distance between bodies or computing contact forces. The solution of such problem is most efficiently computed with iterative procedures, (113) whose convergence however depend upon an initial guess.

On the one hand, a good guess is expensive to compute but, on the other hand, a poor guess may converge to local minima (191). Good guesses may be found with a brute-force approach, which consists of evaluating a NURBS at arbitrary values of the knot span to obtain sample points: the sample which realises the shortest distance is then used as the initial guess. This approach, however, is computationally too expensive. Moreover, it does not guarantee that one of the sampling points provides an initial guess sufficiently close to the solution. On the other side of the spectrum are subdivision methods such as (112, 145). These aim to reduce the computing cost of initial guesses (192), but an ultimate solution has not been found yet.

The novel hierarchical contact search can provide, with no extra coding, initial guesses for solving projection problems. This section presents the idea to estimate

the initial guess on NURBS, and the methodology is illustrated for a representative example of non-convex NURBS.

Let us consider a body $\Omega \subset \mathbb{R}^n$ described by a NURBS surface $S(u, v)$. This is a bivariate function, whose image is z_p, defined over the knot vectors Ξ_u and Ξ_v. $S(u, v)$ is obtained from the tensor product construct of two univariate NURBS (191):

$$S(u,v) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} R_{i,p_u}(u)\, R_{j,p_v}(v)\, P_{i,j} \tag{4.9}$$

where $\{P_{i,j}\}$ form the grid of control points in $\mathbb{R}^n$, and the rational functions $R_{i,p_u}(u)$, $R_{j,p_v}(v)$ are defined in Eq. (4.8).

To project a query point Q on the surface $S(u, v)$, one has to compute a pair of unknown knots $u_i \in \Xi_u$ and $v_i \in \Xi_v$, that realises the shortest distance $d(Q, S(u_i, v_i))$.

The computation of u_i and v_i requires an iterative algorithm to find the roots of $F(u, v)$, where:

$$F(u,v) = \begin{bmatrix} f(u,v) \\ g(u,v) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{S}(u,v)}{\partial u} \cdot (P - S(u,v)) \\ \frac{\partial \mathbf{S}(u,v)}{\partial v} \cdot (P - S(u,v)) \end{bmatrix}. \tag{4.10}$$

Eq. (4.10) is derived from the definition of distance and orthogonal condition in Eq. (2.7), the latter introduces the dot product and ensures that the distance is minimum when $F(u_i, v_i) = 0$.

Algorithm 7 shows a routine for solving Eq. (4.10) with Newton's iteration. The required inputs are Q, the data structure defining $S(u, v)$ and an initial guess of the solution (u^0 and v^0). The user should also specify a tolerance ϵ_1 on the distance $d(P, S(u, v))$ and another one ϵ_2 on the $F(u, v)$. The process terminates if any of these values is below the respective tolerance, or if the updated solutions u^{i+1}, v^{i+1} remain nearly constant between consequent iterations.

The convergence of Algorithm 7 is subject to the initial guess of u^0 and v^0, and the

Algorithm 7 Projection on NURBS surface

Inputs: query point P, surface $\mathbf{S}(u, v)$ and initial values for u^0 and v^0.
ϵ_1, ϵ_2, $i = 0$
Continue = 1
while Continue **do**
    Evaluate surface and its derivatives at u_i, v_i:
    $S_i = S(u_i, v_i)$
    $S_u = \frac{\partial S(u_i,v_i)}{\partial u}$
    $S_v = \frac{\partial S(u_i,v_i)}{\partial v}$
    $S_{uu} = \frac{\partial^2 S(u_i,v_i)}{\partial u^2}$
    $S_{vv} = \frac{\partial^2 S(u_i,v_i)}{\partial v^2}$
    $S_{uv} = \frac{\partial^2 S(u_i,v_i)}{\partial u \partial v}$
    if $|\,(P - S_i)\,| \leq \epsilon_1$ **then**
        Continue = 0 ▷ Points coincide
    if $\frac{|(P-S_i)\cdot S_u|}{|(P-S_i)|\,|S_u|} \leq \epsilon_2$ and $\frac{|(P-S_i)\cdot S_v|}{|(P-S_i)|\,|S_v|} \leq \epsilon_2$ **then**
        Continue = 0 ▷ Orthogonality satisfied
    Assemble:
    $$F = \begin{bmatrix} S_u \cdot (P - S_i) \\ S_v \cdot (P - S_i) \end{bmatrix}$$
    and
    $$J = \begin{bmatrix} S_u \cdot S_u + (P - S_i) \cdot S_{uu} & S_u \cdot S_v + (P - S_i) \cdot S_{uv} \\ S_u \cdot S_v + (P - S_i) \cdot S_{uv} & S_v \cdot S_v + (P - S_i) \cdot S_{vv} \end{bmatrix}$$
    Compute new solutions u_{i+1} and v_{i+1} from:
    $$J \begin{bmatrix} u_{i+1} - u_i \\ v_{i+1} - v_i \end{bmatrix} = -F$$
    Set: $i = i + 1$ ▷ Increment iteration counter
    if $u_i \leq 0$ **then**
        $u_i = 0$
    if $u_i \geq 1$ **then**
        $u_i = 1$
    if $v_i \leq 0$ **then**
        $v_i = 0$
    if $v_i \geq 1$ **then**
        $v_i = 1$
    if $|(u_i - u_{i-1})S_u + (v_i - v_{i-1})S_v| \leq \epsilon_1$ **then**
        Continue = 0 ▷ Parameters do not change

new hierarchical contact search presented in Chapter 3 is designed to provide a good initial guess to it. The idea is to use the knowledge of the witness points computed by the GJK algorithms. This information are acquired during the broad contact search and are therefore available in the narrow phase at no extra cost.

Let us consider an arbitrary NURBS curve $\mathbf{C}(u)$ with $u \in \Xi = [u_0, ..., u_m]$ and control points $\{P_i\}$. When invoked to compute the distance $d(\text{conv}\,\{P_i\}, Q)$, the GJK algorithm returns the witness point $z_P \in \text{conv}\,\{P_i\}$ expressed as a convex combination of a small set $\{s_i\} \subset \{P_i\}$ and barycentric coordinates λ_i. Namely, $z_P = \sum_{i \in I} \lambda_i s_i$, where I is the subset of points of $\{P_i\}$, not necessarily unique, supporting z_P.

By observing that when $d(\text{conv}\,\{P_i\}, Q) = 0$ the points z_P and Q coincide, it is reasonable to ask whether it exists a relationship between the support of z_P and Q, which is the image of a good initial guess u^0. In other words, the support of z_P is expressed by λ_i and a small set I. Since $z_p \equiv Q$, these may contain sufficient information to find a good guess u_0 whose image in Q. The rest of this section discusses on a possible formulation of such empirical relationship.

The idea is to use λ_i as "weights" that the i-th support point in I exerts on the estimated image of a good guess u_0. To do so, firstly, $2n$ sampling knots defined. These are found between the extrema u_0 and u_m of Ξ at $2n-1$ intervals. The value defining the middle knot u^m is defined by the following equation:

$$u^m = \sum_{i \in I} \lambda_i u_i \tag{4.11}$$

where $\{u_i\}$ are the knots in Ξ at the positions defined by I. The first n sampling knots are found at equal intervals between u_0 and u^m, and n more sampling knots are found at equal intervals between u^m and u_m. The initial guess u^0 if found among

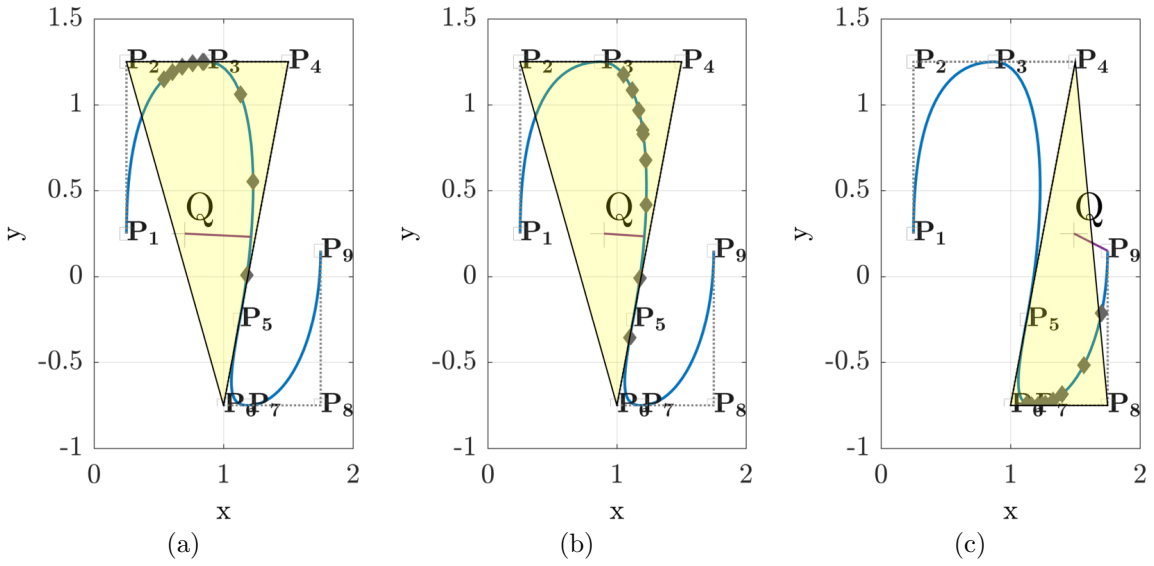


Figure 4.25: Solutions for point projection problems between a query point Q and a NURBS curve. The initial guess for the iterative solution is provided by the support points of Q.

these $2n-1$ samples by comparing the distance of their image on $C(u)$ and Q. The one realising the shortest distance is used to define u_0. Obviously a large number for n in Eq. (4.11) makes the procedure more expensive, but in practise the value $n = 5$ works well. This is a small value that makes the search for u_0 computationally sustainable.

The effectiveness of Eq. (4.11) is presented by the example in Figure 4.25. The figures correspond to three time-steps of the query point q travelling from left to right. The solution of the point projection problem is initialized by the grey points on the NURBS curve that is closer to q than any other. The minimum distance vector is shown in purple.

A test in which q takes 500 steps to move from one part of the domain to the other is performed. The tolerances are set to $\epsilon_1 = \epsilon_2 = 10e-5$, and the result are compared against a brute force approach that distributes 200 sampling knots homogeneously on the NURBS curve. When successful, the root-finding algorithm converges in less than 8 iterations, 5 on average, to the correct solution. However, for the coarse control grid

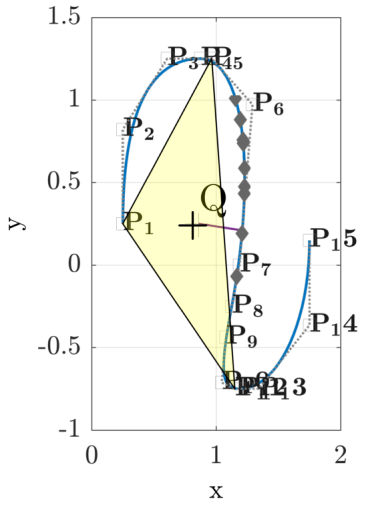


Figure 4.26: Solutions for point projection problems between a query point Q and a h-refined NURBS curve.

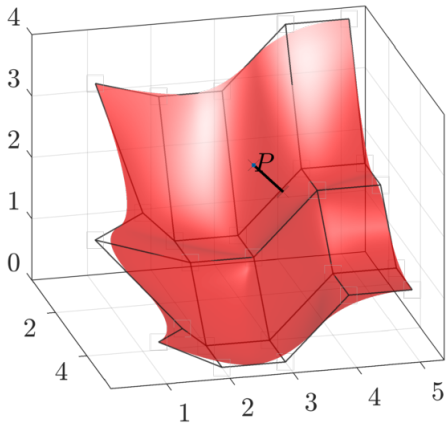


Figure 4.27: Projection of a query point P on NURBS surface.

illustrated in Figure 4.25 the algorithm is successful only 80% of the times due to the inaccurate guess of u^0. The rate of success increases to 92% when the NURBS curve is h-refined, for example in view of an isogeometric analysis, as shown in Figure 4.26.

Similar results are obtained for the NURBS surface depicted in Figure 4.27. Algorithm 7 converges on average in 12 iterations, but the rate of success is strongly dependent on the position of the query point Q.

Overall, the examples presented in this section have shown that the novel algorithmic framework is applicable to non-convex NURBS curves and surfaces. Furthermore:

- A new method for defining a good initial guess for projection problem is presented;
- NURBS bodies are treated intrinsically;
- The robustness increases for h-refined NURBS curves and surfaces.

4.8 Comparison with published works

This section presents numerical experiments that compare the new methods with other published in the literature. Particular attention is paid to speed, accuracy and robustness. The speed of an algorithm is measured by the time that a CPU spends in executing a process, the time that elapses for the execution is the *CPU time*. For iterative algorithms, such as the GJK distance algorithm, another important measure of speed is the *rate of convergence*.

4.8.1 CPU time for primitive testing

The first experiments compare the CPU time required by three different recursive algorithms: Johnson algorithm (106), Voronoi region search (64), and Signed Volumes method (Chapter 3). These are tested to solve a distance query between a point and all simplices in 3D. Degenerate simplices are not considered here.

For the sake of simplicity, all distance queries are formulated between a simplex τ and the origin O. Therefore, these experiments compute the point of minimum norm $\nu(\tau)$. A test is carried out for each subregion of τ. This is done by varying the orientation of τ with respect to O so that $\nu(\tau)$ can be found in different subregions. For example, all 9 subregions of a 2-simplex are tested by applying linear transformations to the vertices of τ.

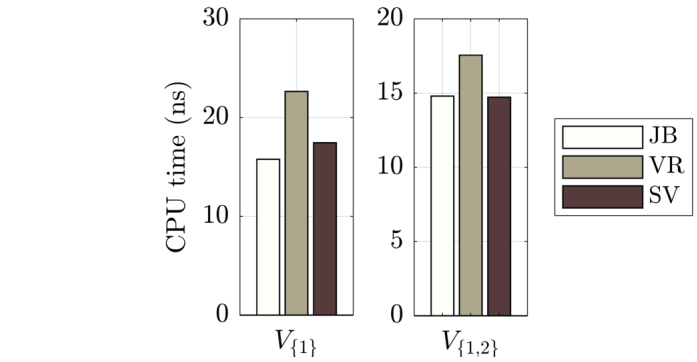
The results indicate that all algorithms perform differently when testing different subregions of a simplex. Johnson algorithm performs particularly well for all simplices and its cost increases with the cardinality of the subset that contains O. The Voronoi region search does not show such a uniform trend and it generally requires more CPU time than other procedures. The Signed Volumes method performs exceptionally well in some cases, but when the point of minimum norm is supported by a single vertex it results expensive. The CPU time of all algorithms used to test 1-simplex, 2-simplex and 3-simplex is shown in Figure 4.28. These results are averages of a million repetitions and refer only to certain subregions of the simplices (see details on the labels of abscissa).

Apart from 1-simplices, in all other cases the performance of these algorithms is remarkably different. The difference in CPU time is due to the peculiar logic that each algorithm follows. In fact, the solution to the distance query consists in finding the subset containing O, and these results show that some logics are more efficient than others. Figure 4.28 clearly indicates that the Johnson algorithm demands more CPU time as the cardinality of the subset that contains O increases. The Signed Volumes methods however shows an opposite trend: when the O is inside the simplex, the cost is minimum.

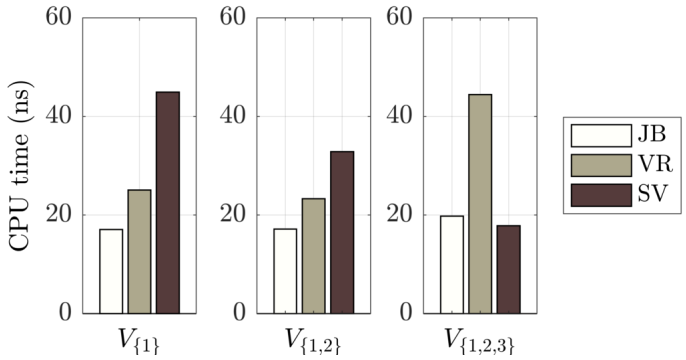
4.8.2 Performance of the GJK algorithm

This section presents a parametric study aiming to assess the impact that different distance sub-algorithms have on speed and accuracy of the GJK algorithm. Four sub-algorithms are compared:

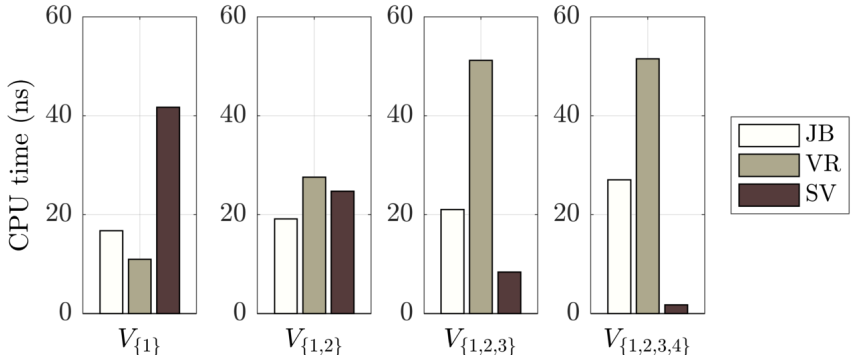
- Johnson algorithm and Backup procedure (JB)
- Johnson algorithm only (JH)



(a) Test on 1-simplex



(b) Test on 2-simplex



(c) Test on 3-simplex

Figure 4.28: Comparison of CPU time for testing primitives using Johnson algorithm (JB), Voronoi region search (VR), and Signed Volumes method (SV).

- Backup procedure only (BK)
- Signed Volumes (SV)

Recall from Chapter 2 that, since the GJK algorithm was first conceived, Johnson's procedure is prune to instability issues. However, in real-time simulations the Backup procedure results too expensive in terms of CPU time (225). Usually, for these applications, JH is preferred to JB. Furthermore, in practise BK is never used on its own, but it will be here considered as reference solution.

The parameters varied in this study are configuration between objects and target accuracy ε_{tol}. The three configurations studied are: distant, touching and overlapping. Two different implementations of GJK algorithms, with and without hill-climbing (38), are used to compute the minimum distance between two polytopes in 3D space.

Firstly, the CPU time dedicated to the distance sub-algorithm is measured for each GJK call. This is the sum of the CPU time that the sub-algorithm requires at each GJK iteration. Secondly, the CPU time required by a distance query is measured as a function of ε_{tol} and the number of vertices in the polygonal spheres. Each test is repeated one million times and the results is the average. The results are described below.

CPU time for distance sub-algorithms. The overall CPU time required by the sub-algorithms for computing the minimum distance between two polygonal spheres is reported in the following tests.

Figure 4.29 compares the total CPU time of JB, BK and SV sub-algorithms for colliding, touching and distant spheres with $\varepsilon_{tol} = 1e{-}8$. Results using hill-climbing are in Figure 4.29(a). All configurations show similar trends: SV exhibits minimum

CPU time, BK is the most computationally intense, and JB is in between the two. Similar results are obtained when disabling hill-climbing, see Figure 4.29(b). There is little difference in CPU time of when enabling or disabling hill-climbing. This is not surprising since incremental techniques only affect the cost of evaluation of the support function. Altogether, for all these experiments SV performs always faster or, occasionally, as fast as JB.

For the next tests, the target accuracy is set to $\varepsilon_{tol} = 10^{-14}$. Results with and without hill-climbing are shown in Figure 4.30(a) and Figure 4.30(b), respectively. These show patterns similar to Figure 4.29, however in some cases hill-climbing on the same mesh is more expensive.

More importantly, there are cases in which BK is faster than JB. This result is counterintuitive and requires particular attention. The lack of accuracy of JB implies that the GJK algorithm does not descend toward the optimal path, instead it takes more iterations to converge and thus more CPU time. The SV sub-algorithm instead does not exhibit such pathology and its computational cost seems independent from the number of vertices defining the polytopes.

The results in Figures 4.29 and 4.30 demonstrate that there are cases in which the accuracy of the sub-distance algorithm plays a key role on the computational cost of the overall distance sub-algorithm. The speed-up achieved by SV over JB is between 10% and 25% for distant spheres and between 15% and 30% for overlapping bodies. Only for touching bodies, their performance is comparable.

Effect of $\varepsilon_{\mathbf{tol}}$ on CPU time of GJK algorithm. Further experiments are carried out to study the effect of the tolerance $\varepsilon_{\mathrm{tol}}$ on the CPU time of the GJK

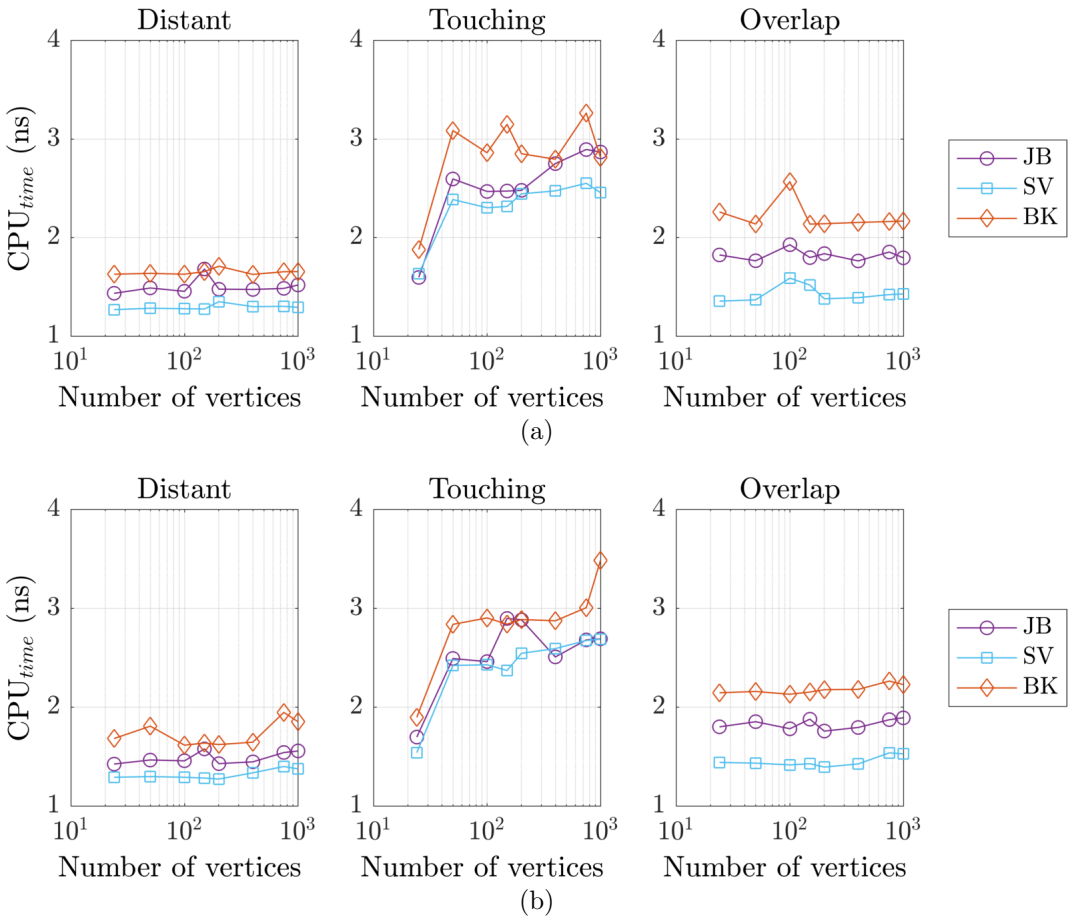


Figure 4.29: Comparison of sub-algorithms CPU time (ns) for $\varepsilon_{tol} = 1 \times 10^{-8}$ with (a) and without (b) hill-climbing. Three configurations are considered: distant, close and overlapping spheres.

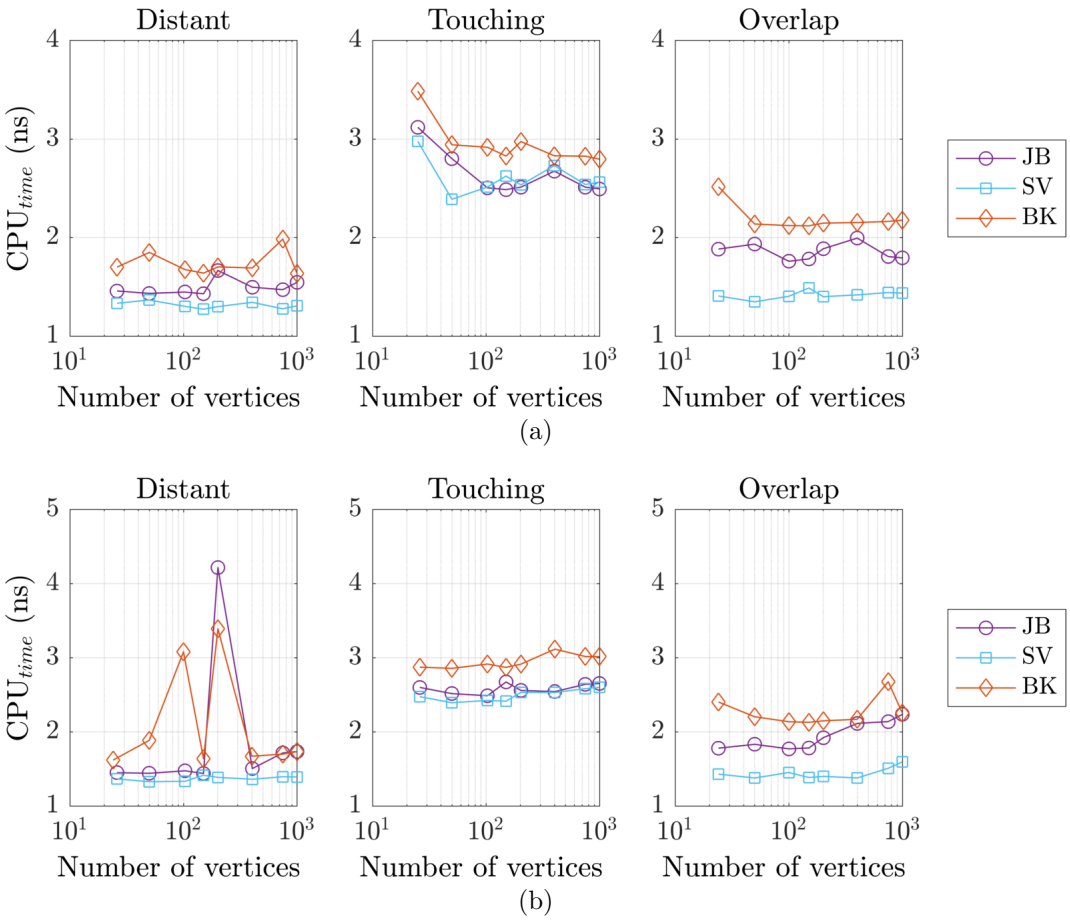


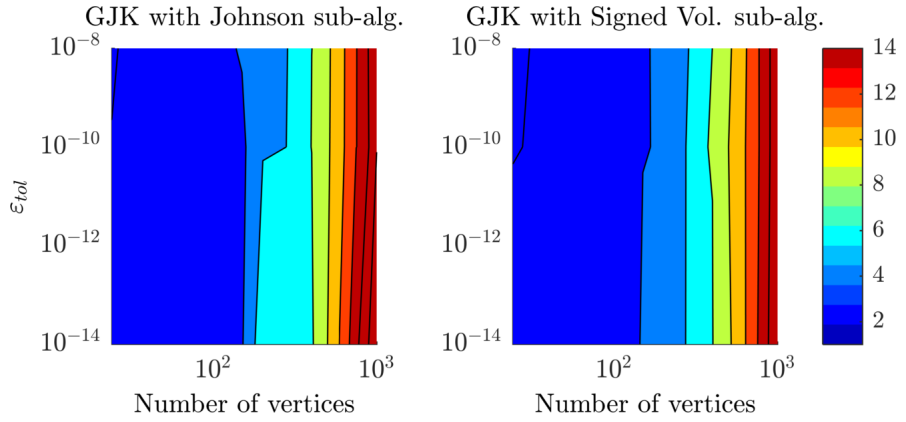
Figure 4.30: Comparison of sub-algorithms CPU time (ns) for $\varepsilon_{tol} = 1 \times 10^{-14}$ with (a) and without (b) hill-climbing. Three configurations are considered: distant, close and overlapping spheres.

algorithm. Distance queries similar to those described earlier, between two polygonal spheres, are repeated for different values in the range $10^{-8} \leq \varepsilon_{\text{tol}} \leq 10^{-14}$.

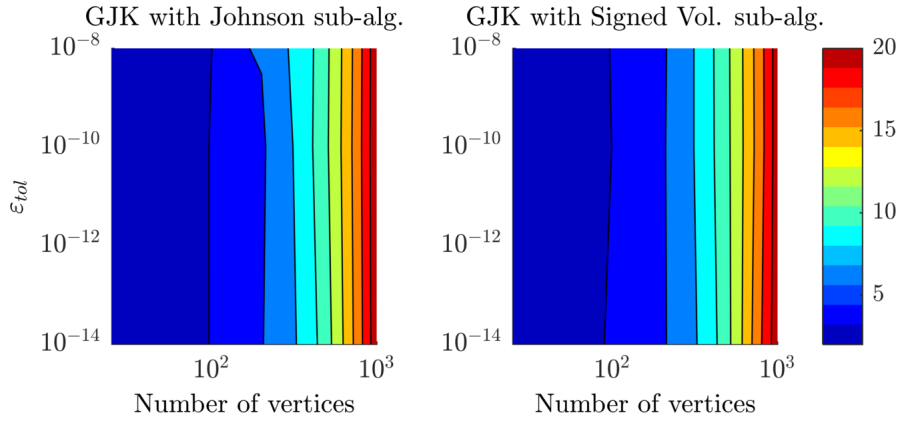
The first tests invoke the GJK algorithm without hill-climbing. Figure 4.31 shows the measurements of the GJK CPU time as a function of accuracy ε_{tol} and number of vertices in one sphere. The graphs on the left use JB sub-algorithm, whilst the right ones use SV. For all graphs in Figure 4.31, the CPU time increases with the number or vertices and shows little dependence from ε_{tol}. This common pattern is not surprising and is due to the evaluation of the support function, see Eq. (2.9), which increases linearly with the number of vertices.

The sub-algorithm makes larger impact on the CPU time when hill-climbing is active. Figure 4.32 shows the GJK CPU time as function of accuracy ε_{tol} for distant, close and overlapping spheres. A comparison between the graphs on the left (JB) and those on the right (SV), highlights that a reduction of CPU time is achieved by SV sub-algorithm. The difference in CPU time is more pronounced when the objects are found in contact (Figure 4.32(c)), in which case SV improves the performance from a minimum of 5% to a maximum of 25%.

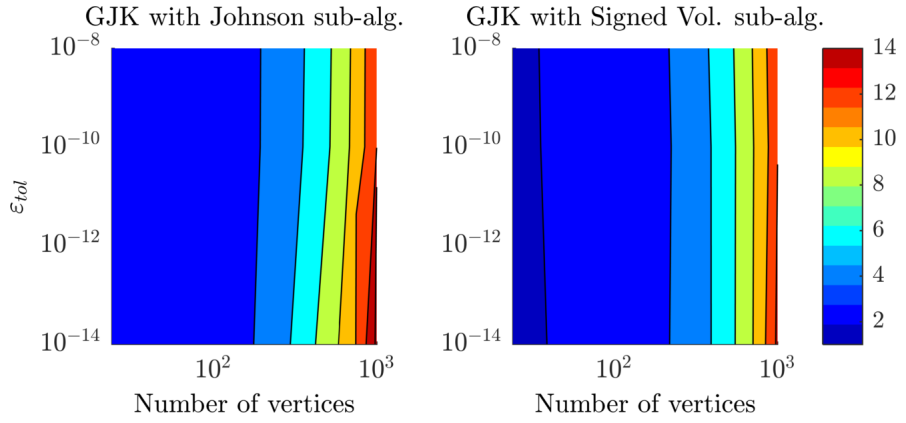
Let us now discuss the reasons behind the different performances of JB and SV. For overlapping objects SV is faster because it finds the solution at the very first inspected Voronoi region than JH. For distant configurations it is argued that SV is faster, or as fast as JH, because of geometrical and numerical reasons. Firstly, the configuration space obstacle CSO is given by the Minkowski difference $P - Q$ (Eq.(2.8)) and its morphology is dictated by the mutual orientation of P and Q. It exists a relationship between the face of the CSO supporting the point closest to the origin and the faces supporting the witness points $z_P \in P$ and $z_Q \in Q$. Table 4.2 considers all possible



(a)

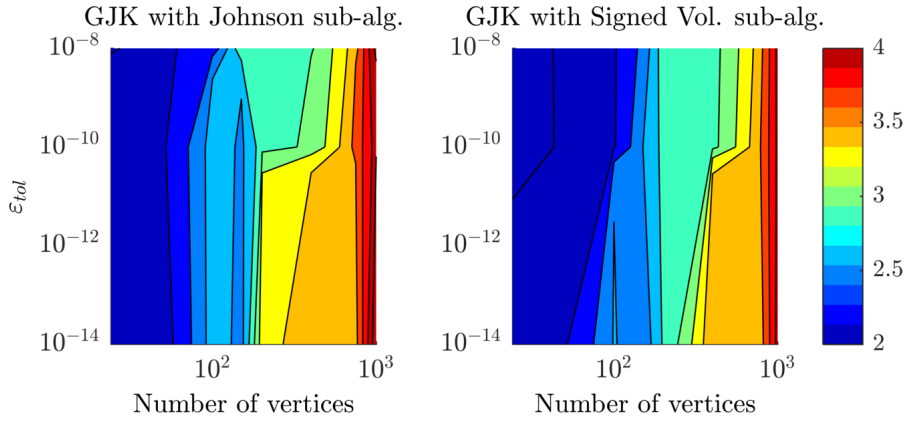


(b)

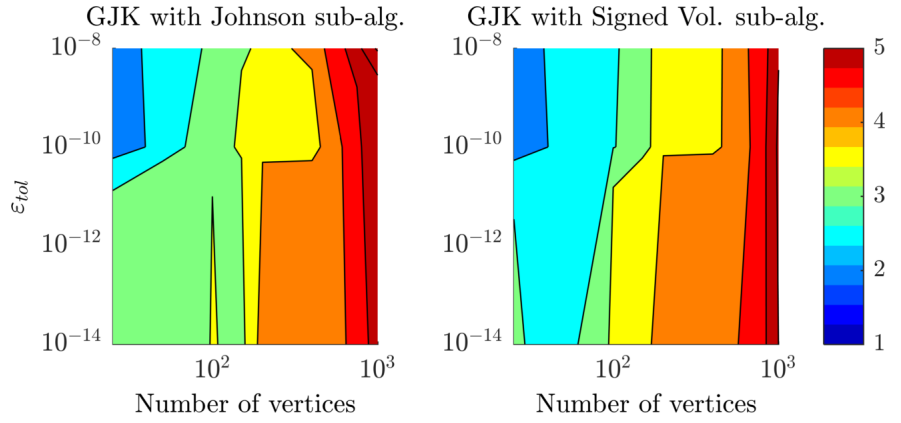


(c)

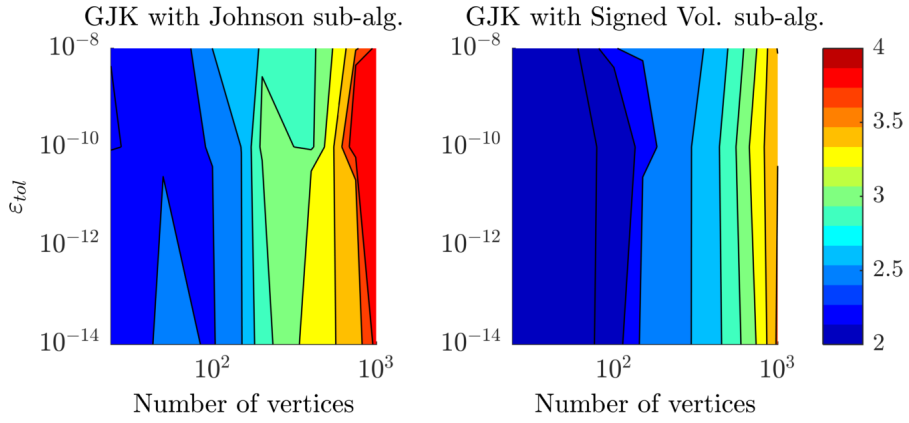
Figure 4.31: Measurements of GJK CPU time for distant (a), close (b) and overlapping (c) polygonal spheres without hill-climbing. The colourmaps represent the time in ns.



(a)



(b)



(c)

Figure 4.32: Measurements of GJK CPU time for distant (a), close (b) and overlapping (c) polygonal spheres using hill-climbing. The colourmaps represent the time in ns.

Table 4.2: All combinations of faces supporting the witness points z_P and z_Q, and resulting supporting faces of $\nu(\text{CSO}) = \nu(\tau)$.

$z_P \in$	$z_Q \in$	$\nu(\tau) \in$
Vertex	Vertex	Vertex
Vertex	Edge	Edge
Vertex	Polygon	Triangle
Edge	Edge	Triangle
Edge	Polygon	Triangle
Polygon	Polygon	Triangle

combinations of faces (vertices, edges and triangles) supporting a pair of witness points in physical space and uses the Minkowski difference to recast this face in configuration space. As a result, the chances that the $\nu(\text{CSO})$ lays on a triangle of the CSO are four times higher than for any other face. The likelihood of finding $\nu(\text{CSO})$ on a triangle of a simplex suggests that the top-down search of the Signed Volumes method will return an answer in fewer operations. Secondly, the accuracy of the distance sub-algorithm makes a bigger impact on meshes with large number of vertices. This is because, on finer meshes, a suboptimal search direction is more likely to compromise the evaluation of the support function. As a result, the support function provides the GJK algorithm with a simplex which is not directed toward the origin in the best possible way.

Overall, because the Signed Volumes method is more accurate than Johnson algorithm, it reduces the CPU time by guiding the GJK algorithm toward an optimal search path. To minimise the GJK CPU time, it is more important to optimise the GJK convergence rate by increasing the accuracy of the sub-algorithm, rather than minimise the number of operations with the risk of compromising the search path. This is particularly important for problems involving large meshes or representations of bodies whose the support function is expensive to evaluate.

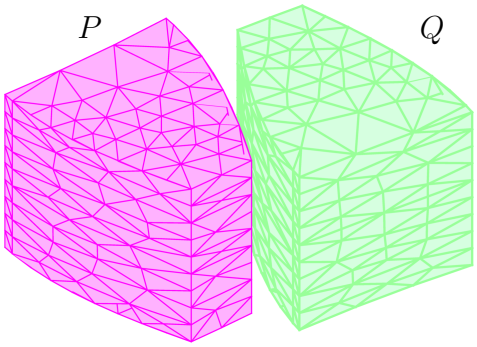


Figure 4.33: Geometries used for the gear teeth benchmark.

4.8.3 Numerical robustness

This section focuses on the algebraic system solved by the distance sub-algorithm at each GJK iteration. The aim is to compare the robustness of the GJK algorithm when employing the Johnson algorithm and the Signed Volumes method. It should be recalled, from Section 3.2, that the former solves $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$, Eq. (2.20), while the latter solves $\mathbf{M}\boldsymbol{\lambda} = \mathbf{p}$, Eq. (3.2). The Backup procedure is here disabled to study more closely the effect of numerical robustness on the GJK algorithm.

Let us consider the 3D meshes of the gear teeth P and Q in Figure 4.33. The GJK algorithm measures, at each solution step, the distance between the teeth as they approach each other. Figure 4.34 shows the values of $|\det \mathbf{A}|$ and $d(P, Q)$ for consecutive simulation steps. The red markers indicate that Johnson's algorithm fails three times.

The failures occur when $|\det \mathbf{A}|$ is in the order of the machine precision, in particular, when degenerate simplices are passed to the distance sub-algorithm. As previously observed in (225), numerical instabilities arise as a consequence of the cancellation error that affects Johnson's algorithm when solving Eq. (2.20).

The test is repeated by substituting Johnson's algorithm with the Signed Volumes method presented in Chapter 3. This time the distance decreases monotonically until

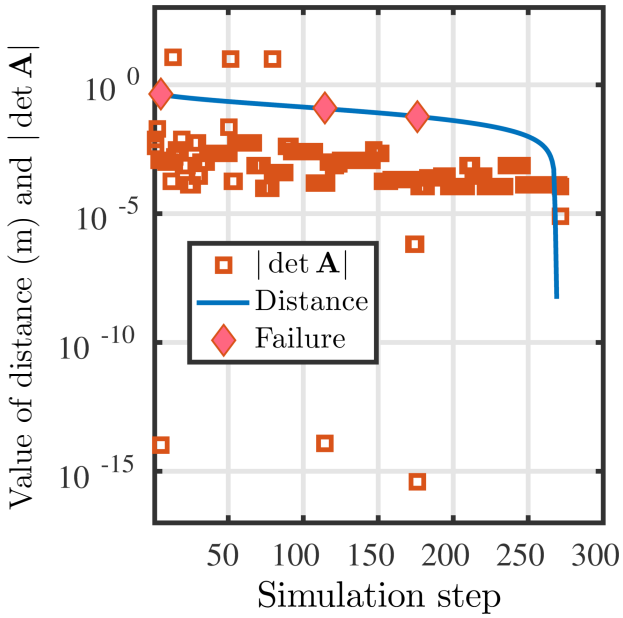


Figure 4.34: Distance measurements for the test in Figure 4.33. $|\det \mathbf{A}|$ is the determinant of the coefficient matrix assembled by Johnson algorithm, when this value approaches the machine precision the GJK algorithm returns erroneous results (red marks).

the gear teeth touch each other, and the smallest value recorded by the GJK algorithm is in the order of the machine precision.

Figure 4.35 compares the results obtained with the two distance sub-algorithms. Unlike Johnson's algorithm, the linear system assembled and solved by the Signed Volumes is always well-conditioned. The results obtained from the GJK algorithm using the JH sub-algorithm are shown in Figure 4.35(a), whilst Figure 4.35(b) shows the results for the SV sub-algorithm. Figure 4.35(b) shows that the absolute value of $\det \mathbf{M}$ remains well above the rounding error for the whole simulation. On both graphs, the coloured bands include the upper and lower limits of the determinant of the matrices. The values of $|\det \mathbf{M}|$ span a range well above the machine precision, instead $|\det \mathbf{A}|$ reaches ϵ several times.

By repeating the same test with different meshes, it is possible to verify that the accuracy achieved by the GJK algorithm is limited when using JH. Figure 4.36 shows

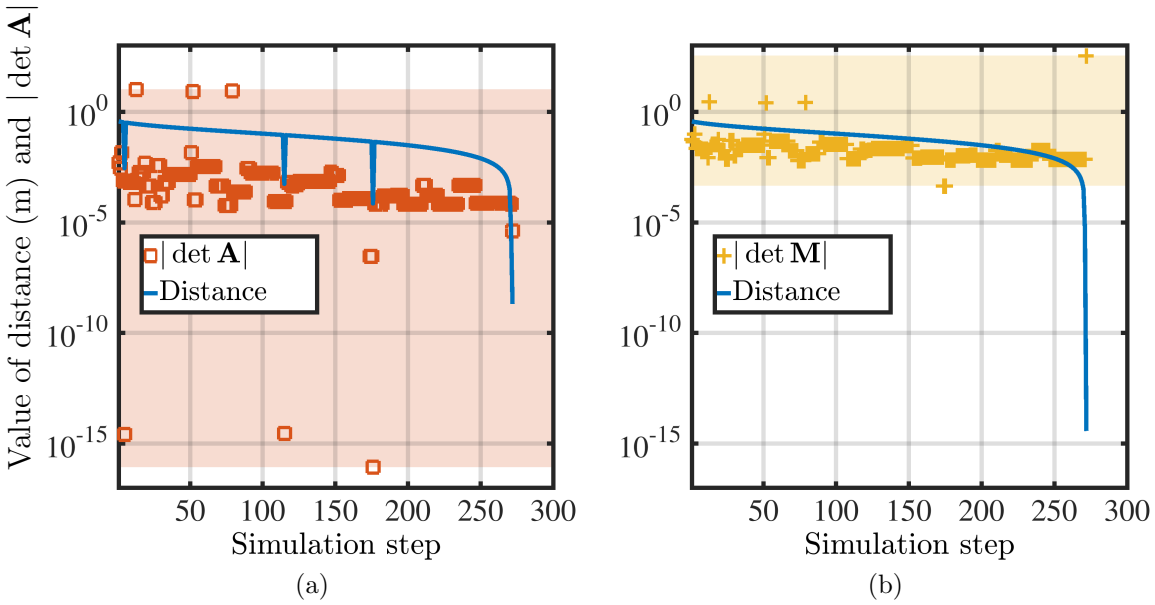


Figure 4.35: Results for the gear teeth benchmark using two different distance sub-algorithms. $\det \mathbf{A}$ and $\det \mathbf{M}$ are the determinants of the coefficient matrices associated to Johnson sub-algorithm (a) and Signed Volumes method (b), respectively.

the different levels of accuracy achieved by the two sub-algorithms. The distance measured at the last solution step using JH is in the order of $\varepsilon^{1/2}$, whereas SV reaches values very close to ε. This is due to the fact that at the last iteration the GJK algorithm generates a flat simplex that is close to the origin.

4.9 Concluding remarks

This chapter has verified the formulation and the computer implementation of the novel algorithmic framework introduced in Chapter 3.

Care was taken to analyse and demonstrate the robustness of the Signed Volumes method. It has been shown that this new method can handle degenerate geometries with double-machine precision as well as with adaptive floating-point arithmetic. In general, the latter increases the computing time, but it is recommended for problems whose final solution depends directly on the outcome of the Signed Volumes method.

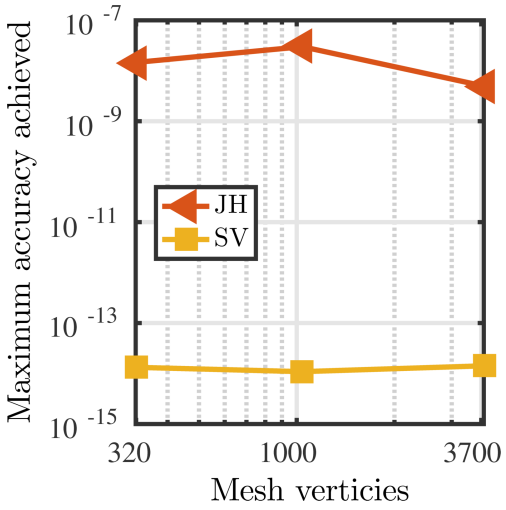


Figure 4.36: Comparison of maximum accuracy achieved by the GJK algorithm using Johnson (JH) and Signed Volumes (SV) procedure for the gear teeth benchmark.

Various scenarios involving rigid and/or deformable elements were investigated. A large suite of tests involving spherical, tetrahedral and hexahedral elements was designed to verify the correct geometrical resolution of contact. All combinations of edge–to–vertex, edge–to–edge, edge–to–facet, etc... were tested for all combinations of elements. In this chapter, only few selected examples were reported in which position, displacement, kinetic energy and contact forces have shown the accuracy of the method. Moreover, the penalty contact formulation was calibrated and assessed against analytical and published work. In particular, the longitudinal impact between collinear rods was modelled to analyse the influence of the penalty factor over the time of contact.

Two tests have validated the logic of the new hierarchical contact search. The first one involved three convex bodies, and the second one consisted of an ironing problem in which a concave body was decomposed in approximatively convex sub-bodies. Afterwards, the worst-case theoretical cost of building and traversing binary trees was studied. From this, it was concluded that if four bodies are mutually in contact, the cost of the new method is comparable to other common methods. However,

for less than four contacting bodies, it could lead to again of 100% or more; whereas, it is expected a similar loss, in the factor of 2, if 5 bodies are tangent to each other.

An important literature gap was addressed by presenting the applicability to NURBS curves and surfaces. In this chapter, the hierarchical contact search was used to handle concave NURBS objects during both broad and narrow phase. For the latter, however, further research is required to apply this algorithm to NURBS-based numerical methods, e.g. isogeometric analysis.

Finally, an extensive assessment compared the performance of the new methods against others published in the literature. Scenarios involving distant, touching and overlapping bodies have shown that the GJK algorithm can now solve distance queries up to 30% faster. CPU time remains the same for touching bodies and for morphologies that yield to degenerate simplices. At the same time, the accuracy was increased to machine precision ε, whilst originally it was $\sqrt{\varepsilon}$.

Chapter 5

Mechanical packing of granular media

The mechanical behaviour of granular materials is governed by the microscopic interaction of particles and their heterogeneous morphologies; these yield to complex contact problems often tackled numerically. Existing methods reduce the problem complexity by coarsely approximating the particle morphologies: a simplistic assumption that is counterbalanced by material models difficult to grasp and calibrate. The computational bottleneck that prevents from using arbitrary particle morphologies, is the contact processing. This is a major limitation even for simplistic approximations, e.g. rigid and spherical particles, and is due to the extrinsic data structures used for contact search. This chapter illustrates how the new algorithms reduces these costs and enables to model granular materials using realistic morphologies acquired with X-ray scans. The case study of a mechanical packing is presented to illustrate the influence of grain discretisation over the final assembly. Furthermore, the hypothesis postulated in Chapter 4, on the sustainability of the new contact search scheme, is here investigated.

5.1 Introduction

Granular materials can either be natural (e.g. rocks, snow, sand) or man-made (e.g. candies, ball bearings, pills) and they are the second-most manipulated material in industry (11, 95). Their ubiquity has impelled a tremendous amount of research aiming to better understand their mechanical behaviour and to provide engineers with tools to predict it.

In recent years, numerical simulations have complemented experimental techniques by offering engineers a scalable solution to study granular materials. These techniques have shed light on the fundamental mechanisms by which particles interact at the microscopic sale (microscale); DEM modelling has enabled the analysis of localised phenomena and force chains (144, 146). More recently, multi-scale frameworks are paving the way for modelling complex materials, such as: porous media, (227, 248), and polydisperse conglomerates (3, 231).

The major limitation to numerical simulations is the computational cost of realistic scenarios which involve billions of particles (120, 135, 260). The computational cost is commonly reduced by introducing various simplifications in the model; however, some of these alternate the overall response and particular care is required.

One of the most debatable assumption concerns the shape of the particles. It is computationally cheaper to model a large number of simple particles, than fewer with arbitrary shapes. However, since the mechanical behaviour of granular materials is dictated by the way particles interact through contact, by modifying their shape, one inevitably compromises the mechanical response.

In order to counterbalance crude approximations on the particle morphology ad hoc

constitutive laws are required. These can either be sophisticated material models or contact models, but both solutions need an experimental calibration. This makes the numerical modelling significantly more expensive and complicated since it introduces a number of (arbitrarily) model parameters. Essentially, simplistic approximations on the particle morphology shift the complexity from an algorithmic perspective to the physical constitutive modelling.

If no assumption is made on the particle shape, constitutive material and contact models may be significantly simplified; however, executing the contact search on particles of arbitrary morphology is the major computational bottleneck (4, 49, 70, 103, 110, 247). As highlighted in the literature review in Chapter 2, existing algorithms make use of extrinsic spatial structures: these are built and updated even for particles unlike to get in contact.

The new algorithms presented in Chapter 3 could circumvent this computational bottleneck. In fact, by conducting the broad search phase using intrinsic spatial structures, these would build and update bisection trees only for particles that are in contact.

In this chapter, the new hierarchical contact search is applied to tackle the algorithmic challenges introduced by non-spherical, concave and deformable particles. The particular case of dense mechanical packing of sand grains with realistic morphology is studied. The aim of this work is therefore to: (i) assess the applicability of the algorithms, (ii) quantify the difference between spherical and polygonal grains, and (iii) investigate the influence of the mesh refinement.

5.2 Motivations

The literature presents several methods for modelling granular materials. These can be classified based on the assumption made about the morphology of particles:

1. Continuum assumption;
2. Idealised particles with statistically representative properties; and,
3. Arbitrary shape.

A large number of constitutive models have been published to model granular media as a continuum, however these are usually applicable only to a specific problem and require experimental calibration. Although they have been successful in civil and geotechnical engineering applications (99, 200, 219), they have difficulties to predict localised phenomena, inhomogeneities and to solve large–displacement problems. Furthermore, the calibration costs increase with the complexity of the model, as this must reflect specific working conditions of the material. For example, if the temperature affects the material behaviour, the calibration requires experimental results obtained at different temperatures — likewise for strain-rate, environment moisture, etc... . These limitations have motivated the development of distinct and coupled continuum-distinct methods (163, 166, 185).

The second approach consists of modelling each particle as distinct computational entity of elementary shape. Introduced by Cundall and Strack (51) to solve geotechnical problems, this idea formulated the discrete element method (DEM). Since then, the DEM has been applied in many engineering problems (146, 163, 180, 232). This method uses an explicit time integration scheme in which the particle interaction is resolved contact by contact and all particles are approximated by elementary shapes (such as

spheres, cones or ellipsoids). The oversimplification on the particle morphology has two consequences: (i) allows to use contact detection and resolution algorithms which require little computing effort per particle, and (ii) varies the contact forces exchanged by particles since it drastically modifies the contact area between them. The latter has been shown to modify the macroscopic behaviour of the material (2, 4, 6, 98, 254) and, similarly to continuum-based methods, requires an experimentally calibrated contact model (146). Without this, the simplistic shapes would invalidated the modelling results (146).

An increasing number of researchers withdraw the spherical-particle assumption to preserve the geometry and to capture the mechanisms which govern the microscopic behaviour. Some of these works have been driven by the advances of visualisation and experimental characterisation tools. For example, 3D X-ray computed tomography was used to study natural (79, 181, 252, 253) and man-made (111) geo-materials. X-ray tomography enabled to acquire the real morphology of grains, described by triangular meshes, and to use this to improve the fidelity of numerical models (4, 222, 230).

Particles described by polygonal meshes allow to capture inertia effects, compute internal stresses and simulate fragmentation, thus enabling the study of explosions, crushing, rock slicing, mechanical erosion and more. Other shape descriptions, reviewed in (144), make use of clustered spheres (83), or NURBS (4, 129, 130, 131). The latter is particularly promising, but in a early stage development and it is still not clear whether, as stated in (4), NURBS reduce the computing time with respect to polyhedral particles.

Polygonal meshes are indeed a versatile solution, but the literature does not present a mesh sensitivity analysis that investigates the relationship between mesh resolution

and final results. It is well-known, that in computational contact mechanics the discretisation plays a crucial role in the solution of continuum problems (247), and a recent study emphasised the need for accurate descriptions of the morphology of granular materials (252). However, in DEM and other distinct methods, the relationship between mesh resolution and final result is often overlooked.

Regardless of whether particles are rigid or deformable, the minimum and maximum lengths of the mesh edges affect the final result of a numerical study. In what follows, the minimum edge length and maximum edge length are described by $H_{\min}$ and $H_{\max}$, respectively.

The values of $H_{\min}$ and $H_{\max}$ influence: the particle morphology, the way these interact, and therefore the macroscopic behaviour of an assembly of particles. For example, let us assume that $1.25\ H_{\min} = H_{\max}$. Given a particle size distribution and three arbitrary values for $H_{\min}$: $0.18\mu\text{m}$, $0.10\mu\text{m}$ and $0.07\mu\text{m}$, the particle volumes vary as show in Figure 5.1. This figure illustrates the probability density function (PDF) of the particle volumes as these are described by a coarse, medium and fine mesh size.

Figure 5.2 shows the influence of $H_{\min}$ on the volume of a grain. Basically, one can interpret the discrepancy between coarse and fine mesh as a numerical artefacts similar to a chance of material properties.

To make use of finely meshed particles, and hence to carry out a mesh sensitivity analysis, one needs to address the curse of contact detection. This motivates the adoption of the new hierarchical framework for modelling granular materials.

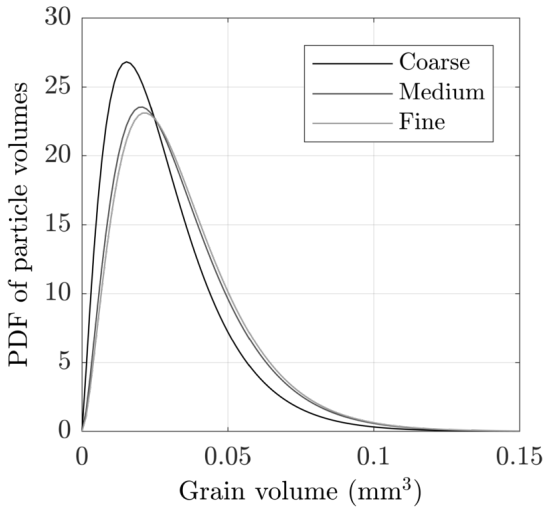


Figure 5.1: Probability density function (PDF) for particle volumes for different mesh refinements.

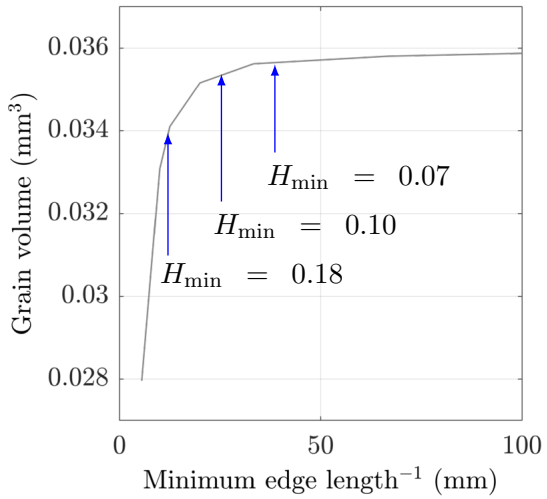


Figure 5.2: Mesh convergence of mean grain volume for different values of minimum edge length H_{min}.

5.3 Method

Three models are defined with the primary goal of comparing performance and outcome of different contact search algorithms. These are implemented in the commercial software LS-DYNA (143) and in a in-house DEM/FEM code. The latter makes use of the new hierarchical contact search described in Chapter 3.

The models simulate the mechanical packing of sand. This consists of applying gravitational load to a sample of particles poured, or sedimented, into a confined domain until they are settled (137, 175, 243, 245). Figure 5.3 illustrates four methods for mechanical packing of particles. In Figures 5.3(a)-(b), the particles are poured, either one after the other or from a hopper, into a confined containment. Alike sedimentation methods, illustrated in Figures 5.3(c), all particles fall under gravitational load; however, in this case the particles fall all together (151).

On the one hand, pouring methods tend to generate a more realistic assembly of particles, but the process in itself is slower than particle sedimentation; therefore, the computing time is significantly higher. On the other hand, sedimentation methods are computationally more efficient, but have to be coupled with geometrical deposition algorithms to improve the fidelity of the final assembly.

This section presents a model to simulate the sedimentation process with a uniform force applied at the top of the specimen, as depicted in Figure 5.3(d).

5.3.1 Metrics

This section introduces the two kinds of metrics used to characterise the assembled particles. The first one concerns the description of the morphology of real sand grains

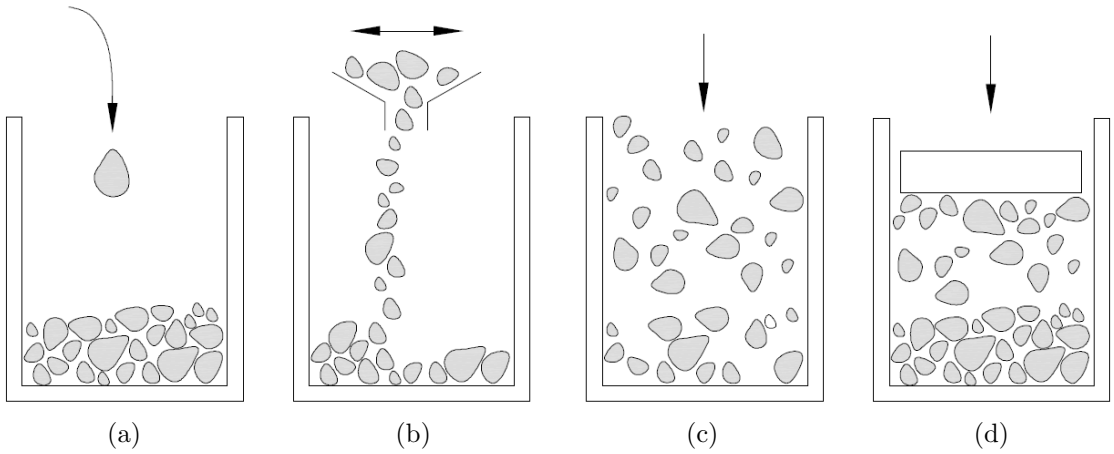


Figure 5.3: Different pouring and sedimentation methods for mechanical packing of particles: (a) one particle at the time, (b) via a hopper, (b) pre-assembled geometrical packing and (d) forced packing by a weight.

and correlates these with the particles in the numerical model. The second one refers to properties of the whole assembly.

The particle properties are mainly concerned with the morphology and define the so-called *granulometry*: a mean to measure and describe the particle size and shape (10, 29). The size of a particle is expressed with respect to an *equivalent sphere*, that is: the sphere with identical area or volume of a grain. The radius of the equivalent sphere is then related to the radius of the *bounding sphere*: the smallest sphere that fully contains the grain. To describe the shape of a particle new parameters are introduced; these include roundness and surface texture are commonly used to distinguish between different types of sand (10, 233, 243, 252) and are often referred as *first order* properties (55).

The properties of the assembly are more deterministic and particularly important to assess the outcome of the numerical simulation. The *void ratio* is one of the most

important parameters and may be defined as follows:

$$e = \frac{V_{\text{box}} - V_{\text{particles}}}{V_{\text{particles}}} \tag{5.1}$$

Where V_{box} is the volume of the containment in which the sand is poured, and $V_{\text{particles}}$ is the sum of all volumes V_i of all N particles: $V_{\text{particles}} = \sum_{i=1}^{N} V_i$.

Another important parameter is the *coordination number*. This quantifies the average number of contacts per particle and is defined by the following equation:

$$Z = \frac{\sum_{i=1}^{N} C_i}{N} \tag{5.2}$$

where C_i is the number of contacts of the i-th particle.

5.3.2 Material models

The container, its lid and the sand particles are all modelled with a simple linear elastic material model. The material properties for the particles, assumed to be silica sand, are reported in Table 5.1 from (56). Refer to Table 5.2 for the material properties of the container.

All objects, including the sand particles, are deformable objects discretised by standard FEM elements. Allowing the particles to deform increases the complexity of the contact search, but is required to capture fragmentation and inertia effects, for example, when modelling impacts and particle crushing. The rigid-body assumption is a simplification that is often made to reduce the computational cost, but also allows to make use of simpler contact search algorithms. For example, under the rigid-body assumption one can readily make use of the incremental GJK algorithm presented in (38). However, this cannot be used for deformable bodies. The rigid-body assumption, is here discharged to show that the new algorithmic framework is applicable in the most

Table 5.1: Material properties of silica sand.

Density	$2.65\ \mathrm{kg\,m^{-3}}$
Young's modulus	$7.6 \times 10^{1}\ \mathrm{N\,mm^{-2}}$
Poisson's ratio	0.3

Table 5.2: Material properties of steel containment box.

Density	$7.85 \times 10^{3}\ \mathrm{kg\,m^{-3}}$
Young's modulus	$2.1 \times 10^{8}\ \mathrm{N\,mm^{-2}}$
Poisson's ratio	0.33

challenging scenario.

A further increase of complexity for the contact search comes from the assumption of frictionless contact. This may lead to unrealistically dense packings, but the assemblies thus generated have a lower void ratio, consequently, the solvers invoke the contact processing functions more frequently. Of course any frictional law may be implemented, but the frictionless assumption makes the contact search more demanding.

5.3.3 Geometry

The containment box and the sand particles are the two main geometrical entities in the model. Each one comprises distinct bodies which, in turn, are decomposed in approximately convex sub-bodies. All the geometries are described by tetrahedral finite element and will be here detailed separately.

Containment A CAD model defines the geometry of the containment box. Outer dimensions are illustrated in Figure 5.4, the wall thickness is 1.5 mm and there is a gap of 0.25 mm between the lid and the walls of the box.

Figure 5.4 is also showing the minimum bounding spheres of few grains. These are not used for contact search purposes, but only in the geometrical assembly: a pre-processing

step detailed in Section 5.3.4.

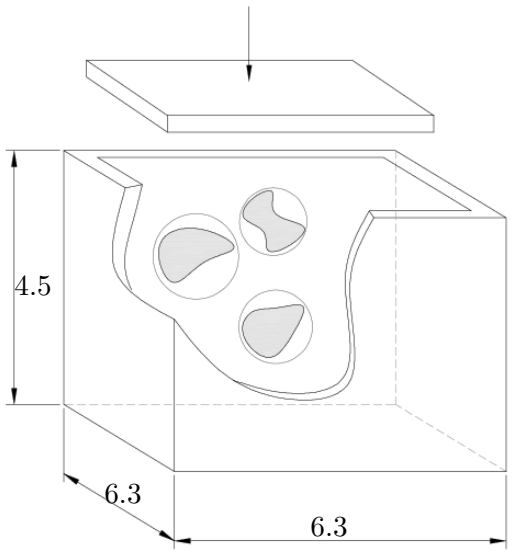
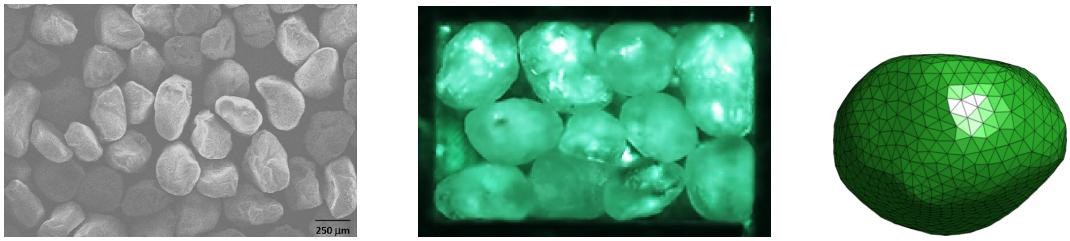


Figure 5.4: Illustration of sectioned containment box with lid, few sand grains and their minimum bounding spheres.

Particles Examples of real sand packings and an X-ray tomography are illustrated in Figure 5.5. The first assembly is a loose packing (Figure 5.5(a)), while the second is more dense and obtained by applying an external load (Figure 5.5(b)). The Ottawa sand is widely studied in engineering and is often considered a good representative example of sand. Its first order property are well-known and important for the numerical modelling; however, to save computing time, it is often approximated by spherical particles (65, 74).

This study uses the sand grain geometry in Figure 5.5(c), that corresponds to a high-resolution 3D X-ray tomography of Ottawa sand. This is repeated and scaled to obtain the size distribution outlined in Table 5.3. Even though not necessarily representative of a specific scenario, the distribution in Table 5.3 serves the purpose of this study, that is, the assessment of contact detection algorithms.



(a) (b) (c)

Figure 5.5: Examples of Ottawa sand: loose assembly (a) from (237), dense packing (b) from (55), and concave grain from 3D X-ray scan (c).

Table 5.3: Size distribution of the particles used for the numerical tests.

Radius	Percentage
3.55×10^{-1} mm	7%
2.50×10^{-1} mm	76%
1.75×10^{-1} mm	17%

5.3.4 Initial and boundary conditions

Initially all bodies are at rest. At time $t = 0$ sec the simulation begins and the grains and the lid fall under gravitational load. The lid however is constrained to remain perpendicular to the bottom of the box, which is fixed in all directions. This affects the final assembly, but makes straight forward the post-processing of the results presented in the following section.

The pre-processing operation assigns the material properties to the geometrical entities, respectively described in Sections 5.3.2 and 5.3.3, and geometrically assembles the grains into the containment box. These operations are performed in the following order:

1. Generate the mesh of a reference grain. The refinement level is defined by the minimum and maximum edge lengths of the tetrahedral elements, respectively $H_{\min}$ and $H_{\max} = 1.25\ H_{\min}$.
2. Generate a list of N bounding spheres with radius distribution defined in Ta-

ble 5.3 and assemble them with the geometrical packing algorithm described in (55) (since better than (92)). All spheres must be included and not overlapping with the walls of the containment box.

3. Replace each bounding sphere with the meshed grain scaled according to the radius of the sphere and apply to it a random rotation. Notice that, since the bounding spheres do not overlap, the grains randomly rotated do not overlap.
4. Place the lid of the containment box 2×10^{-1} mm above the highest node of the particle meshes.
5. Generate the mesh for the walls and the lid of the containment box.

Notice that, while the size distribution remains invariant, the third step introduces a remarkable variation in the distribution of the particle volumes. In fact, a grain has smaller volume than the bounding sphere which replaces it. If the sand particles were modelled as simple spheres, the probability density function of the particles volume would vary as shown in Figure 5.6. This of course means that particular care must be taken when modelling sands grains starting from a list of bounding spheres; the effect observed here is that the first order properties may be strongly alternated when following the steps above. An optimisation process would be required to retain the same size and volume distribution, however this would not affect the contact processing and is therefore ignored in this study.

An important aspect which does affect the contact processing is the mesh refinement of the sand grains. As depicted in Figure 5.7, the geometrical assembly algorithm presented in (55) positions a number of bounding spheres placed and the polygonal grains are placed within them (step 3) before the random rotation is applied. Figures 5.7(b)-(d) show the three levels of mesh refinement for three values of $H_{\min}$: 0.18µm , 0.10µm

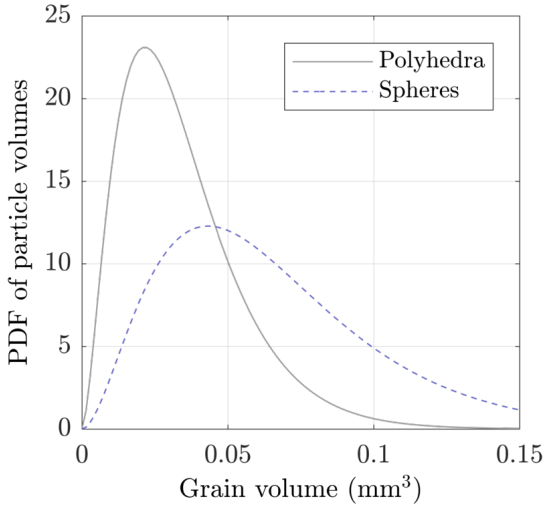


Figure 5.6: Probability density function (pdf) for polyhedra (fine mesh) and sphere volumes with same radii distribution.

and 0.07µm. The results obtained with these three levels of mesh refinement are presented in the next section.

5.4 Results

The results obtained from the mechanical packing of 100, 1200 and 2400 grains are presented in this section. The analysis is executed for polygonal grains and spherical particles as these settle. This is terminated after 5 s, when the kinetic energy reaches a plateau.

Each model is solved with the in-house code DEST and with the commercial software LS-DYNA (143). Both solvers use an updated Lagrangian finite element formulation, and the time integration is carried out explicitly. Moreover, two versions of the in-house code are tested: with and without the new hierarchical contact search described in Chapter 3.

The comparison between DEST and LS-DYNA aims to validate quantities such as void ratio e and kinetic energy, but it cannot be extended to CPU time for three

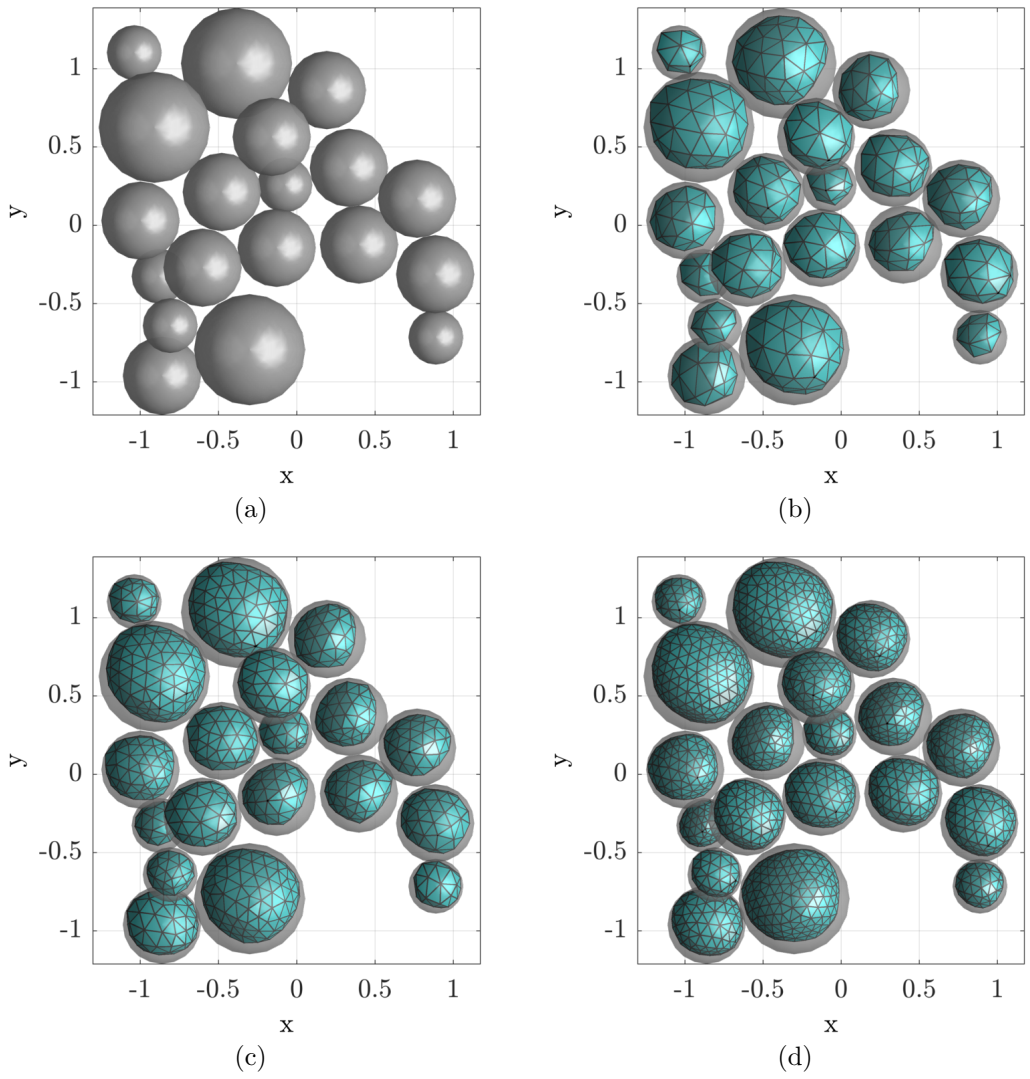


Figure 5.7: Geometrically pre-assembled spheres (a) used to bound polygonal grains discretised with coarse (b), medium (c) and fine (d) meshes.

reasons. Firstly, because the two solvers use substantially different criteria for defining the critical time-step; secondly, because they solve contact constraints with different formulations; and, finally, the executables of the two solvers were generated with different compilers. Furthermore, LS-DYNA very little is known about the algorithms implemented in it. For these reasons, a comparison of CPU time is meaningful only for the two versions of DEST.

The post-process pays particular attention to the mesh sensitivity of the final position of these particles. All tests run on a Linux machine with Intel® Xeon® E5-2630 and 32 GB RAM. Double-precision is used for all solvers.

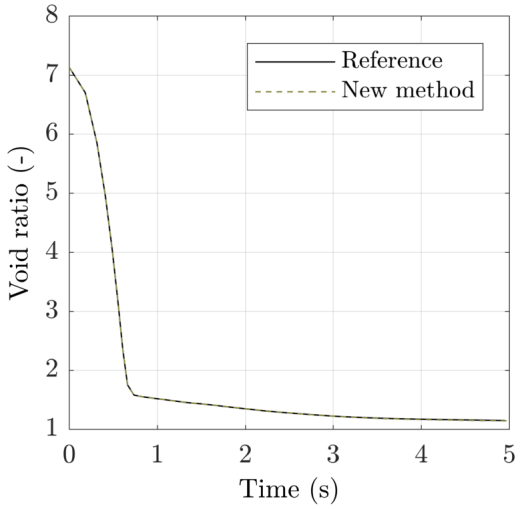
5.4.1 Software verification

The tests presented in this section aim to: (i) assess the correct software implementation, and (ii) validate the assumption made in Chapter 3 about the limited number of bodies mutually in contact[1].

Firstly, the kinetic energy and the void ratio are compared using different solvers to verify the correct implementation. The results are presented in Figure 5.8 which shows that the void ratio obtained with the reference version and the new version of the in-house code match almost perfectly. More results are shown in Figure 5.9, which focuses on the time history of the kinetic energy predicted by the in-house solver and LS-DYNA. The solves present congruent results only at the beginning and at the end of the simulation, while bodies sediment the dissipation of the kinetic energy is different. LS-DYNA presents a mild decay rate as well as spurious oscillation. Given that both

[1] In Sections 3.3 and 4.6, it was postulated that only few bodies can be mutually in contact, and on this argument it was assumed that the novel hierarchical contact search would be computationally sustainable.

model add the identical damping coefficient to the contact processing, it not clear why LS-DYNA generates such oscillations. Again, the fact that LS-DYNA is closed-source results the major constrain to further investigation; however, since both solves converge to the value, which is the stopping criterion, the implementation is validated.



(a)

Figure 5.8: Verification of the evolution of void ratio for different contact search algorithms.

Let us now compare the void ration e obtained with DEST and with LS-DYNA, respectively the in-house and the commercial solver. Figure 5.10 illustrates the time history for three assemblies with 100, 1200 and 2400 grains, respectively. In the three graphs, the curves show a little discrepancy at the beginning, which however decreases to almost zero as the simulations terminate. This is due to the random rotation assigned to each grain at the beginning of the simulation which affects the sedimentation time, and it has an insignificant influence on the final void ratio. In fact, the discrepancy reduces as the simulation continues and is almost null when the termination time $t = 5$ s is reached. Notice that the whole time history is here considered for the sake of validation only, the final value of the void ratio is of course the most important value.

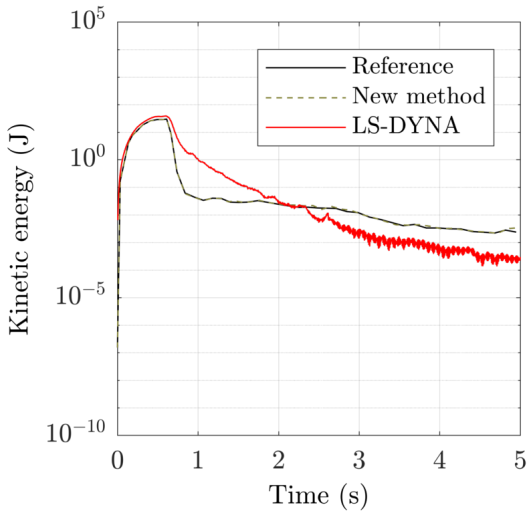


Figure 5.9: Comparison of the evolution of kinetic energy using different contact search algorithms for the in-house solver and the commercial code LS-DYNA.

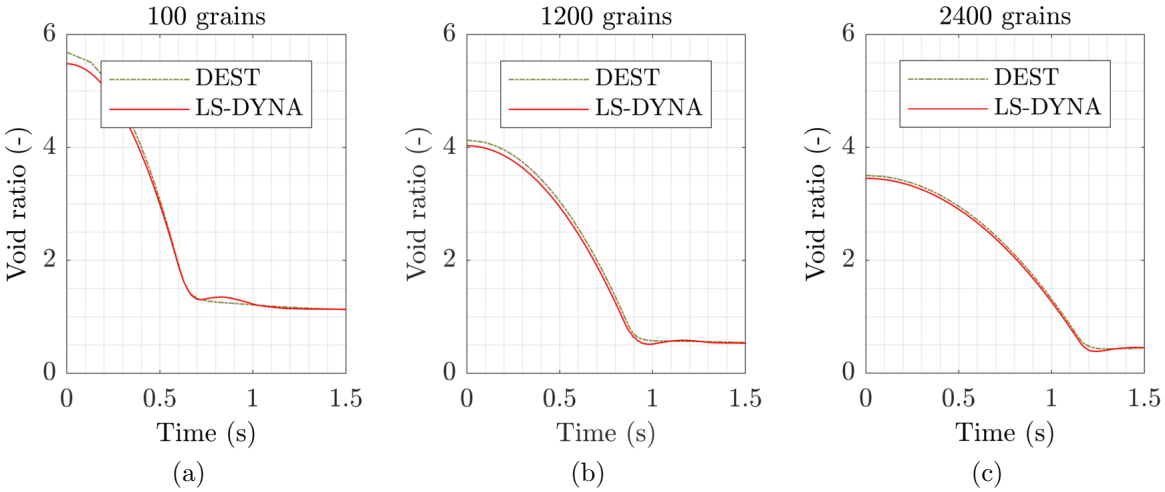
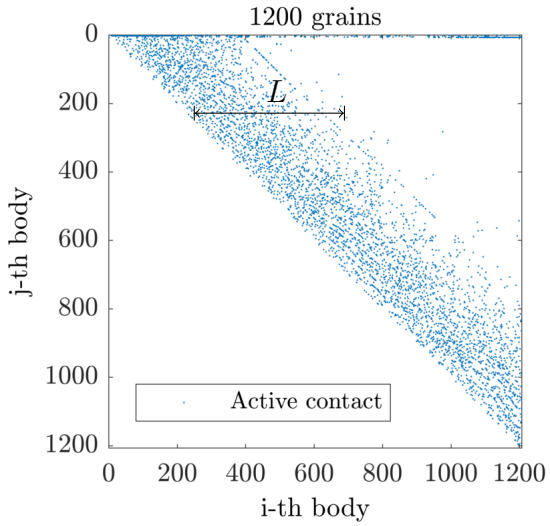


Figure 5.10: Void ratio history comparing the in-house code DEST with the novel contact detection algorithms, and LS-DYNA for 100 (a), 1200 (b) and 2400 (c) grains discretised with fine meshes.

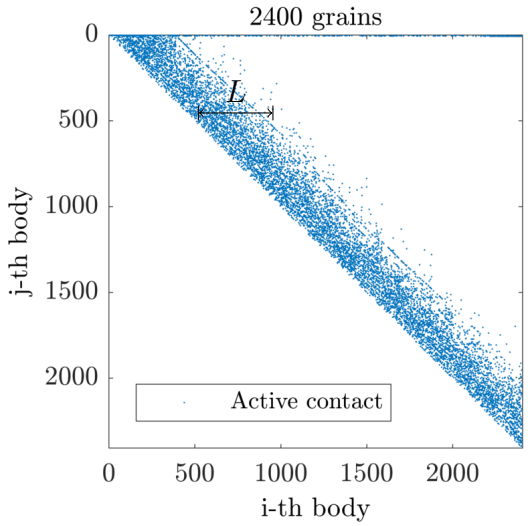
The outcome of the models above may now be post-processed to validate the theoretical analysis discussed in Section 4.6. After 5 sec of simulation time, when the particles and lid are basically at rest, the label of the bodies in contact are exported. These populate the i-th column and j-th row of the upper triangle of a N-by-N matrix, where N is the number of bodies in the simulation. Figure 5.11 illustrates this matrix for 1200 and 2400 sand grains. The matrices have a common pattern which is a consequence of the algorithm defining the bodies and is not particularly relevant in itself; it can be observed however that the latter has a more dense filling, whereas the matrix bandwidth (denoted by segments of equal length L) is comparable in both cases. This is because the active number of contact pair increases, but the number of bodies mutually in contact remains essentially the same. This is an important result that is useful for the validation of the assumption made in Chapter 3 about the limited number of bodies mutually in contact. For the sake of completeness, it is worth recalling from Chapter 4 that for less than 4 mutually contacting bodies the new contact search is advantageous. For more than 4, it may become excessively expensive. Moreover, one may observe at the top of both graphs in Figure 5.11 a thin area densely marked; this corresponds to contact pairs formed by lids, containment and grains.

The number of mutually contacting bodies can be obtained by processing further the data illustrated in Figure 5.11. For the test involving 1200 concave grains, 4881 active body–contact–pairs are found. By counting the number of non-empty rows and columns of the matrix it can be found that less than 5% of the particles involved are mutually in contact with other five or six particles. A similar result is found for the model with 2400 particles.

The results in Figure 5.12 show that the hypothesis made in Chapter 4 is valid. In



(a)



(b)

Figure 5.11: Visualisation of bodies mutually in contact for 1200 (a) and 2400 (b) grains. The body–contact–pairs which have their bodies touching or overlapping are marked with a dot. Notice that the bandwidth, approximated by a segment L, is comparable in both cases.

fact, the percentage of grains which mutually collide with five or more grains is only a tiny percentage, whereas the majority collide with three or less. Figure 5.12(a) and Figures 5.12(b) show these results for the tests with 1200 and 2400 grains.

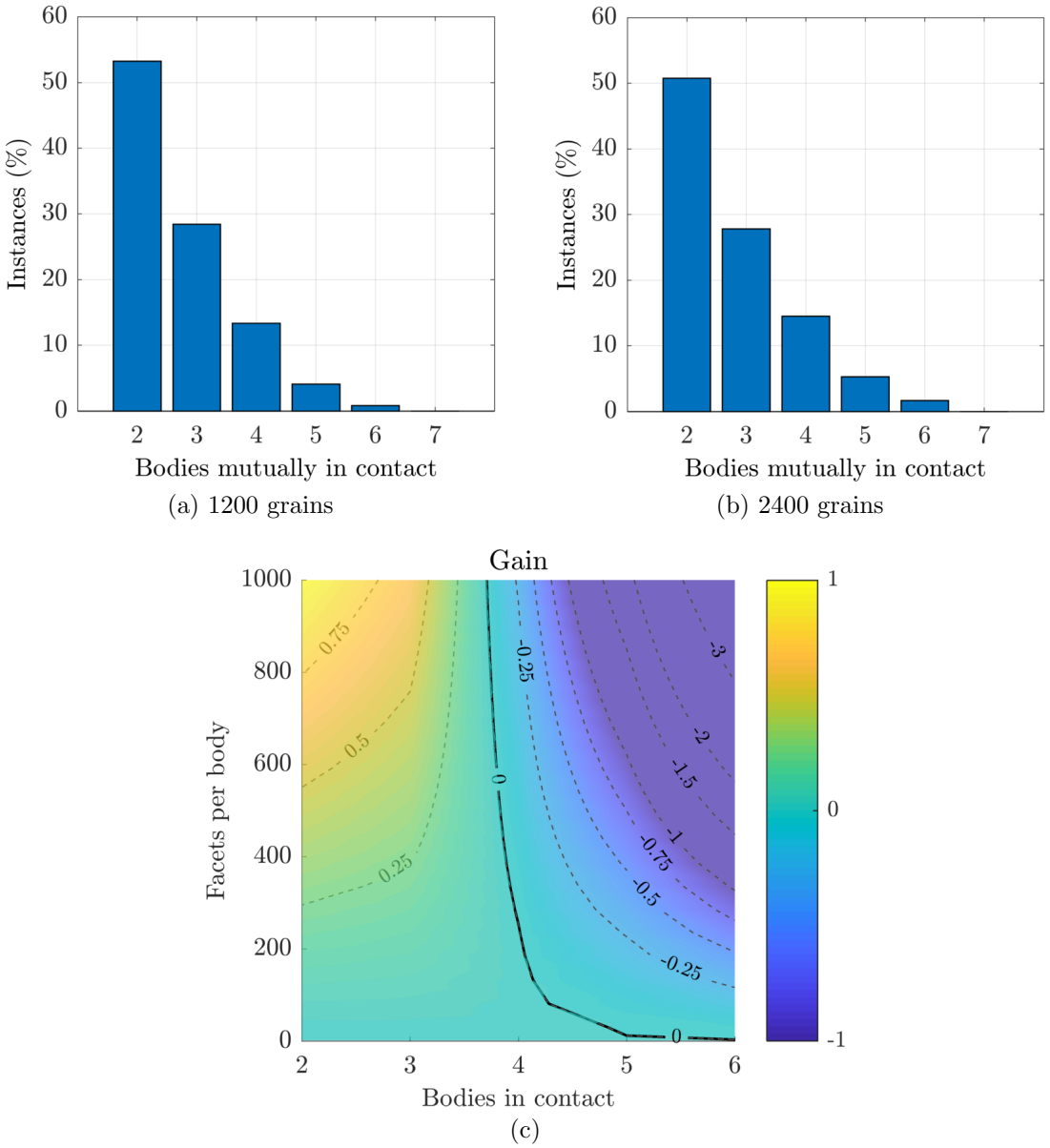
The results presented in this section have demonstrated that the new algorithmic framework is sustainable and can, at least qualitatively, save computing time while retaining accuracy. The next section aims to quantify this saving.

5.4.2 CPU time

The results of three models with 100, 1200 and 2400 grains are presented. Each model is discretised with three different meshes: coarse ($H_{\min} = 0.18\mu\text{m}$), medium ($H_{\min} = 0.10\mu\text{m}$) and fine ($H_{\min} = 0.07\mu\text{m}$). Each of these compares the CPU time spent in collision detection by different implementations of the in-house solver. As already mentioned, it is not possible to provide a comparison between the DEST and LS-DYNA since they use different formulae for computing critical time-step and contact forces. For example, LS-DYNA terminates the most demanding simulation in a third of the clock-time, but DEST runs three orders of magnitude more iterations.

The comparison of CPU time spent for collision detection is shown in Figure 5.13. Each graph presents the CPU time used by the reference and the new solver, that is: the in-house code before and after the implementation of the new algorithms.

The CPU time reduction is, for all cases, between 35% and 60%. The reason behind such a high speed-up is the tight bounding volume which employed the GJK algorithm in the broad phase. This reduces dramatically the number of overlapping AABBs and therefore the calls made to the most expensive facet-to-facet test functions. As mentioned in Chapter 4, the GJK algorithm is invoked and triggers AABBs bisection



(a) 1200 grains

(b) 2400 grains

(c)

Figure 5.12: Distribution of bodies mutually in contact for models with 1200 (a) and 2400 (b) grains. Figure 4.23 is here reported in (c) for completeness and to recall that the new contact search is advantageous only for 4 or less mutually contacting bodies.

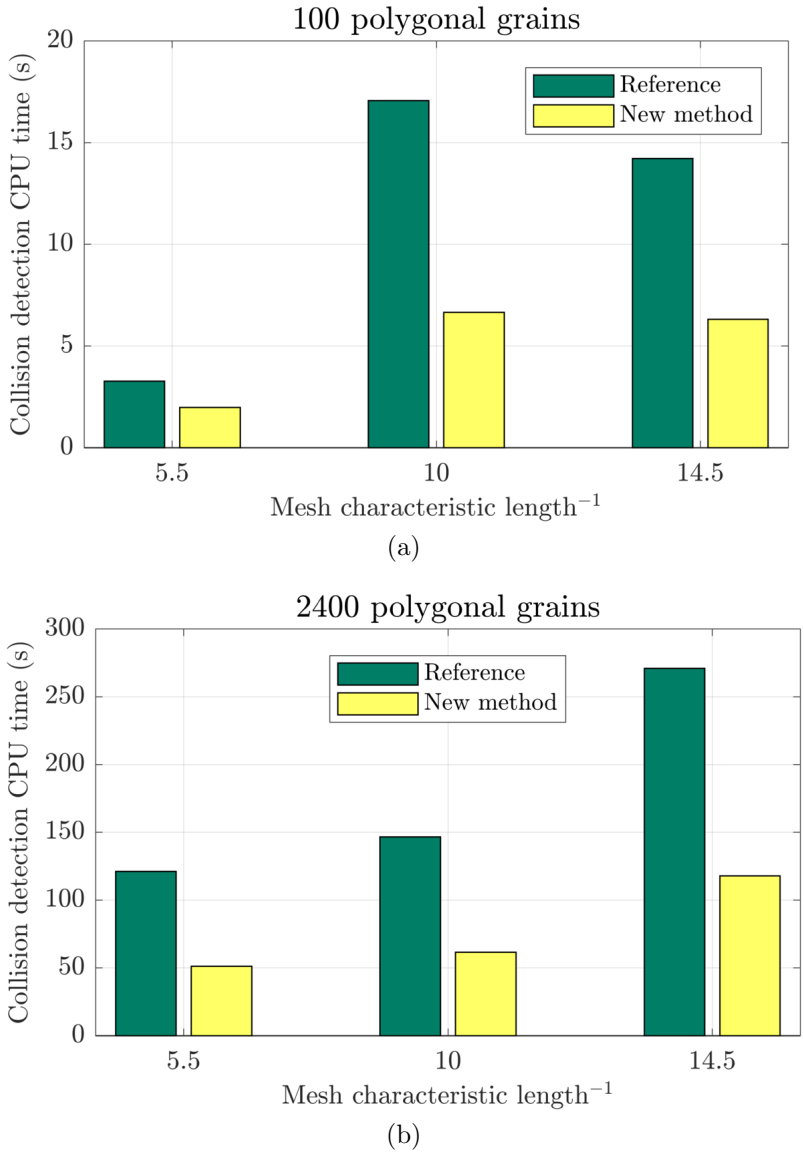


Figure 5.13: CPU time dedicate to contact detection of 100 grains (a) and 2400 grains (b) using DEST, the in-house code, before and after implementing the novel contact search algorithms.

trees to be build, updated and processed only when the bounding volumes of two sub-bodies are in contact.

The amount of CPU time required by the GJK algorithm is shown in Figure 5.14. The three curves represent the seconds of CPU time required to solve the model with 100, 1200 and 2400 grains, respectively. The CPU time increases as the mesh size is reduced or as the number of grains increases, but the gradient of all curves decreases

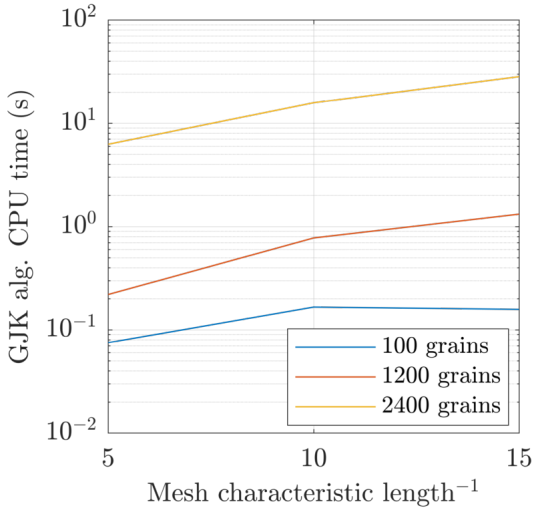


Figure 5.14: CPU time required by the GJK algorithm for the simulation of sand deposition using DEST.

for finer meshes and the reason for that is non-trivial.

Further investigation have shown that the steep increase follows from the variation of active body–contact–pairs. These are fewer when employing a coarse mesh, but the number increases and converges to a unique value when reducing the edge length $H_{\min}$. This suggests that the mesh refinement influences both the contact processing computing time and the final assembly. The next section pays further attention to this observation.

5.4.3 Mesh sensitivity analysis

The drastic reduction of CPU time for contact processing makes sustainable the simulation of sand grains discretised with fine meshes, thus enabling simulations to include realistic grain morphologies. The mesh sensitivity analysis presented in this section shows the effect that a finer mesh, closer to the realistic morphology, has on the final assembly. This analysis examines whether the solution depends on the mesh or not. Whilst this is an ordinary test for continuum-based methods, it is unfortunate that in

Table 5.4: Coordination number Z obtained for models with 1200 and 2400 grains using different edge lengths $H_{\min}$.

$H_{\min}$(µm)	Number of grains	Z
0.15	1200	8.09
0.15	2400	8.02
0.10	1200	8.15
0.10	2400	8.68
0.07	1200	9.25
0.07	2400	8.67

DEM the influence of mesh refinement is often overlooked. Section 5.4.2 has shown results suggesting that the mesh refinement affects parameters such as coordination number and void ratio. These observations are now closely inspected.

Let us consider the simple test in Figure 5.15 which involves 100 concave grains. These are discretised with a coarse and a fine mesh. Upon termination of the simulation, the lid finds a rest position which varies for the two mesh refinements.

The example in Figure 5.15 readily demonstrates the influence of the mesh over the final packing, and therefore motivates the following mesh sensitivity analysis.

Coordination number The coordination number Z, introduced in Eq. (5.2), is computed in a post-processing phase for the models with 1200 and 2400 grains.
In all tests, Z is found dependent on the mesh resolution, in particular, it increases as the mesh is refined. The results summarised in Table 5.4 show that Z can vary up to 10% as the mesh is refined.

Void ratio The void ratio e, defined in Eq. (5.1), is measured at runtime for grains modelled as spheres and as realistic polygonal meshes. A comparison of the time evolution of e with spherical, coarsely and finely meshed particles is shown in

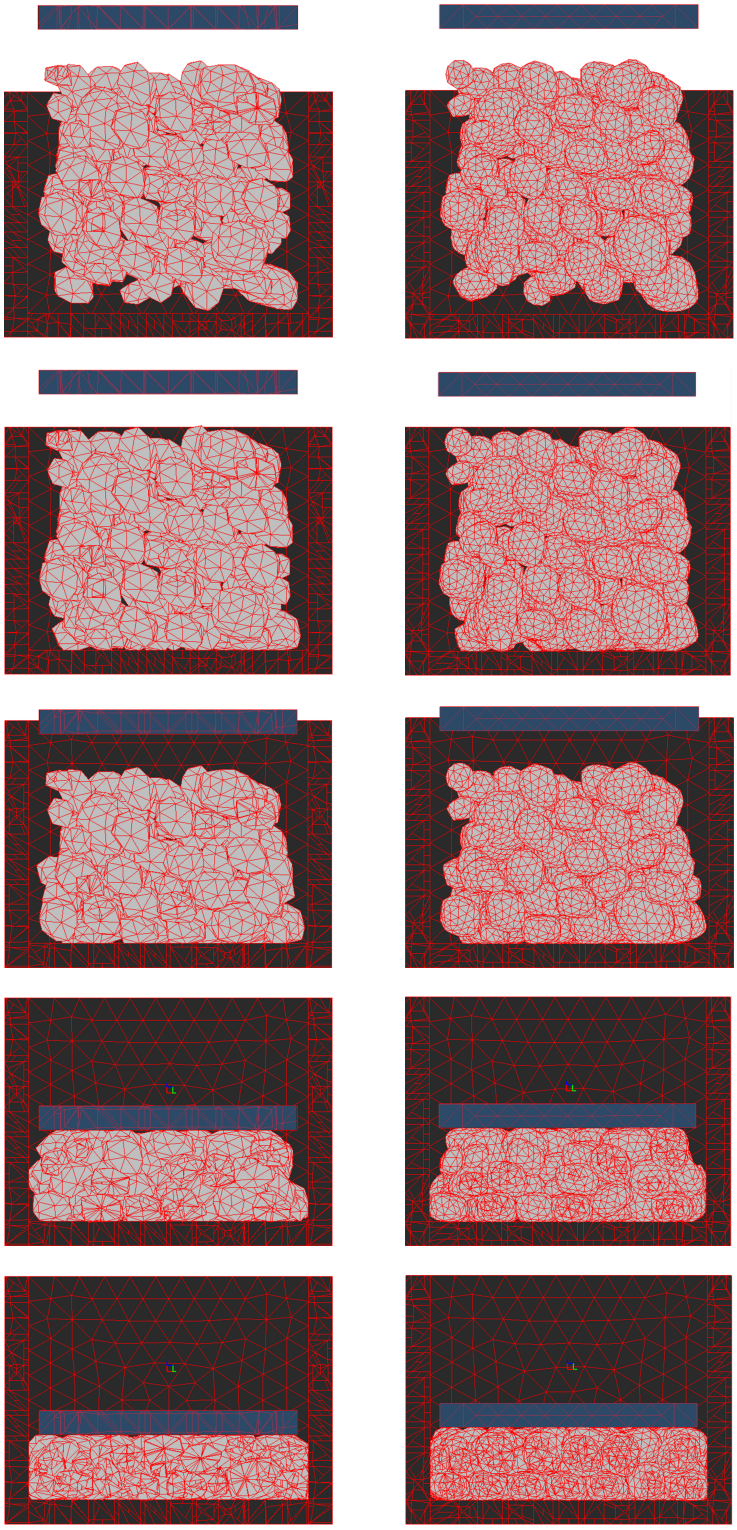


Figure 5.15: Comparison between packings of 100 grains discretised with coarse (left) and fine (right) meshes.

Figure 5.16. These results are obtained from the simulation illustrated in Figure 5.15 with 100 grains. The large initial discrepancy is irrelevant, what is important is the convergence shown after the first second of simulation. The results for the last three seconds of simulation, magnified in the zoom area, show that the curves remain about 10%-20% apart from each other. The coarse mesh predicts a higher value of e, whereas spherical particles are in between the two.

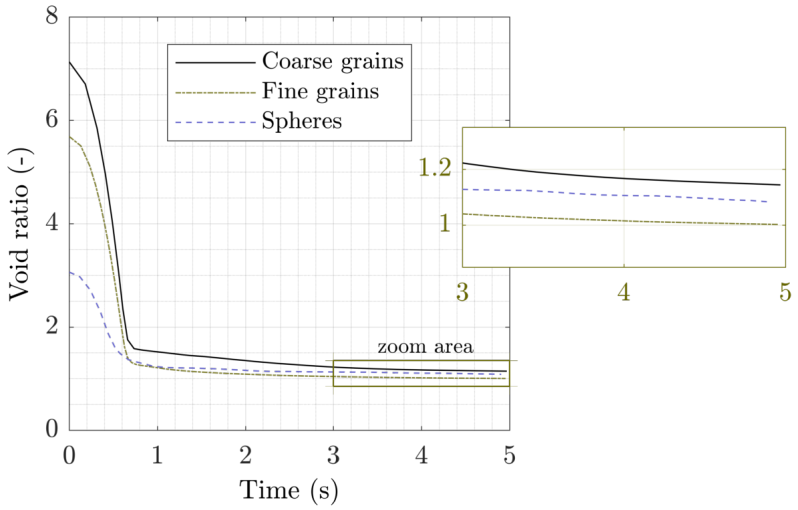


Figure 5.16: Effect of mesh refinement on void ratio for 100 grains.

Notice that the value of the void ratio represented in Figure 5.16 is not physically representative. Firstly, because the number of grains is too small, and also because the box boundaries affect the value of e. To remove this effect, an extra post-processing step is required, but it is here ignored since beyond the scope of this study — which is not to compute e of a particular sand, but rather to evaluate its sensitivity to the mesh size.

The final void ratio for all mesh refinements is shown in Figure 5.17 for 100, 1200 and 2400 particles. It appears that the medium and the fine meshes converge toward the value of 0.4, whilst coarsely meshed grains remain above 0.6. Furthermore, when

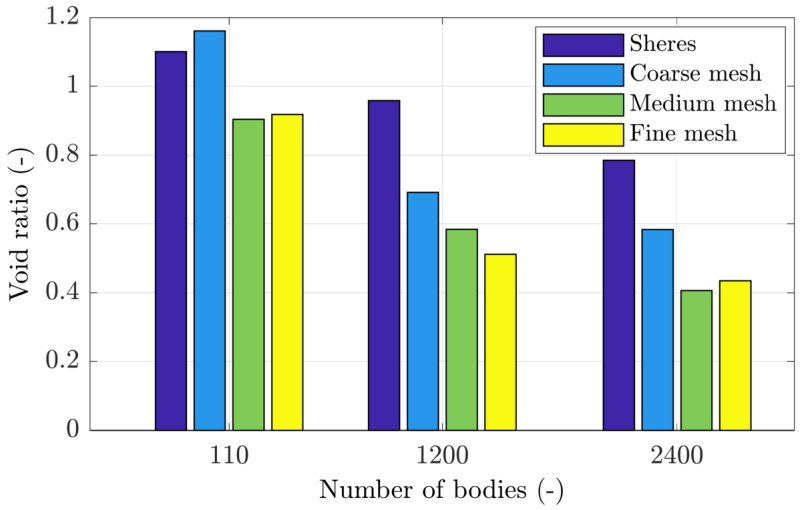
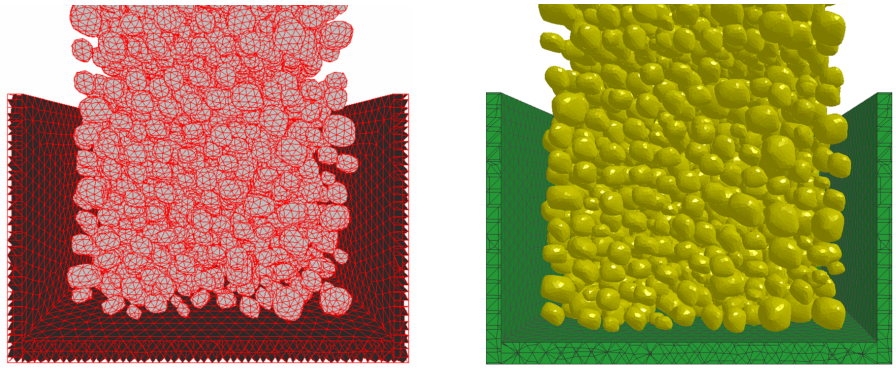


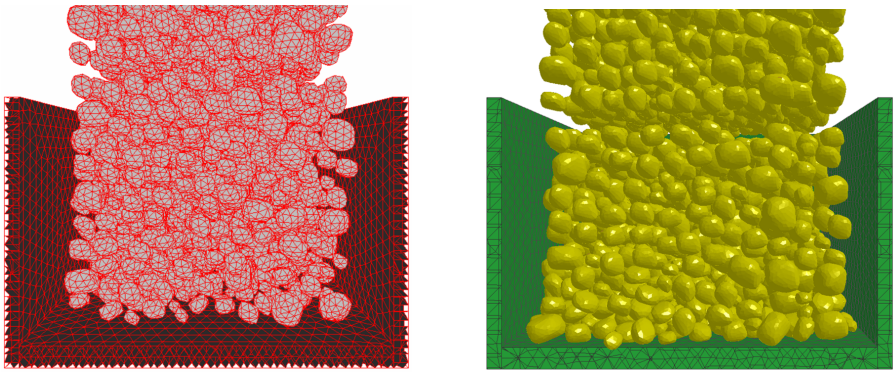
Figure 5.17: Final void ratio for spheres and polyhedral grains with different mesh refinement levels.

the grains are approximated with spheres, the void ratio is even higher: for the model with 2400 particles, this is nearly double the value predicted by fine polygonal mesh.

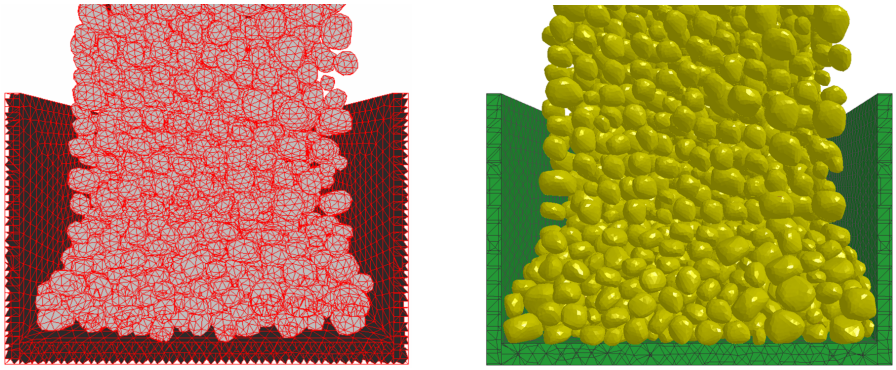
Finally, a visual comparison of the results with 2400 grains is presented. Two mesh refinements are compared: medium $H_{\min} = 0.10\,\mu\text{m}$ and fine $H_{\min} = 0.07\,\mu\text{m}$. The former is visualised by means of the in-house software DEST, the latter using the commercial solver LS-DYNA — the choice of different visualisation software is purely a matter of taste and it has already been shown in Figure 5.10 that these solvers produce similar results. The first column of screenshots in Figure 5.17 shows the results obtained with the medium mesh refinement, the second column regards the finest mesh. By comparing the time frames, it is possible to see that the position of the lid is identical until $t = 1.00$ s. Afterwards, at $t = 1.25$ s, the solution shows mesh dependency and, upon termination at time $t = 5.00$ s, this influences the height at which the lid settles.



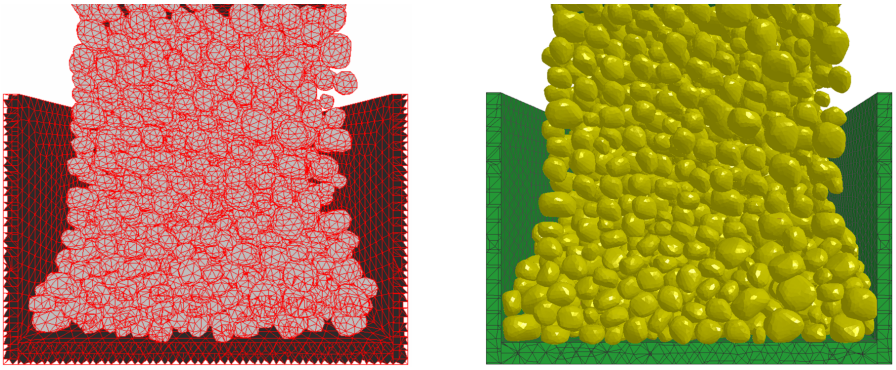
(a) DEST at $t = 0.00$ s (b) LS-DYNA at $t = 0.00$ s



(c) DEST at $t = 0.45$s (d) LS-DYNA at $t = 0.45$s

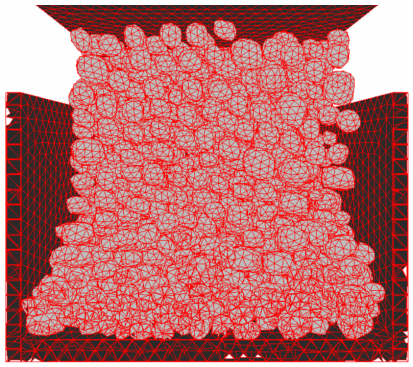


(e) DEST at $t = 0.80$s (f) LS-DYNA at $t = 0.80$s

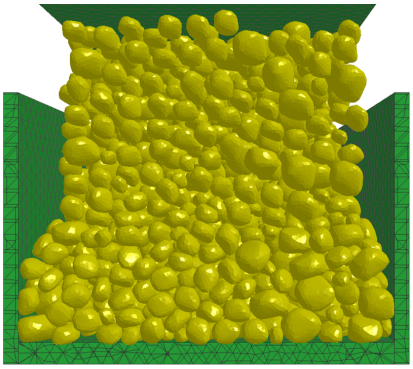


(g) DEST at $t = 0.85$s (h) LS-DYNA at $t = 0.85$s

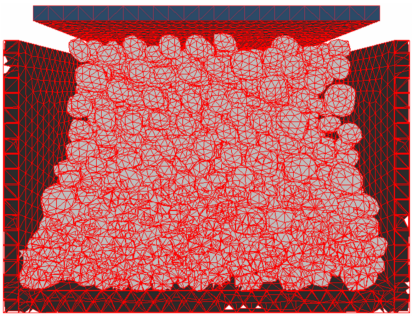
Figure 5.18: Mechanical packing of 2400 grains meshed with minimum edge length $H_{\text{min}} = 0.10\mu\text{m}$ using in-house DEM/FEM solver DEST, and with minimum edge length $H_{\text{min}} = 0.07\mu\text{m}$ using commercial solver LS-DYNA (continues).



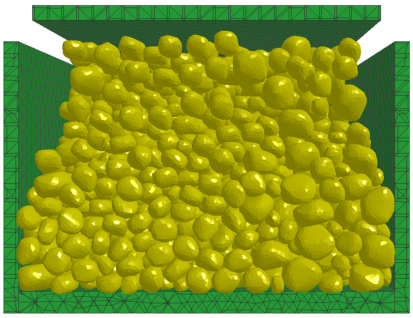
(i) DEST at $t = 0.92$s



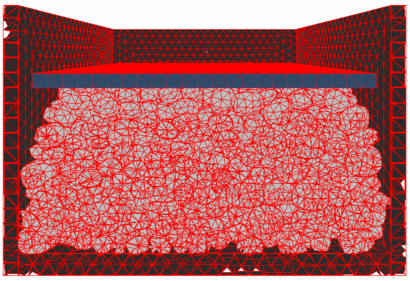
(j) LS-DYNA at $t = 0.92$s



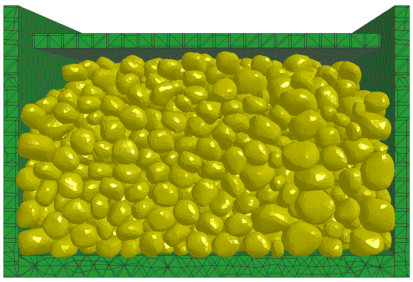
(k) DEST a $t = 1.00$s



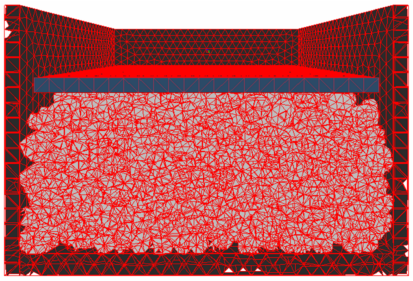
(l) LS-DYNA at $t = 1.00$s



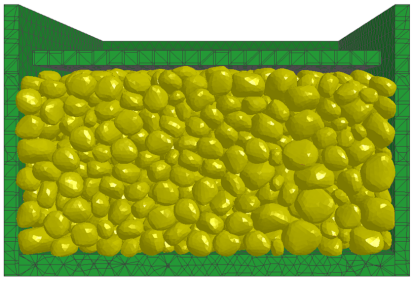
(m) DEST at $t = 1.25$s



(n) LS-DYNA at $t = 1.25$s



(o) DEST at $t = 5.00$s



(p) LS-DYNA at $t = 5.00$s

Figure 5.17: (Continued) Mechanical packing of 2400 grains meshed with minimum edge length $H_{\min} = 0.10\mu$m using in-house DEM/FEM solver DEST, and with minimum edge length $H_{\min} = 0.07\mu$m using commercial solver LS-DYNA.

5.5 Concluding remarks

This chapter has shown how the new distance algorithms can improve the simulations of contact problems. The particular application of sand particles was studied by means of a FEM model. The sand grains were modelled using realistic concave morphology and packed by a weight falling under gravitational load. The numerical results obtained with a in-house software were validated against the commercial solver LS-DYNA.

The new hierarchical framework, composed by the distance algorithms presented in Chapter 3, has shown a significant reduction in CPU time for the contact detection. Three assemblies with 100, 1200 and 2400 particles were studied and, in all cases, the saving were between 30% and 60%.

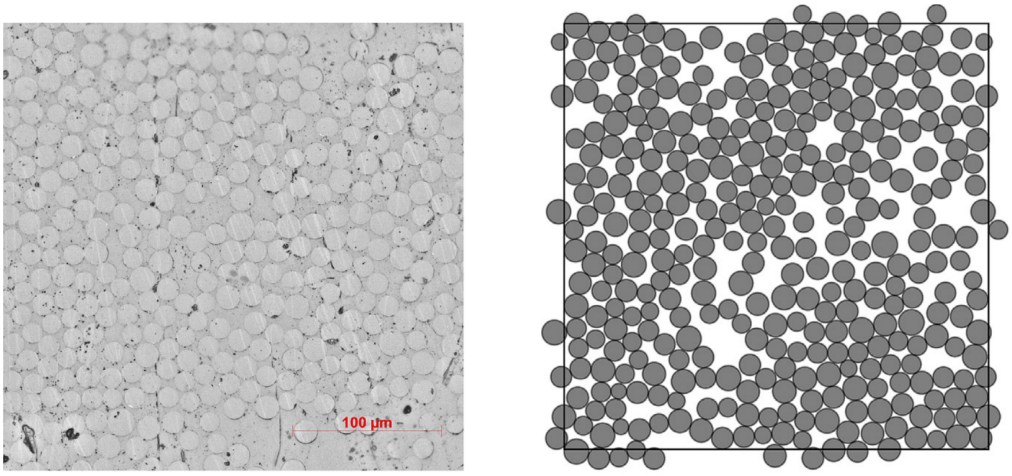
The tests were repeated for different mesh refinements to demonstrate the sensitivity of the final assembly to the mesh resolution. The value of the void ratio is found to converge toward a unique value as the mesh is refined. This conclusion can now be drawn thanks to the improved distance algorithms, which drastically reduce the computing time for the whole simulation.

Finally, an important hypothesis postulated in Chapter 3 about the performance of the hierarchical search was validated. It was found that only few bodies are mutually in contact, and this demonstrates the sustainability of the proposed contact search method.

Chapter 6

Optimisation of polycrystalline microstructures

This chapter illustrates how engineers and material scientists can enhance their analyses of heterogeneous microstructures by using the new methods presented in Chapter 3. These are applied to solve morphology optimisation problems for the generation of representative volume elements (RVEs) for multi-scale analysis. The optimisation process makes a large number of calls to distance functions which are the computational bottleneck for these problems — particularly the solution of distance queries between convex polytopes. Herein, the improved GJK algorithm is used to accelerate the convergence of such optimisation. Applications to RVE generation are shown for polycrystalline materials only, although the method is applicable to any material whose microstructures exhibits convex inclusions or voids. The tests presented in this chapter reproduce an experimental morphology obtained from X-Ray scan. Conclusions are drawn based on the comparison of CPU time and convergence rate for different GJK sub-algorithms.



(a) (b)

Figure 6.1: Real carbon fibre composite (a) and virtual RVE (b). From (187).

6.1 Introduction

The macromechanical properties of metals, ceramics, composite, and other heterogeneous materials, are influenced heavily by their microstructures. Properties which are relevant to engineers include: strength, ductility, thermal conductivity or permeability. In order to predict and to better understand the mechanisms determining these properties, homogenisation techniques have been developed. In recent times, multi-scale analysis has used homogenisation to include the microscopic effects into large-scale models.

Multi-scale modelling begins with the definition of RVE. The purpose of the RVE is to establish a link between micro-scale model and macro-scale models. Intuitively, an RVE may be seen as a "small cell" of apparently homogeneous material at the macro-scale, but heterogeneous at the micro-scale.

From the geometrical point of view, an RVE is a model defined over a simple domain within which several inclusions are scattered. The morphology and distribution of the inclusions are defined by algorithms called *RVE generator* which take experimental

data and generate the geometry for a micro-scale model. For example, the diameters of the fibres in a glass fibre/epoxy composite may be measured experimentally from a microscopic image similar to the one in Figure 6.1(a). The geometry of the corresponding RVE, illustrated in Figure 6.1(b), is generated so that it statistically reproduces the measurement on the fibres.

The generation of numerical RVEs requires few steps. Since FEM is the most widely adopted method for multi-scale analysis, let us focus on the steps required to create an RVE for FEM:

Step 1: input experimental data,

Step 2: populate the computational domain with inclusions or voids, and

Step 3: mesh the microstructure (and occasionally re-mesh at runtime).

Several studies following these steps have been published. For example, investigations on open cell foams (108), composites (109), the evolution of surface roughness of metal under dynamic loading (251), additive manufacturing (107), the temperature effect at the grain boundary of metals (126) and synthetic polycrystalline materials (8, 57). However, none of these methods could accomplish the three steps above without difficulties.

The problem of generating a micro-scale FE model does not admit closed-form solution and an optimisation procedure is therefore required. This takes in input data that could either be *specific* to a given microstructure, or *generic* to a family of materials. In both cases the procedure follows a sequence similar to the diagram in Figure 6.2. Firstly, experimental data are interpreted to form an initial guess. This is then used to initiate the iterative optimisation which continues until one or more convergence criteria are met. The outcome is a geometry that quantitatively reflects

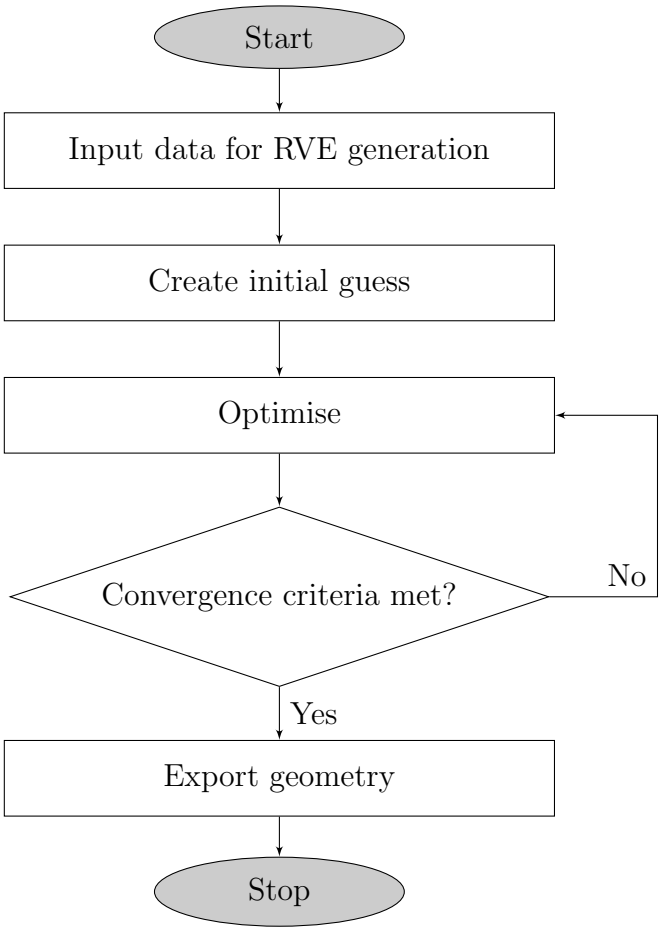


Figure 6.2: Flow chart for common RVE generation algorithms.

the input data.

Overall, generating an RVE involves frequent calls to distance functions that optimise the morphology. Beside being the most expensive task, distance queries also play an important role on the convergence rate of the optimisation.

6.2 Motivations

All steps in the RVE generation involve distance queries. Step 2, which populates the domain, generates a virtual microstructure by assembling into the RVE a large number of polydisperse inclusions. These can have complex morphologies, sometimes unknown a priori, and they must fulfil certain geometrical constraints, such as peri-

odic boundaries or non-overlap. These constraints are usually enforced by mean of distance queries. Step 3, which meshes the RVE, requires a pre-processing step for the regularisation of the geometry to guarantee high-quality meshing.

As mentioned in Chapter 2, there are many ways to represent voids and inclusions. In composites materials, the fibres are simplified as monodisperse disks (7, 68, 102) or cylinders (40). More rarely these are polydisperse (187). Ceramics and metals may be roughly approximated by regular polytopes, such as cubes (203, 255). Regular inclusions are computationally simple to handle, but have limited ability to represent realistic microstructures. Complex shapes might be represented by level sets (212) or phase field functions (115, 201), but the computational complexity is rarely paid off and their applicability seems confined to foams or single crystals (9). These examples emphasis that distance algorithms must be versatile to cope with more than one representation.

This chapter focuses on irregular polytopes, which are widely adopted and allow the numerical modelling of many heterogeneous materials. Polyhedral microstructures can represent accurately the size distribution of cells and other morphological parameters (e.g. sphericity). The straight edges and flat surfaces provide computational advantages from the geometrical point of view and, despite their "regular" geometries, they have shown good results in challenging scenarios of polycrystalline aggregates (66, 78) or open and closed cell foams (198, 229, 241). Even for polytopes however, the optimisation cost is governed by the frequent calls to distance functions. These are the major computational bottleneck to RVE generation.

Little research has been devoted to reduce the computing time of RVE generators. Most of the published work focuses on inclusions described by quadrics (e.g. spheres,

ellipses or compounds of the two) and yet, recent works show, assembling one thousand fibres in a composite RVE takes between 1.5 and 10 hours (CPU time) (187, 220). This time increases when RVEs use polytopes an arbitrary number of edges and faces. The reason is the higher cost for the evaluation of support mappings (see Chapter 2 for further details) and for imposing geometrical constraints (e.g. tangency, exact distance) between inclusions. Therefore, irregular polytopes require more complex algorithms than quadrics. These often are the computational bottleneck which limits the generation of large and elaborated RVEs.

Aiming to improve the generation of RVEs, the following sections investigates the applicability of the methods presented in Chapter 3 for the generation of virtual microstructures. Rather than defining the RVE for a specific material, this research focuses on algorithmic details for general RVEs of materials whose microstructure contains crystals or voids described by irregular polytopes.

6.3 Generation of microstructures

The major challenges to RVE generation are due to the large number of grains and their complex morphology. Previous works on ice (123), magnesium alloys (72) and ceramics (239) have shown that a single RVE require between 10^3 and 10^6 crystals and that each crystal has a unique morphology. For this reason, a suitable mathematical framework that allows to address such a complex scenario has to be identified.

Voronoi diagrams are by far the most commonly used tool for describing the geometry of polycrystalline microstructures. They offer a good compromise between versatility and computational cost. Geometrically, Voronoi diagrams are a family

of spatial partitioning techniques (177). They are used to divide the 3D space into non-overlapping regions, usually known as *cells*. A partition requires a set of points $P = \{p_i : i \in I_P\} \subset \mathbb{R}^m$, with $I_P = \{1, ..., m\}$, and a weight $w_i \in \mathbb{R}$ associated to each point. A Voronoi region $V(p_i)$ is defined for all points in P so that $\bigcap_{i \in I} V(p_i) = \emptyset$ and $\bigcup_{i \in I} V(p_i) = \mathbb{R}^m$. The point p_i is called *seed*.

A type of Voronoi diagram differs from another by the formulae that establishes whether a point $p \in \mathbb{R}^m$ belongs to $V(p_i)$ or $V(p_j)$, with $i, j \in I_P$. The usual approach is to define *general dominance cell* $H_w(p_i, p_j)$ of a point p_i over p_j:

$$H_w(p_i, p_j) = \{p : d_w(p, p_i) \leq d_w(p, p_j)\,,\; i \neq j\} \tag{6.1}$$

where the function d_w is the *weighted distance* between two points. It should be emphasised that d_w effectively defines the morphology of the frontier between neighbouring cells and hence, for this specific application, the grain boundary. The subscript w highlights the dependence of the distance function from the weights.

Eq. (6.1) is a generalised form from which a *generalised Voronoi diagram* $\mathcal{V}_w$ is obtained by extending the idea of dominance cells to all points in P. $\mathcal{V}_w$ is the set of all cells $V(p_i)$ such that:

$$V(p_i) = \bigcap_{j \in I_P \setminus i} H_w(p_i, p_j) \quad \text{for} \quad I_P = \{1, ..., m\}. \tag{6.2}$$

In practise, once d_w is specified, a Voronoi diagram is able to approximate a polycrystalline microstructure whose crystals correspond to $V(p_i)$.

To specify a digram $\mathcal{V}_w$, a distance function d_w has to be defined. Common choices of weighted distance functions d_w and associated weights w, are reported in Table 6.1. Each formulae can shape the grain boundary in different ways, as detailed in the table. The weights play a crucial role: on the one hand, they allow extra control over the shape

Table 6.1: Common definitions of weighted distance d_w for Voronoi diagrams.

Type	Function	Weight	Boundary in $\mathbb{R}^2$
Ordinary	$d_w = w_i\|x - x_i\|$	$w_i = 1$	Straight line
Multiplicatively	$d_w = w_i\|x - x_i\|$	$w_i < 0$	Circle, arc or straight line
Additively	$d_w = \|x - x_i\| - w_i$: $\|x - x_i\| > w_i - w_j \geq 0$	$w_i \in \mathbb{R}$	Hyperbolic arc or straight line
Compoundly	$d_w = \frac{1}{w_{i1}}\|x - x_i\| - w_{i2}$	$w_{i1} > 0$ $w_{i2} \in \mathbb{R}$	Forth order polynomial curve hyperbolic arc, circular arc or straight line
Additively power	$d_w = \|x - x_i\|^2 - w_i^2$	$w_i \in \mathbb{R}$	Straight line

of grain boundaries, on the other hand, they rarely have a physical interpretation and add a layer of complexity to the computational procedure. For these reasons, to date, the only formulations exploited in engineering applications are those which construct simple cell boundaries: ordinary and additively weighted power diagrams.

The rest of this section illustrates how to generate polycrystalline microstructures starting from a raster image from X-ray scan using Neper (195), a software package for RVE generation and meshing. Neper is open-source and publicly available at http://neper.sourceforge.net/ licensed under the GNU General Public Licence.

6.3.1 Diffraction contrast tomography (DCT)

Neper allows to generate an RVE by matching a virtual reconstruction of a specific, real, microstructure. Diffraction contrast tomography (DCT) is a technique that makes use of X-ray diffraction to generate 3D *raster images* of existing microstructures. These images provide the target morphology that the optimisation process tries to reproduce.

A raster image is a set of voxels $\mathcal{G} = \{v_j\}$. Each v_j is labelled by means of a labelling function I to identify a unique grain $G \subset \mathcal{G}$. The i-th grain is therefore

described by a subset of voxels:

$$G_i = \{v_j \in \mathcal{G} : I(v_j) = i\}. \tag{6.3}$$

The grain G_i is approximated by the cell $V(p_i)$.

The position of the seed p_i and its associated weight w_i are found iteratively. To do so, Neper solves an optimisation problem that minimises the following object function:

$$\mathcal{O} = c\sqrt{\sum_{i=1}^{N} \delta_i^2} \tag{6.4}$$

where c depends on the scan resolution and δ_i is a measure of the morphological discrepancy between a grain G_i and its associate Voronoi cell $V(p_i)$. Essentially, the optimisation aims to reduce $\mathcal{O}$ by varying the morphology of $V(p_i)$ so that the discrepancy is small.

In practise, a DTC scan produces high-resolution images where the number of voxels N is in the order of billions, making Eq. (6.4) prohibitive to compute. A viable solution is to consider only the subset of voxels on the boundary of the grain: $\{v_k\} \in G_i^b$, so that the object function can be approximated as follows:

$$\mathcal{O} \approx c\sqrt{\sum_{v_k \in G_i^b} d(v_k, V(p_i))^2}. \tag{6.5}$$

Eq. (6.5) measures the discrepancy between the boundaries of the target tessellation obtained from DTC scan and the boundaries of the Voronoi cell. Smaller values of the objective functions are sought iteratively so that, upon convergence, the distance between the grain boundaries represent by voxels and Voronoi tessellation is reduced.

As reported in (196), Neper relies on the GJK algorithm to compute the distance between a voxel v_k and a cell $V(p_i)$. This is a natural choice since Eq. (6.5) requires to solve a distance query between a point and a cloud of points.

The evaluation of Eq. (6.5) has a tremendous impact on the optimisation process.

Firstly, the high volume of distance queries per optimisation step makes it an expensive task. Secondly, the rounding error occurring in the evaluation of the Euclidean distance propagates. Furthermore, because of the high-resolution of DTC scans, there is a high chance that voxels close to the faces of a cell cause instability issues to the GJK algorithm.

Unless the distance in Eq. (6.5) is computed accurately to machine precision, the convergence of the optimisation procedure, hence the generated RVEs, will be sub-optimal.

6.3.2 Mesh regularisation

Once a virtual microstructure is available, this can be meshed and used for numerical analysis. Polytopes with straight edges and flat faces are most commonly meshed with unstructured grids of tetrahedral elements. Meshing is a relatively simple procedure which essentially consists of subdividing a cell with tetrahedral elements of a desired element size.

It has been shown that fine meshes can improve the agreement between numerical and experimental results (255), however a cell may include edges significantly smaller that the desired element size. The short elements are a consequence of the unconstrained optimisation and they have been reported as source of numerical artefacts in various applications (17, 179, 195). When a cell with an edge shorter than a given element size is passed to a mesh generator, this creates one or more elements around that edge. The element thus generated has poor aspect ratio and can lead to unrealistic results in large-strain applications. Furthermore, short edges may constrain the time-step in explicit dynamic simulations.

Short edges can be removed by means of a *regularisation* process. The idea is to replace the edge whose length is below a threshold value with a vertex. Regularisation methodologies are presented in (174, 195).

Neper regularises a tessellation by merging two vertices of an edge with length below a threshold value into one. This of course influences the topology of the target edge and its neighbouring cells. Up to four (three) cells intersect in 3D (2D) at a single vertex; therefore, up to four (three) neighbouring cells may require adjustments as a consequence of an edge removal. The position of the new common vertex needs to be determined so that the impact on the microstructure is as little as possible.

The position of a new vertex p is computed so that the distances to the initial faces is minimal. A least-squares criterion is used to minimise the following function:

$$\mathcal{O} = \sqrt{\sum_{i=1}^{n_f} d_i^2} \tag{6.6}$$

where the distance d_i between the vertex and the initial faces of i-th face, for all faces n_f of the vertex.

While the computational cost of evaluation Eq. (6.6) is small compared to the optimisations steps, the accuracy of all distances d_i is crucial. The next section illustrates how the algorithms presented in Chapter 3 may improve this computation.

6.4 Improving the generation process

Currently, to solve distance queries, Neper implements the GJK algorithm and the Voronoi search as sub-algorithm (194, 196). As mentioned in Chapter 2, this is the simplest sub-algorithm and careful coding can make it robust, but its performance is notoriously poor (see (64) for implementation details and Chapter 4 for performance

evaluation). Since the GJK algorithm is invoked many times per optimisation step, it is reasonable to question whether a faster distance sub-algorithm could reduce the optimisation cost.

The next section represents the results of an investigation which compares three distance sub-algorithms for the generation of RVEs. In particular, the procedure described in Section 3.2 will be tested to reproduce the morphology for a microstructure obtained from DCT scan.

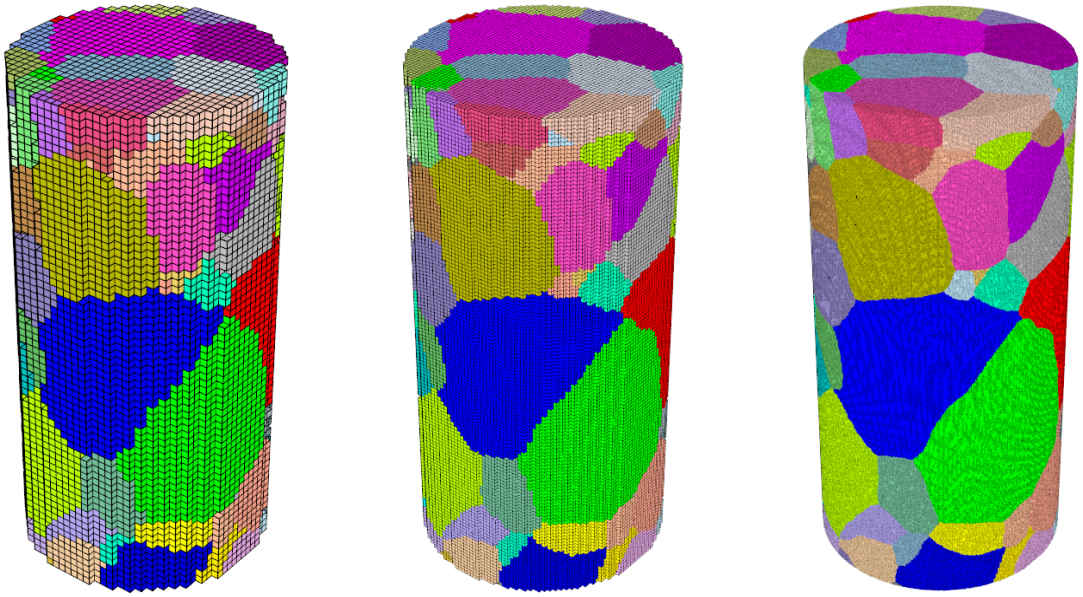
6.5 Results

Numerical experiments investigating the influence of different versions of the GJK algorithm on the generation of polycrystalline microstructures are presented in this section. Performance and quality of the resulting microstructures will be compared based on the metrics defined in Section 6.5.1. Three distance sub-algorithms are studied: (i) the Voronoi search algorithm currently implemented in Neper, (ii) Johnson's algorithm (including the Backup procedure as first described in (89)), and (iii) the Signed Volumes method introduced in Chapter 3. All tests are carried out on a Linux machine with Intel® Xeon® E5-2630 and 32 GB RAM using double-precision floating point arithmetic.

6.5.1 Metrics

The following quantities are investigated:

- computing time,
- number of iterations to convergence,



(a) (b) (c)

Figure 6.3: Raster image of DCT scan. Three resolutions: (a) low, (b) medium and (c) high.

- length of the edges of the Voronoi cells,
- convergence ratio of the object function $\mathcal{O}_{\min}$, and
- minimum value of the object function $\mathcal{O}_{\min}$.

6.5.2 Reproduction of DCT scan

This section aims to generate 3D microstructures whose morphology matches a raster image obtained from a DCT scan. The raster image of the specimen used in this section was published in (193).

The method described in Section 6.3.1 is applied to three resolution levels of the DCT scan, shown in Figure 6.3. In what follows, the results of low, medium and high resolution raster images are discussed separately.

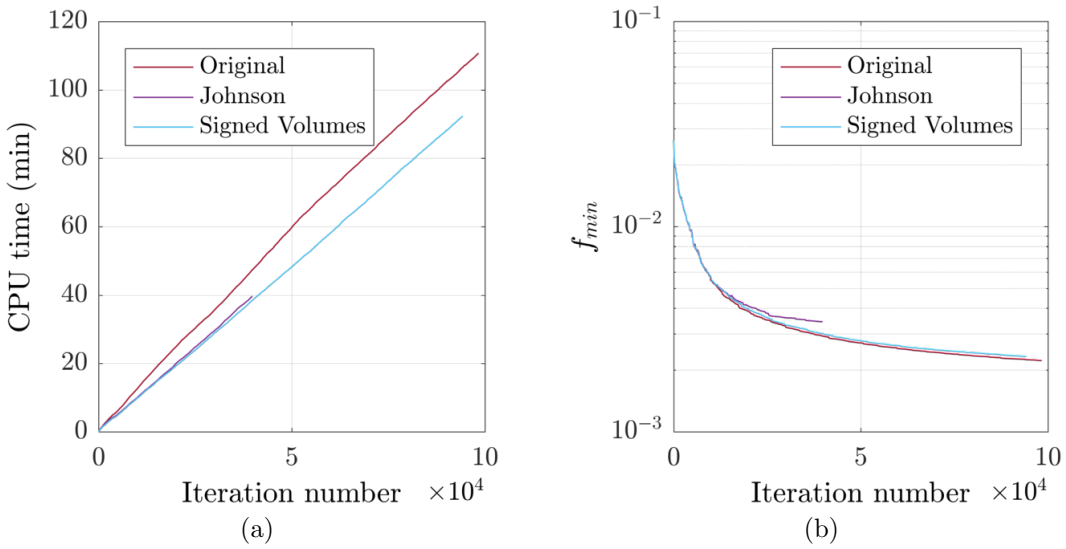


Figure 6.4: Comparison of CPU time (a) and convergence of target function (b) for low resolution DCT scans.

Low resolution Neper is used to generate a tessellation from the raster image in Figure 6.3(a). The convergence criterion is set to $\mathcal{O}^{i-1}_{\text{min}} - \mathcal{O}^{i}_{\text{min}} < 1 \times 10^{-5}$.

The history of CPU time and $\mathcal{O}_{\text{min}}$ are depicted in Figure 6.4 for the three GJK sub-algorithms. The overall CPU time of the procedure originally implemented in Neper is significantly higher compared to the other algorithms. Figure 6.4(a) shows that Johnson and the Signed Volumes algorithms have the same slope, i.e. the cost, but the latter terminates in about half iterations. This is because, as can be seen in Figure 6.4(b), the Johnson algorithm does not converge toward the minimum solution. The minimum value of $\mathcal{O}_{\text{min}}$ is reached by the Voronoi search algorithm, whilst the Signed Volumes leads to a slightly higher value.

Medium resolution A slightly higher resolution scan (Figure 6.3(b)) of the same specimen is considered. The convergence criterion is now set to 10^{-6} with the aim of analysing the influence over the overall convergence of different distance sub-algorithms.

The history of CPU time and $\mathcal{O}_{\text{min}}$ are depicted in Figure 6.5 for the three GJK

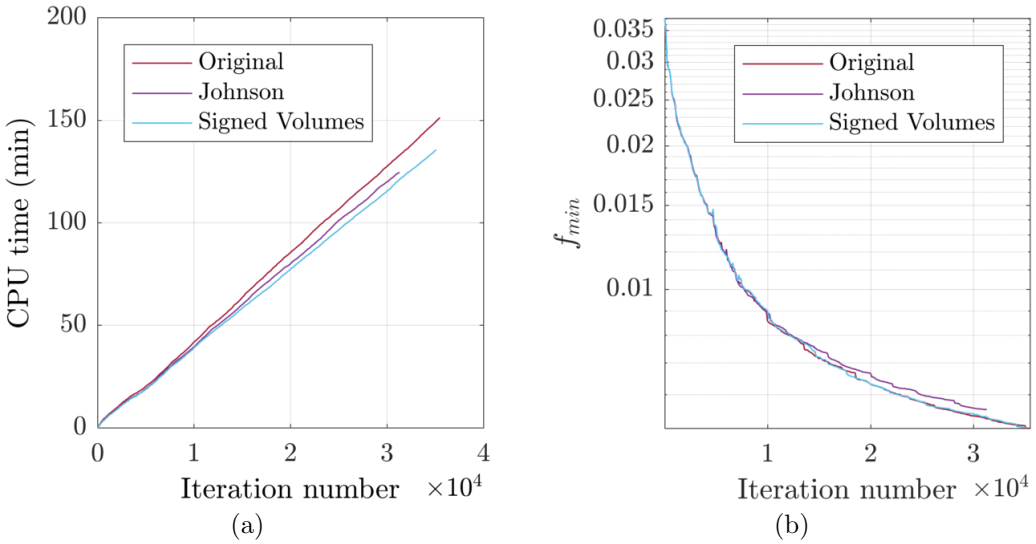


Figure 6.5: Comparison of CPU time (a) and convergence of target funtion (b) for medium resolution DCT scans.

Table 6.2: Results for different sub-algorithms for medium resolution DTC scan.

Method	Final $\mathcal{O}_{\min}$	Iterations	Elapsed time (min)
Voronoi search	5.102×10^{-3}	35536	151
Johnson algorithm	5.583×10^{-3}	31301	125
Signed Volumes	5.123×10^{-3}	35131	132

sub-algorithms. The overall CPU time, shown in Figure 6.5(a), reflects the trend of the previous experiment. The Signed Volumes algorithms requires about 10% less CPU time than the Voronoi Search, and the Johnson sub-algorithm terminates in remarkably fewer iterations.

The reduced tolerance on the convergence criterion seems to have a positive impact on the Johnson which however does not converge toward the minimum vale of $\mathcal{O}_{\min}$. Figure 6.5(b) also shows that the convergence of $\mathcal{O}_{\min}$ with the Johnson algorithm is rather irregular when compared to the other methods.

The graph in Figure 6.6 highlights that the Signed Volumes and the sub-algorithms originally implemented in Neper reach the same value of $\mathcal{O}_{\min}$ upon converge. These results are also summarised in Table 6.2.

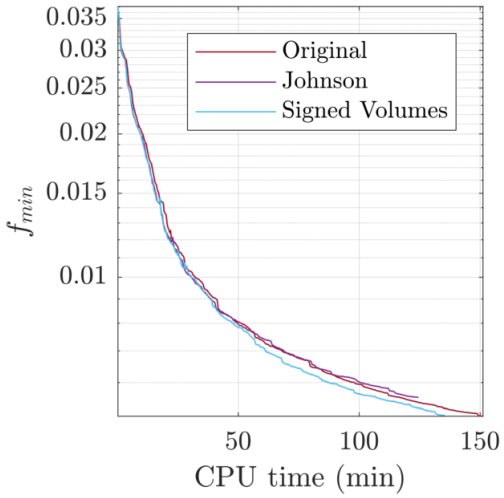


Figure 6.6: Comparison of object function convergence obtained with different distance algorithms for high resolution DTC scan.

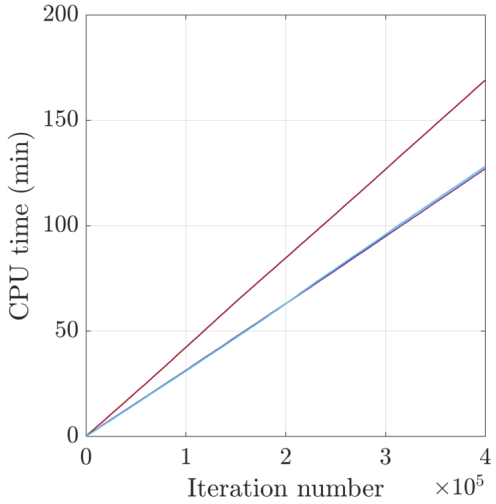


Figure 6.7: Comparison of CPU time of different distance algorithms for high resolution DCT scan.

High resolution The high-resolution DCT scan in Figure 6.3(c) is now used to generate a polycrystalline microstructure. Due to the large number of voxels, the optimisation process is carried out using 24 threads and iterations are limited to 44000.

Surprisingly, the optimisation did not converge when using the Johnson distance sub-algorithm. The cancellation error, which limits its accuracy, affects the optimisation from the very first iterations and the solution results unstable. Indeed, the resolution of this scan is not compatible with limited robustness of Johnson sub-algorithm.

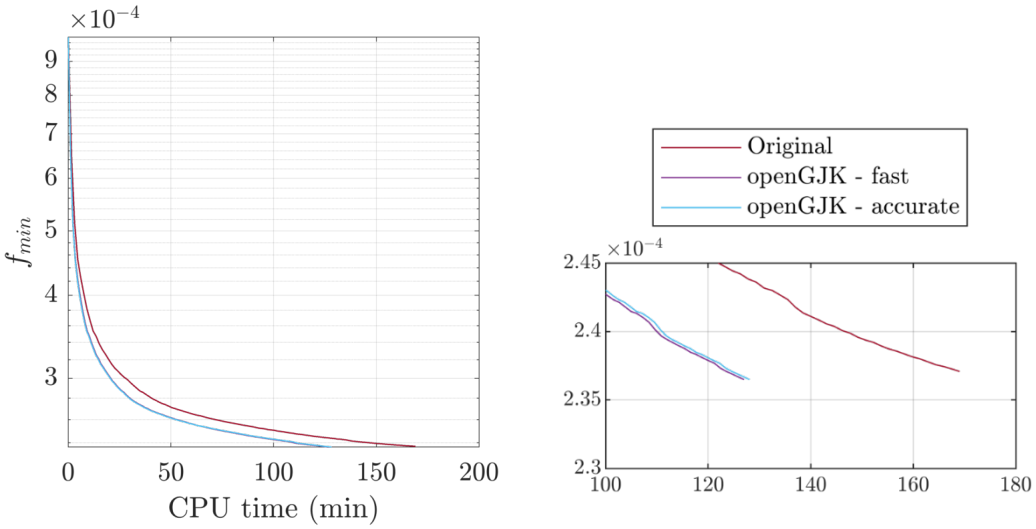


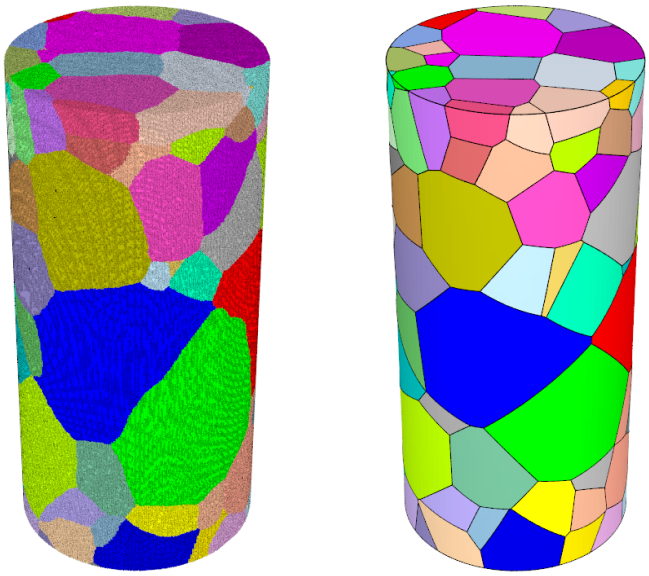
Figure 6.8: Comparison of object function convergence obtained with original method and Signed Volumes method for high resolution test.

The wall-clock time and history of $\mathcal{O}_{\text{min}}$ are depicted in Figure 6.7 and Figure 6.8, respectively. These reflect the trend found in the previous experiments, that is: when running the Signed Volumes algorithm Neper requires about 25% less CPU time than the Voronoi Search and both converge to a similar values of $\mathcal{O}_{\text{min}}$.

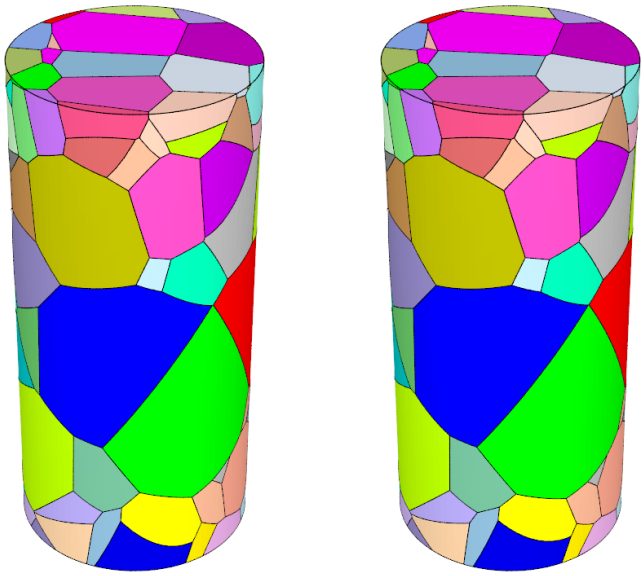
A comparison of the target microstructure, the initial tessellation and the result from the optimisation with Voronoi search and Signed Volumes is shown in Figure 6.9.

The graphs in Figure 6.10 shows the convergence of $\mathcal{O}_{\text{min}}$ against the wall-clock time. The curves are about 10%–15% apart; the original Neper implementation terminates at 6.38×10^{-5}, whilst the new GJK sub-algorithm leads to a small improvement as the smallest $\mathcal{O}_{\text{min}}$ value is 6.41×10^{-5}.

The smaller value of $\mathcal{O}_{\text{min}}$ achieved with the new Signed Volumes method has a moderate influence on the final tessellation and its regularisation, particularly on the frequency of small edges. Figure 6.11 reports the normalised count of the small edges before and after regularisation. The distribution is identical at the beginning



(a) (b)



(c) (d)

Figure 6.9: Optimisation of polycrystalline microstructure: (a) target DCT scan image, (b) initial guess, (c) Original GJK and (d) GJK with Signed Volumes.

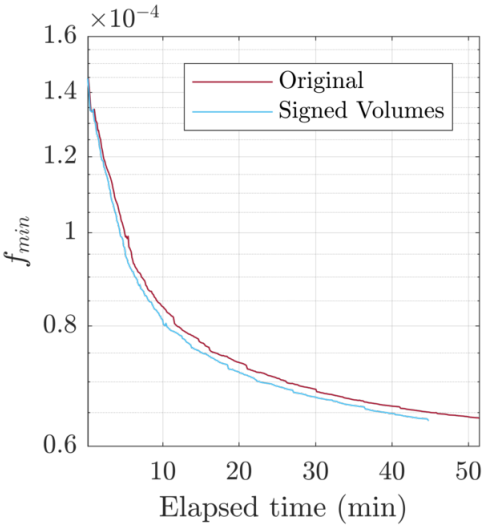


Figure 6.10: Objective function $\mathcal{O}_{min}$ versus CPU time.

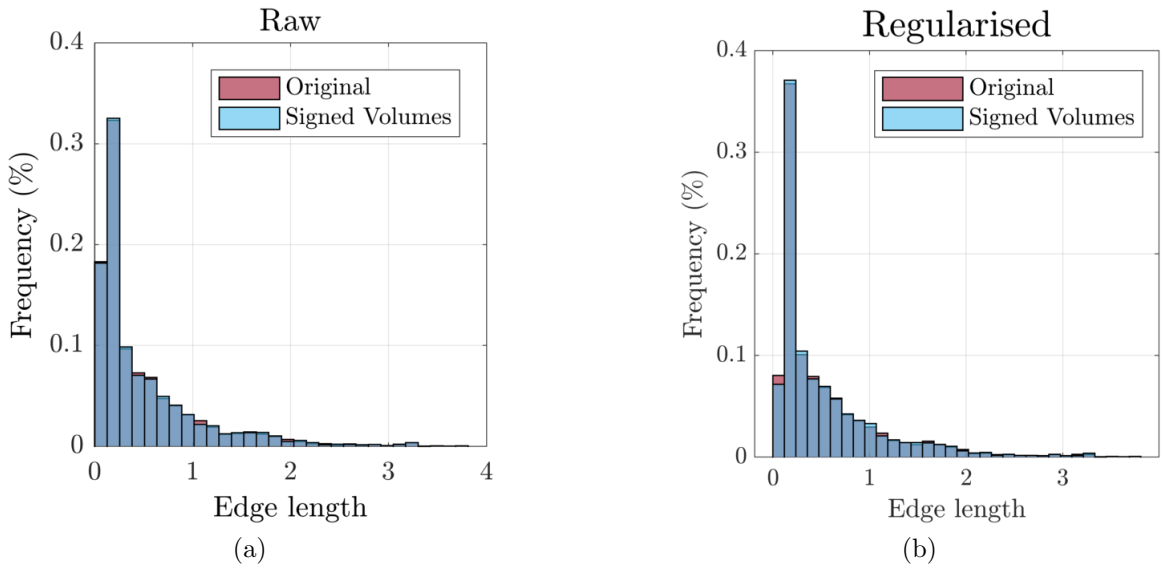
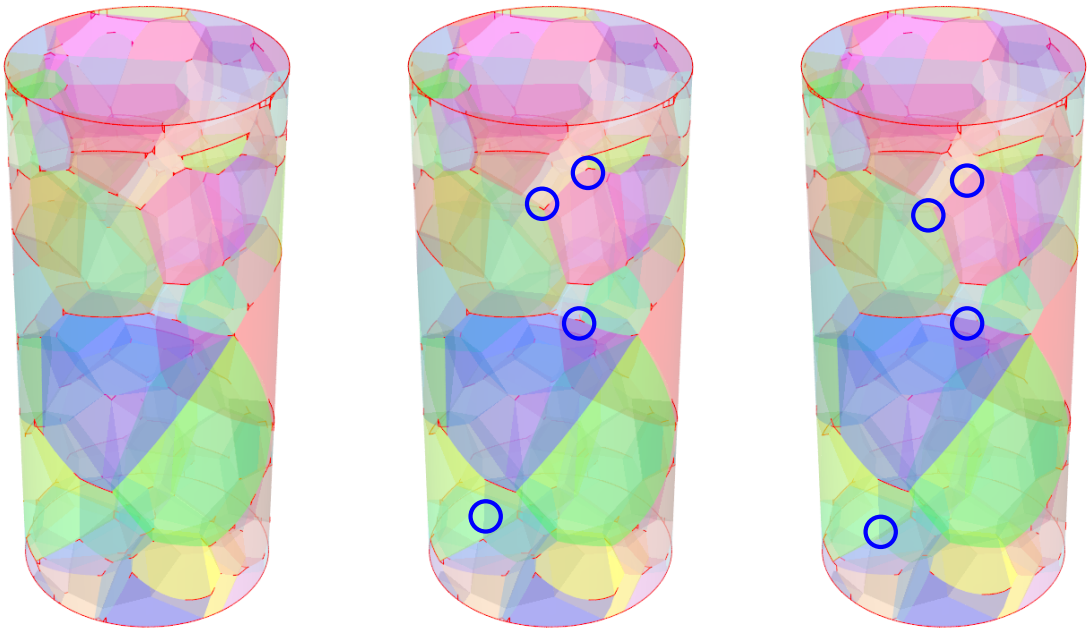


Figure 6.11: Comparison of edge length obtained with different distance algorithms high resolution.



(a) (b) (c)

Figure 6.12: Regularisation of a microstructure (a) using the original Neper implementation and (b) the Signed Volumes method (c).

(Figure 6.11(a)) and slightly improved when employing the Signed Volumes algorithm (Figure 6.11(b)). A close inspection on the regularised microstructures reveals that the former method overlooks some of the small edges, whilst the latter removes them, see Figure 6.12. This difference is a consequence of the different values of $\mathcal{O}_{\text{min}}$ achieved by the two implementations and the high accuracy of the Signed Volumes sub-algorithm.

6.6 Concluding remarks

In this chapter, the Signed Volumes distance sub-algorithm was applied to the generation of microstructures for multi-scale modelling of polycrystalline materials. The routines described in Chapter 3 were implemented into the open-source software Neper

and a performance comparison was carried out.

A specific microstructures obtained from DCT scan was reproduced by means of an optimisation process with Neper. Three levels of resolutions of the DCT scan were used. It has been found that the Johnson's sub-algorithm is not applicable to high-resolution DCT scans. Its limited accuracy leads to a poor convergence rate and, in one case, Neper could not converge because of the approximations introduced by the rounding error in the GJK sub-algorithm.

The Voronoi search and the Signed Volumes sub-algorithms are equally accurate, but the latter is remarkably faster. In all tests the elapsed time is reduced from 10% to 25% when GJK invokes the Signed Volumes sub-algorithm. For the high-resolution test case, this procedure has generated a regularised microstructure with fewer short edges, a result particularly desirable in large-strain and dynamic FE simulations.

Little coding effort was invested in implementing the new algorithms into Neper. The improvements did not require invasive changes to the existing software, as only the distance sub-algorithm has been replaced. Finally, it is claimed that this method is applicable to any material whose microstructure is described by Voronoi diagram or convex inclusions and voids, such as composite materials.

Chapter 7

Conclusions

This thesis has filled various gaps in the literature by developing novel distance algorithms for computational mechanics applications. These offer an improvement with respect to existing methodologies as they are: extremely versatile, more robust and designed to reuse cached data for reducing the computational cost. Three new algorithms were designed and implemented to tackle distance queries at three levels: primitives (elementary shapes), sub-body (semi-convex) and body (non-convex). These were extensively tested for all principal descriptions of solids employed in computational mechanics, namely: quadrics, polytopes and non-uniform rational B-splines (NURBS). The validation tests included an analytical calibration for contact problems, an analysis of the theoretical costs for the space-partitioning algorithm and elementary contact search tests. Finally, the algorithms were compared with existing methods for the solution of contact mechanics and optimisation problems. This chapter summarises the main findings and provides recommendations for future works.

7.1 Summary and main findings

7.1.1 The Signed Volumes method

None of the existing algorithms for the solution of point–simplex distance queries is robust and designed to exploit spatial coherence. The literature review has shown that robustness has been repeatedly overlooked by the engineering community, and in particular the impact that ill-conditioned problems have on the coupling of numerical methods (such as DEM and FEM).

The Signed Volume method has been formulated after an in-depth analysis of the effect of cancellation error. This provided insight on the mechanisms that led existing algorithms to fail and enables the formulation of well-posed distance queries. The result is a recursive procedure that handles naturally degenerate simplices and can be tailored to exploit spatial coherence.

The key element which makes this algorithm extremely robust and accurate to machine precision is the careful formulation of a well-posed problem. The sources of numerical instabilities that could not be avoided have been placed where they cannot compromise the final solution. This enhanced the robustness of the Signed Volumes method.

7.1.2 Improving the GJK algorithm

The GJK algorithm could not be employed in engineering simulations because of its lack of accuracy. Several authors pointed out that the the original distance sub-algorithm, due to Johnson, is the main source of numerical instability. However, the studies published to date did not provide sufficient information to address these instabilities.

All attempts have focused on the termination conditions of the GJK algorithm rather than the distance sub-algorithm.

A significant effort has been spent in Section 2.2 to shed light on the mechanisms that lead the GJK algorithm to failure. It has been found that instabilities are not caused by degenerate simplices, but rather by the volume form of the Voronoi region supporting the point of minimum norm. This thesis has shown that the GJK algorithm fails when the volume form of this region is null, however, degenerate simplices cannot be avoided since thye are are somehow intrinsic to the GJK algorithm.

A solution to improve the GJK algorithm was found by replacing the original sub-algorithm with the new Signed Volumes method. Its higher accuracy improved the whole convergence of the GJK algorithm for both degenerate and well-conditioned simplices. This translated into faster and more accurate distance queries between arbitrary convex bodies, including polytopes, quadrics, NURBS or compounds of these.

Numerical tests demonstrated that this effectively is a more robust procedure. In particular, when the objects are found in contact, the newly proposed sub-algorithm runs from 15% to 30% faster than the original one.

7.1.3 Novel hierarchical search

In order to reduce the costs of the broad search for distance queries between non-convex bodies, a new methodology was introduced in Section 3.3. Unlike other procedures, only intrinsic quantities govern the broad search of this new method.

The procedure is designed for problems in which bodies have a complex shapes that cannot be represented by simple polygons or quadrics. Basically, no assumption was made on the shape of the bodies as these may displace, deform and change morphology.

The only assumption made is that a large number of small binary trees may be built and traversed more quickly than a unique tree. This hypothesis was then verified by means fo a theoretical and numerical studies in Chapters 4 and 5.

7.1.4 Verification tests

A large number of verification tests were presented in Chapter 4. The tests have proven that:

- Adaptive floating-point arithmetic can reduce the number of operations of the Signed Volumes method and improve its accuracy.
- The routines implemented for the contact detection of FEM (tetrahedral and hexahedral elements) and DEM simulations are robust.
- The contact resolution using the penalty formulation is robust and was calibrated analytically for frictionless contact.
- The logic of the hierarchical search is sound.
- The separating distance can be safely approximated by the total displacement of a body and this reduces by 95% the number of calls to the GJK algorithm.
- Contact is correctly detected even for large displacement problems involving sliding over a body decomposed into sub-bodies.
- A gain in terms of CPU time is expected only if less than three bodies are mutually in contact, for five or more an exponential loss is expected.
- The hypothesis formulated in Chapter 3 was validated.
- NURBS curves and surfaces may be treated by the new algorithms even at the broad search phase.

Finally, an extensive comparison with published methods have been presented. For

the first time the Voronoi search (64) and the Johnson's algorithm (89) have been compared. These were also compared with the Signed Volumes method which resulted either faster or slightly slower than Johnson's algorithm. Two versions (incremental and non-incremental) of the GJK procedure were also tested with different distance sub-algorithm. These have shown that the Signed Volumes method is indeed beneficial for the convergence of the GJK procedure which ran 15%-20% faster.

7.1.5 Application to contact mechanics

The first application presented in this thesis aims to improve the modelling of granular media. Chapter 5 has demonstrated the ability of the novel numerical tools to simulate the mechanical packing of sand.

The reduction of CPU time obtained with the new hierarchical framework allows to simulate grains with realistic shapes. Existing procedures would also be able to run the same simulation, but the new algorithms dramatically reduce the computational time for the detection of contact.

Three mesh refinements were tested to show that indeed the final assembly depends on the mesh size. The coordination number and void ratio obtained with different solvers were compared. Good agreement was found for samples of 100, 1200 and 2400 sand grains as in all these cases the CPU time saved was always above 50%.

Finally, the underlying hypothesis made in Chapter 3 about the number of bodies mutually in contact was validated numerically. For the largest sample, about 80% of the bodies collided with less than four other grains, thus justifying the reduction of operations for the contact search.

Whilst this application involved only the combination of FEM and DEM, the new

algorithms allow for the coupling of more numerical methods — such as meshless methods or isogeometric analysis (IGA). For all of these, one could solve distance queries between solid, shell, truss elements as well as rigid particles, spheres, ellipsoids and NURBS.

7.1.6 Application to morphology optimisation

Chapter 6presented a case study in which the improved GJK algorithm is used to advance the modelling of heterogeneous materials. This involved the generation of representative volume elements (RVEs) for the multi-scale analysis of polycrystalline materials. The generation process does not admit a closed-form solution and it therefore needs an optimisation process in which distance queries play a pivotal role.

The improved GJK algorithm was implemented in NEPER, a third-party software package for the generation of RVEs. Two procedures were explored. The first one aimed at adjusting the morphological properties of the RVE to reflect a given statistical distribution, and the second one to match a particular experimental morphology. In both cases the GJK algorithm is extensively used to drive the optimisation process.

Both generation processes were improved in terms of CPU time and quality. The microstructures can now be generated 25% faster, and the regularisation step is now performed more accurately.

7.2 Recommendations for further research

This section presents some of the future works which would build on top of what has been developed and verified in this thesis. A trivial novelty would be to apply the

novel algorithms to fields other than engineering and computer graphics, but there are a number of (challenging) contributions from which these communities could benefit extensively. Below are mentioned the three research directions the author values the most.

7.2.1 Continuous contact detection in engineering

The concept of *continuous contact detection* has been developed for computer graphics applications (199) to accelerate the contact search in dynamic simulations. This method, however, is currently not available to engineers.

The idea is to search for collision between two time-steps by using information from the past and (possibly estimated) from the future. This has been shown to speed up the broad search phase and it suits all representations of solid bodies.

The way continuous contact detection is currently exploited makes use of the GJK algorithm (36, 225), for which reason this thesis represents a step forward. However, the high level of details and accuracy which characterise engineering simulations are barriers not tackled by the computer graphics community. Making a coarse approximation that enables to estimate collision and the subsequent time-step is probably the biggest challenge to tackle before continuous contact detection can be applied in engineering.

7.2.2 Extension to large NURBS models

This thesis has shown how the new algorithmic framework can solve distance queries between NURBS curves, surfaces and solids, however, engineering simulations usually involve larger bodies than the one considered here. These could bring new challenges to

the solution of distance queries for methods requiring an initial guess, as in Chapter 4.

Furthermore, the fact that in this thesis NURBS were not approximated by tiles, or other primitives, has been only used for contact detection purposes. From an engineering point of view, one should make use of the accurate representation offered by NURBS also for the resolution of contact constraints. That is, the phase following the collision detection search.

An extension to the work presented in this thesis would therefore be the design of contact resolution algorithms for isogeometric analysis (IGA) and similar NURBS-based methods. Since the algorithms currently available in the literature do not preserve the accurate representation offered by NURBS, but this thesis fills that gap, future works can rely on the new algorithms to improve the fidelity of computational contact mechanical models.

7.2.3 Heterogeneous computing on distributed-memory systems

Initially, the algorithms developed in this thesis were implemented from scratch, and therefore on a serial code. Lately, they were made multi-thread safe and implemented in a parallel code which, however, is limited to CPU and shared-memory architectures.

Future works should assess the performance of the new hierarchical framework on distributed-memory systems and make use of processors units other than CPU. The advances of programming frameworks for heterogeneous computing, such as openACC and openCL, are enabling engineers to run simulations significantly faster. The new distance algorithms should therefore make the most out of the emerging general-purpose computing on graphics processing units (GPGPU).

Bibliography

[1] Al-Bluwi, I., Siméon, T., and Cortés, J. (2012). Motion planning algorithms for molecular simulations: A survey. *Computer Science Review*, 6(4):125–143.

[2] Alonso-Marroquin, F. and Herrmann, H. J. (2002). Calculation of the incremental stress-strain relation of a polygonal packing. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, 66(2 Pt 1):021301.

[3] Andrade, J. E., Avila, C. F., Hall, S. A., Lenoir, N., and Viggiani, G. (2011). Multiscale modeling and characterization of granular matter: From grain kinematics to continuum mechanics. *Journal of the Mechanics and Physics of Solids*, 59(2):237–250.

[4] Andrade, J. E., Lim, K.-W., Avila, C. F., and Vlahinić, I. (2012). Granular element method for computational particle mechanics. *Computer Methods in Applied Mechanics and Engineering*, 241-244:262–274.

[5] André, D., Jebahi, M., Iordanoff, I., Charles, J.-l., and Néauport, J. (2013). Using the discrete element method to simulate brittle fracture in the indentation of a silica glass with a blunt indenter. *Computer Methods in Applied Mechanics and Engineering*, 265(Supplement C):136–147.

[6] Andreotti, B., Forterre, Y., and Pouliquen, O. (2013). *Granular Media: Between Fluid and Solid*. Cambridge University Press, Cambridge.

[7] Bagi, K. (2005). An algorithm to generate random dense arrangements for discrete element simulations of granular assemblies. *Granular Matter*, 7(1):31–43.

[8] Barbe, F., Forest, S., and Cailletaud, G. (2001). Intergranular and intragranular behavior of polycrystalline aggregates.Part 2: Results. *International Journal of Plasticity*, 17(4):537–563.

[9] Barbe, F. and Quey, R. (2011). A numerical modelling of 3D polycrystal-to-polycrystal diffusive phase transformations involving crystal plasticity. *International Journal of Plasticity*, 27(6):823–840.

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

[11] Bates, L. (2006). The need for industrial education in bulk technology. *Bulk Solids Handling*, 26(7):464–474.

[12] Batra, R. C. and Zhang, G. M. (2007). Search algorithm, and simulation of elastodynamic crack propagation by modified smoothed particle hydrodynamics (MSPH) method. *Computational Mechanics*, 40(3):531–546.

[13] Bellan, S., Matsubara, K., Cho, H. S., Gokon, N., and Kodama, T. (2018). A CFD-DEM study of hydrodynamics with heat transfer in a gas-solid fluidized bed reactor for solar thermal applications. *International Journal of Heat and Mass Transfer*, 116(Supplement C):377–392.

[14] Belytschko, T. and Lin, J. I. (1987). A three-dimensional impact-penetration algorithm with erosion. *International Journal of Impact Engineering*, 5(1–4):111–127.

[15] Belytschko, T., Lu, Y. Y., and Gu, L. (1994). Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256.

[16] Bendsoe, M. P. and Sigmund, O. (2003). *Topology Optimization: Theory, Methods, and Applications*. Springer Science & Business Media.

[17] Benedetti, I. and Aliabadi, M. H. (2013). A three-dimensional cohesive-frictional grain-boundary micromechanical model for intergranular degradation and failure in polycrystalline materials. *Computer Methods in Applied Mechanics and Engineering*, 265:36–62.

[18] Benson, D. J. (1992). Computational methods in Lagrangian and Eulerian hydrocodes. *Computer Methods in Applied Mechanics and Engineering*, 99(2-3):235–394.

[19] Benson, D. J. (1997). A mixture theory for contact in multi-material Eulerian formulations. *Computer Methods in Applied Mechanics and Engineering*, 140(1-2):59–86.

[20] Benson, D. J., Bazilevs, Y., De Luycker, E., Hsu, M.-C., Scott, M., Hughes, T. J. R., and Belytschko, T. (2010). A generalized finite element formulation for arbitrary basis functions: From isogeometric analysis to XFEM. *International Journal for Numerical Methods in Engineering*, 83(6):765–785.

[21] Benson, D. J. and Hallquist, J. O. (1990). A single surface contact algorithm for the post-buckling analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering*, 78(2):141–163.

[22] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commununications of the ACM*, 18(9):509–517.

[23] Bentley, J. L. and Friedman, J. H. (1979). Data structures for range searching. *ACM Computing Surveys*, 11(4):397–409.

[24] Boeing, A. and Bräunl, T. (2007). Evaluation of real-time physics simulation systems. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 281–288, New York, NY, USA. ACM.

[25] Bonet, J. and Peraire, J. (1991). An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31(1):1–17.

[26] Boon, C., Houlsby, G., and Utili, S. (2012). A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method. *Computers and Geotechnics*, 44:73–82.

[27] Boon, C., Houlsby, G., and Utili, S. (2013). A new contact detection algorithm for three-dimensional non-spherical particles. *Powder Technology*, 248:94–102.

[28] Boris, J. (1986). A vectorized "near neighbors" algorithm of order N using a monotonic logical grid. *Journal of Computational Physics*, 66(1):1–20.

[29] Bowman, E. T., Soga, K., and Drummond, W. (2001). Particle shape characterisation using Fourier descriptor analysis. *Géotechnique*, 51(6):545–554.

[30] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, UK.

[31] Boyse, J. W. (1979). Interference detection among solids and surfaces. *Commununications of the ACM*, 22(1):3–9.

[32] Brodu, N., Richard, P., and Delannay, R. (2013). Shallow granular flows down flat frictional channels: Steady flows and longitudinal vortices. *Physical Review E*, 87(2):022202.

[33] Brooks, R. A. and Lozano-Pérez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(2):224–233.

[34] Bruneel, H. C. J. and De Rycke, I. (2002). QuickTrace: A fast algorithm to detect contact. *International Journal for Numerical Methods in Engineering*, 54(2):299–316.

[35] Cameron, S. (1985). A study of the clash detection problem in robotics. In *1985 IEEE International Conference on Robotics and Automation Proceedings*, volume 2, pages 488–493.

[36] Cameron, S. (1990). Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automation*, 6(3):291–302.

[37] Cameron, S. (1997a). A comparison of two fast algorithms for computing the distance between convex polyhedra. *IEEE Transactions on Robotics and Automation*, 13(6):915–920.

[38] Cameron, S. (1997b). Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In *In Proc. of IEEE International Conference on Robotics and Automation*, volume 4, pages 3112–3117 vol.4.

[39] Cameron, S. and Culley, R. (1986). Determining the minimum translational distance between two convex polyhedra. In *IEEE International Conference on Robotics and Automation*, pages 591–596, San Francisco, CA, USA.

[40] Catalanotti, G. (2016). On the generation of RVE-based models of composites reinforced with long fibres or spherical particles. *Composite Structures*, 138:84–95.

[41] Chang, S.-W. and Chen, C.-S. (2008). A non-iterative derivation of the common plane for contact detection of polyhedral blocks. *International Journal for Numerical Methods in Engineering*, 74(5):734–753.

[42] Chen, H., Lei, Z., and Zang, M. (2014). LC-Grid: A linear global contact search algorithm for finite element analysis. *Computational Mechanics*, 54(5):1285–1301.

[43] Chen, H., Zhang, Y. X., Zang, M., and Hazell, P. J. (2015). An accurate and robust contact detection algorithm for particle-solid interaction in combined finite-discrete element analysis. *International Journal for Numerical Methods in Engineering*, 103(8):598–624.

[44] Chinesta, F., Ladeveze, P., and Cueto, E. (2011). A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404.

[45] Chung, K. and Wang, W. (1996). Quick collision detection of polytopes in virtual environments. pages 125–131.

[46] Cohen, J. D., Lin, M. C., Manocha, D., and Ponamgi, M. (1995). I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, I3D '95, pages 189–ff., New York, NY, USA. ACM.

[47] Comba, P. G. (1968). A procedure for detecting intersections of three-dimensional objects. *Journal of the Association for Computing Machinery*, 15(3):354–366.

[48] Coxeter, H. S. M. (1989). *Introduction to Geometry*. Wiley.

[49] Cristiani, E., Piccoli, B., and Tosin, A. (2011). Multiscale modeling of granular flows with application to crowd dynamics. *Multiscale Modeling & Simulation*, 9(1):155–182.

[50] Cundall, P. A. (1988). 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. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics*, 25(3):107–116.

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

[52] Dax, A. (2006). The distance between two convex sets. *Linear Algebra and its Applications*, 416(1):184–213.

[53] de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry*. Springer, Berlin, Heidelberg.

[54] de Boor, C. (1972). On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62.

[55] De Cola, F., Falco, S., Barbieri, E., and Petrinic, N. (2015). New 3D geometrical deposition methods for efficient packing of spheres based on tangency. *International Journal for Numerical Methods in Engineering*, 104(12):1085–1114.

[56] De Cola, F., Pellegrino, A., Glößner, C., Penumadu, D., and Petrinic, N. (2017). Effect of particle morphology, compaction, and confinement on the high strain rate behavior of sand. *Experimental Mechanics*, pages 1–20.

[57] Diard, O., Leclercq, S., Rousselier, G., and Cailletaud, G. (2005). Evaluation of finite element based analysis of 3D multicrystalline aggregates plasticity: Application to crystal plasticity model identification and the study of stress and strain fields near grain boundaries. *International Journal of Plasticity*, 21(4):691–722.

[58] Dietrich, A., Wimbock, T., Albu-Schaffer, A., and Hirzinger, G. (2012). Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. *IEEE Robotics & Automation Magazine*, 19(2):20–33.

[59] Dolbow, J. and Belytschko, T. (1999). Numerical integration of the Galerkin weak form in meshfree methods. *Computational Mechanics*, 23(3):219–230.

[60] Eberly, D. H. (2000). *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. CRC Press, San Francisco, har/cdr edition edition.

[61] Edelsbrunner, H. and Mücke, E. P. (1990). Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104.

[62] Ehmann, S. A. and Lin, M. C. (2001). Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum*, 20(3):500–511.

[63] Eliáš, J. (2014). Simulation of railway ballast using crushable polyhedral particles. *Powder Technology*, 264:458–465.

[64] Ericson, C. (2004). *Real-Time Collision Detection*. Morgan Kaufmann.

[65] Evans, T. M. and Frost, J. D. (2010). Multiscale investigation of shear bands in sand: Physical and numerical experiments. *International Journal for Numerical and Analytical Methods in Geomechanics*, 34(15):1634–1650.

[66] Falco, S., Jiang, J., De Cola, F., and Petrinic, N. (2017). Generation of 3D polycrystalline microstructures with a conditioned Laguerre-Voronoi tessellation technique. *Computational Materials Science*, 136:20–28.

[67] Fang, Z. Q., Hu, G. M., Du, J., Fan, Z., and Liu, J. (2015). A contact detection algorithm for multi-sphere particles by means of two-level-grid-searching in DEM simulations. *International Journal for Numerical Methods in Engineering*, 102(13):1869–1893.

[68] Feng, Y. T., Han, K., and Owen, D. R. J. (2003). Filling domains with disks: An advancing front approach. *International Journal for Numerical Methods in Engineering*, 56(5):699–713.

[69] Feng, Y. T., Han, K., and Owen, D. R. J. (2007). Coupled lattice Boltzmann method and discrete element modelling of particle transport in turbulent fluid flows: Computational issues. *International Journal for Numerical Methods in Engineering*, 72(9):1111–1134.

[70] Feng, Y. T., Han, K., and Owen, D. R. J. (2017). A generic contact detection framework for cylindrical particles in discrete element modelling. *Computer Methods in Applied Mechanics and Engineering*, 315:632–651.

[71] Feng, Y. T. and Owen, D. R. J. (2002). An augmented spatial digital tree algorithm for contact detection in computational mechanics. *International Journal for Numerical Methods in Engineering*, 55(2):159–176.

[72] Fernández, A., Jérusalem, A., Gutiérrez-Urrutia, I., and Pérez-Prado, M. T. (2013). Three-dimensional investigation of grain boundary–twin interactions in a Mg AZ31 alloy by electron backscatter diffraction and continuum modeling. *Acta Materialia*, 61(20):7679–7692.

[73] Fischer, K. and Gartner, B. (2003). The smallest enclosing ball of balls: Combinatorial structure and algorithms. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pages 292–301, New York, NY, USA. ACM.

[74] Fleischmann J. A., Plesha M. E., and Drugan W. J. (2013). Quantitative comparison of two-dimensional and three-dimensional discrete-element simulations of nominally two-dimensional shear flow. *International Journal of Geomechanics*, 13(3):205–212.

[75] Fortune, S. and Van Wyk, C. J. (1993). Efficient exact arithmetic for computational geometry. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, SCG '93, pages 163–172, New York, NY, USA. ACM.

[76] Fortune, S. and Van Wyk, C. J. (1996). Static analysis yields efficient exact integer arithmetic for computational geometry. *ACM Transactions on Graphics*, 15(3):223–248.

[77] Frey, P. J. and George, P.-L. (2010). *Mesh Generation: Application to Finite Elements*. Mesh Generation: Application to Finite Elements: Second Edition. Wiley.

[78] Fritzen, F., Böhlke, T., and Schnack, E. (2009). Periodic three-dimensional mesh generation for crystalline aggregates based on Voronoi tessellations. *Computational Mechanics*, 43(5):701–713.

[79] Fu, R., Hu, X., and Zhou, B. (2017). Discrete element modeling of crushable sands considering realistic particle shape effect. *Computers and Geotechnics*, 91(Supplement C):179–191.

[80] Fujimura, K. and Samet, Sr., H. (1989). A hierarchical strategy for path planning among moving obstacles. *IEEE Transactions on Robotics and Automation*, 5(1):61–69.

[81] Galindo-Torres, S. A., Scheuermann, A., Mühlhaus, H. B., and Williams, D. J. (2015). A micro-mechanical approach for the study of contact erosion. *Acta Geotechnica*, 10(3):357–368.

[82] Gallier, J. (2011). *Geometric Methods and Applications*, volume 38 of *Texts in Applied Mathematics*. Springer New York, New York, NY.

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

[84] Gärtner, B. (1999). Fast and robust smallest enclosing balls. In *Algorithms - ESA' 99*, Lecture Notes in Computer Science, pages 325–338, Berlin, Heidelberg. Springer.

[85] Geijtenbeek, T. and Pronost, N. (2012). Interactive character animation using simulated physics: A state-of-the-art review. *Computer Graphics Forum*, 31(8):2492–2515.

[86] Ghosh, M., Amato, N. M., Lu, Y., and Lien, J.-M. (2013). Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2):494–504.

[87] Gilardi, G. and Sharf, I. (2002). Literature survey of contact dynamics modelling. *Mechanism and Machine Theory*, 37(10):1213–1239.

[88] Gilbert, E. and Foo, C.-P. (1990). Computing the distance between general convex objects in three-dimensional space. *IEEE Transactions on Robotics and Automation*, 6(1):53–61.

[89] Gilbert, E., Johnson, D., and Keerthi, S. (1988). A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203.

[90] Gupta, R. and Gao, J. (2006). United States Patent: 7030875 - Environmental reasoning using geometric data structure.

[91] Haji-Akbari, A., Chen, E. R., Engel, M., and Glotzer, S. C. (2013). Packing and self-assembly of truncated triangular bipyramids. *Physical Review E*, 88(1):012127.

[92] Han, K., Feng, Y. T., and Owen, D. R. J. (2005). Sphere packing with a geometric based compression algorithm. *Powder Technology*, 155(1):33–41.

[93] Han, K., Feng, Y. T., and Owen, D. R. J. (2007). Performance comparisons of tree-based and cell-based contact detection algorithms. *International Journal for Computer-Aided Engineering and Software*, 24(2):165–181.

[94] Hartmann, E. (1999). On the curvature of curves and surfaces defined by normalforms. *Computer Aided Geometric Design*, 16(5):355–376.

[95] Hattori, G., Trevelyan, J., Augarde, C. E., Coombs, W. M., and Aplin, A. C. (2017). Numerical simulation of fracking in shale rocks: Current state and future approaches. *Archives of Computational Methods in Engineering*, 24(2):281–317.

[96] Hoffmann, C. M. (2001). Robustness in geometric computations. *Journal of Computing and Information Science in Engineering*, 1(2):143–155.

[97] Hornus, S. (2017). Detecting the intersection of two convex shapes by searching on the 2-sphere. *Computer-Aided Design*, 90(Supplement C):71–83.

[98] Houlsby, G. T. (2009). Potential particles: A method for modelling non-circular particles in DEM. *Computers and Geotechnics*, 36(6):953–959.

[99] Houlsby, G. T. and Cassidy, M. J. (2002). A plasticity model for the behaviour of footings on sand under combined loading. *Géotechnique*, 52(2):117–129.

[100] Hu, L., Hu, G., Fang, Z., and Zhang, Y. (2013). A new algorithm for contact detection between spherical particle and triangulated mesh boundary in discrete element method simulations. *International Journal for Numerical Methods in Engineering*, 94(8):787–804.

[101] Hughes, T., Cottrell, J., and Bazilevs, Y. (2005). Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39–41):4135–4195.

[102] Ilin, D. N. and Bernacki, M. (2016). Advancing layer algorithm of dense ellipse packing for generating statistically equivalent polygonal structures. *Granular Matter*, 18(3):43.

[103] Jerves, A. X., Kawamoto, R. Y., and Andrade, J. E. (2016). Effects of grain morphology on critical state: A computational analysis. *Acta Geotechnica*, 11(3):493–503.

[104] Ji, H., Lien, F.-S., and Yee, E. (2008). A robust and efficient hybrid cut-cell/ghost-cell method with adaptive mesh refinement for moving boundaries on irregular domains. *Computer Methods in Applied Mechanics and Engineering*, 198(3):432–448.

[105] Jiménez, P., Thomas, F., and Torras, C. (2000). 3D Collision detection: A survey. *Computers and Graphics*, 25:269–285.

[106] Johnson, D. W. (1987). *The Optimization of Robot Motion in the Presence of Obstacles*. PhD thesis, University of Michigan, Ann Arbor, USA. AAI8720285.

[107] Kabir, M. R. and Richter, H. (2017). Modeling of processing-induced pore morphology in an additively-manufactured Ti-6Al-4V alloy. *Materials*, 10(2):145.

[108] Kanaun, S. and Tkachenko, O. (2006). Mechanical properties of open cell foams: Simulations by Laguerre tesselation procedure. *International Journal of Fracture*, 140(1-4):305–312.

[109] Kanit, T., Forest, S., Galliet, I., Mounoury, V., and Jeulin, D. (2003). Determination of the size of the representative volume element for random composites: Statistical and numerical approach. *International Journal of Solids and Structures*, 40(13–14):3647–3679.

[110] Kawamoto, R., Andò, E., Viggiani, G., and Andrade, J. E. (2016). Level set discrete element method for three-dimensional computations with triaxial case study. *Journal of the Mechanics and Physics of Solids*, 91(Supplement C):1–13.

[111] Kikuchi, Y. (2006). Investigation of engineering properties of man-made composite geo-materials with micro-focus x-ray CT. In Desrues, J., Viggiani, G., and Bésuelle, P., editors, *Advances in X-Ray Tomography for Geomaterials*, pages 53–78. ISTE.

[112] Ko, K. and Sakkalis, T. (2014). Orthogonal projection of points in CAD/-CAM applications: An overview. *Journal of Computational Design and Engineering*, 1(2):116–127.

[113] Kopačka, J., Gabriel, D., Plešek, J., and Ulbin, M. (2016). Assessment of methods for computing the closest point projection, penetration, and gap functions in contact searching problems. *International Journal for Numerical Methods in Engineering*, 105(11):803–833.

[114] Kremmer, M. and Favier, J. F. (2001). A method for representing boundaries in discrete element modelling—part I: Geometry and contact detection. *International Journal for Numerical Methods in Engineering*, 51(12):1407–1421.

[115] Krill III, C. E. and Chen, L. Q. (2002). Computer simulation of 3-D grain growth using a phase-field model. *Acta Materialia*, 50(12):3059–3075.

[116] Krishnamurthy, A., McMains, S., and Haller, K. (2011). GPU-accelerated minimum distance and clearance queries. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):729–742.

[117] Krstulović-Opara, L., Wriggers, P., and Korelc, J. (2002). A C1-continuous formulation for 3D finite deformation frictional contact. *Computational Mechanics*, 29(1):27–42.

[118] Ladevèze, P., Passieux, J.-C., and Néron, D. (2010). The LATIN multiscale computational method and the Proper Generalized Decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21–22):1287–1296.

[119] Lagarias, J. C., Mallows, C. L., and Wilks, A. R. (2001). Beyond the Descartes circle theorem. *American Mathematical Monthly*, 109(4):338–361.

[120] Latham, J.-P. and Munjiza, A. (2004). The modeling of particle systems with real shapes. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 362:1953–72.

[121] Laursen, T. A. and Simo, J. C. (1993). A continuum-based finite element formulation for the implicit solution of multibody, large deformation-frictional contact problems. *International Journal for Numerical Methods in Engineering*, 36(20):3451–3485.

[122] Laycock, S. and Day, A. (2007). A survey of haptic rendering techniques. *Computer Graphics Forum*, 26(1):50–65.

[123] Lebensohn, R. A., Montagnat, M., Mansuy, P., Duval, P., Meysonnier, J., and Philip, A. (2009). Modeling viscoplastic behavior and heterogeneous intracrystalline deformation of columnar ice polycrystals. *Acta Materialia*, 57(5):1405–1415.

[124] Lee, C.-H. (2013). United States Patent: 8497875 - System, method, and computer program product for determining a translation vector.

[125] Lewis, R. W., Gethin, D. T., Yang, X. S., and Rowe, R. C. (2005). A combined finite-discrete element method for simulating pharmaceutical powder tableting. *International Journal for Numerical Methods in Engineering*, 62(7):853–869.

[126] Lhoutellier, G., Ledue, D., Patte, R., and Baltz, V. (2016). Monte Carlo investigation of how interfacial magnetic couplings affect blocking temperature distributions in exchange bias bilayers. *Journal of Applied Physics*, 120(19):193902.

[127] Li, C. F., Feng, Y. T., and Owen, D. R. J. (2006). SMB: Collision detection based on temporal coherence. *Computer Methods in Applied Mechanics and Engineering*, 195(19):2252–2269.

[128] Li, S., Qian, D., Kam Liu, W., and Belytschko, T. (2001). A meshfree contact-detection algorithm. *Computer Methods in Applied Mechanics and Engineering*, 190(24–25):3271–3292.

[129] Lim, K.-W. and Andrade, J. E. (2014). Granular element method for three-dimensional discrete element calculations. *International Journal for Numerical and Analytical Methods in Geomechanics*, 38(2):167–188.

[130] Lim, K.-W., Krabbenhoft, K., and Andrade, J. E. (2014a). A contact dynamics approach to the Granular Element Method. *Computer Methods in Applied Mechanics and Engineering*, 268(Supplement C):557–573.

[131] Lim, K.-W., Krabbenhoft, K., and Andrade, J. E. (2014b). On the contact treatment of non-convex particles in the granular element method. *Computational Particle Mechanics*, 1(3):257–275.

[132] Lin, M. and Canny, J. (1991). A fast algorithm for incremental distance calculation. In *Proceedings of IEEE International Conference on Robotics and Automation, 1991. Proceedings*, volume 2, pages 1008–1014.

[133] Lintermann, A., Schlimpert, S., Grimmen, J., Günther, C., Meinke, M., and Schröder, W. (2014). Massively parallel grid generation on HPC systems. *Computer Methods in Applied Mechanics and Engineering*, 277:131–153.

[134] Lisjak, A. and Grasselli, G. (2014). A review of discrete modeling techniques for fracturing processes in discontinuous rock masses. *Journal of Rock Mechanics and Geotechnical Engineering*, 6(4):301–314.

[135] Liu, A., Tendick, F., Cleary, K., and Kaufmann, C. (2003). A survey of surgical simulation: Applications, technology, and education. *Presence*, 12(6):599–614.

[136] Liu, G., Xi, Z., and Lien, J.-M. (2016). Nearly convex segmentation of polyhedra through convex ridge separation. *Computer-Aided Design*, 78:137–146.

[137] Liu, H., Zhang, S.-H., Cheng, M., Song, H.-W., and Trentadue, F. (2015). A minimum principle for contact forces in random packings of elastic frictionless particles. *Granular Matter*, 17(4):475–482.

[138] Liu, Y.-J., Zhou, Q.-Y., and Hu, S.-M. (2007). Handling degenerate cases in exact geodesic computation on triangle meshes. *The Visual Computer*, 23(9-11):661–668.

[139] Lorenzis, L. D., Wriggers, P., and Zavarise, G. (2012). A mortar formulation for 3D large deformation contact using NURBS-based isogeometric analysis and the augmented Lagrangian method. *Computational Mechanics*, 49(1):1–20.

[140] Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120.

[141] Lozano-Pérez, T. (1987). A simple motion-planning algorithm for general robot manipulators. *IEEE Journal on Robotics and Automation*, 3(3):224–238.

[142] Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Commununications of the ACM*, 22(10):560–570.

[143] LSTC (2017). LS-DYNA keyword user's manual.

[144] Lu, G., Third, J., and Müller, C. (2015). Discrete element models for non-spherical particle systems: From theoretical developments to applications. *Chemical Engineering Science*, 127:425–465.

[145] Ma, Y. L. and Hewitt, W. T. (2003). Point inversion and projection for NURBS curve and surface: Control polygon approach. *Computer Aided Geometric Design*, 20(2):79–99.

[146] Macaro, G. (2015). *Distinct Element Modelling of Pipe-Soil Interaction for Offshore Pipelines on Granular Soils*. D.phil thesis, University of Oxford.

[147] Malone, J. G. and Johnson, N. L. (1994). A parallel finite element contact/impact algorithm for non-linear explicit transient analysis: Part II—Parallel implementation. *International Journal for Numerical Methods in Engineering*, 37(4):591–603.

[148] Mamou, K. and Ghorbel, F. (2009). A simple and efficient approach for 3D mesh approximate convex decomposition. In *Proceedings of the 16th IEEE International Conference on Image Processing*, pages 3501–3504.

[149] Martin, W., Cohen, E., Fish, R., and Shirley, P. (2000). Practical ray tracing of trimmed NURBS surfaces. *Journal of Graphics Tools*, 5(1):27–52.

[150] Maruyama, K. (1972). A procedure to determine intersections between polyhedral objects. *International Journal of Computer & Information Sciences*, 1(3):255–266.

[151] McGeary, R. K. (1961). Mechanical packing of spherical particles. *Journal of the American Ceramic Society*, 44(10):513–522.

[152] Millan, J., Ortiz, D., Van, A., and Glotzer, S. (2014). Self-assembly of archimedean tilings with enthalpically and entropically patchy polygons. *ACS Nano*, 8(3):2918–2928.

[153] Millington, I. (2010). *Game Physics Engine Development: How to Build a Robust Commercial-Grade Physics Engine for Your Game*. CRC Press, Amsterdam ; Boston, 2 edition.

[154] Mirtich, B. (1998). V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208.

[155] Mitchell, J. S. B., Mount, D. M., and Papadimitriou, C. H. (1987). The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668.

[156] Mousakhani, M. and Jafari, A. (2016). A new model of edge-to-edge contact for three dimensional discontinuous deformation analysis. *Geomechanics and Geoengineering*, 11(2):135–148.

[157] Movshovitz, N., Asphaug, E., and Korycansky, D. (2012). Numerical modeling of the disruption of comet D/1993 F2 shoemaker-levy representing the progenitor by a gravitationally bound assemblage of randomly shaped polyhedra. *Astrophysical Journal*, 759(2).

[158] Munjiza, A. (2004). *The Combined Finite-Discrete Element Method*. Wiley edition.

[159] Munjiza, A. and Andrews, K. R. F. (1998). NBS contact detection algorithm for bodies of similar size. *International Journal for Numerical Methods in Engineering*, 43(1):131–149.

[160] Munjiza, A. and Andrews, K. R. F. (2000). Penalty function method for combined finite–discrete element systems comprising large number of separate bodies. *International Journal for Numerical Methods in Engineering*, 49(11):1377–1396.

[161] Munjiza, A., Knight, E., and Rougier, E. (2012). *Computational Mechanics of Discontinua*. Wiley.

[162] Munjiza, A., Rougier, E., and John, N. W. M. (2006). MR linear contact detection algorithm. *International Journal for Numerical Methods in Engineering*, 66(1):46–71.

[163] Murugaratnam, K., Utili, S., and Petrinic, N. (2015). A combined DEM–FEM numerical method for Shot Peening parameter optimisation. *Advances in Engineering Software*, 79(Supplement C):13–26.

[164] Museth, K., Breen, D., Whitaker, R., Mauch, S., and Johnson, D. (2005). Algorithms for interactive editing of level set models. *Computer Graphics Forum*, 24(4):821–841.

[165] N G Bourago, V. N. K. (2005). A review of contact algorithms. *Mechanics of Solids*, (1):44–85.

[166] Nakashima, H. and Oida, A. (2004). Algorithm and implementation of soil–tire contact analysis code based on dynamic FE–DE method. *Journal of Terramechanics*, 41(2–3):127–137.

[167] Neto, D. M., Oliveira, M. C., and Menezes, L. F. (2017). Surface smoothing procedures in computational contact mechanics. *Archives of Computational Methods in Engineering*, 24(1):37–87.

[168] Nezami, E. G., Hashash, Y. M., Zhao, D., and Ghaboussi, J. (2004). A fast contact detection algorithm for 3-D discrete element method. *Computers and Geotechnics*, 31(7):575–587.

[169] Ng-Thow-Hing, V., Hauser, K., and Gonzalez-Banos, H. (2012). United States Patent: 8116908 - Multi-modal push planner for humanoid robots.

[170] Nicolas, C. (29th June 2001). Smallest Enclosing Spheres.

[171] Nishita, T., Sederberg, T. W., Kakimoto, M., Nishita, T., Sederberg, T. W., and Kakimoto, M. (1990). Ray tracing trimmed rational surface patches, Ray tracing trimmed rational surface patches. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 337–345. ACM.

[172] Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer New York.

[173] Nye, B., Kulchitsky, A. V., and Johnson, J. B. (2014). Intersecting dilated convex polyhedra method for modeling complex particles in discrete element method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 38(9):978–990.

[174] Nygårds, M. and Gudmundson, P. (2002). Three-dimensional periodic Voronoi grain models and micromechanical FE-simulations of a two-phase steel. *Computational Materials Science*, 24(4):513–519.

[175] O'Connor, R. M., Torczynski, J. R., Preece, D. S., Klosek, J. T., and Williams, J. R. (1997). Discrete element modeling of sand production. *International Journal of Rock Mechanics and Mining Sciences*, 34(3):231.e1–231.e15.

[176] Oh, Y.-T., Kim, Y.-J., Lee, J., Kim, M.-S., and Elber, G. (2012). Efficient point-projection to freeform curves and surfaces. *Computer Aided Geometric Design*, 29(5):242–254.

[177] Okabe, A., Boots, B., and Sugihara, K. (1992). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, NY, USA.

[178] Ong, C.-J. and Gilbert, E. (2001). Fast versions of the Gilbert-Johnson-Keerthi distance algorithm: Additional results and comparisons. *IEEE Transactions on Robotics and Automation*, 17(4):531–539.

[179] Osipov, N., Gourgues-Lorenzon, A.-F., Marini, B., Mounoury, V., Nguyen, F., and Cailletaud, G. (2008). FE modelling of bainitic steels using crystal plasticity. *Philosophical Magazine*, 88(30-32):3757–3777.

[180] O'Sullivan, C. (2011). *Particulate Discrete Element Modelling: A Geomechanics Perspective*. Taylor & Francis.

[181] Otani, J. (2006). X-ray Computed Tomography for Geotechnical Engineering. In Desrues, J., Viggiani, G., and Bésuelle, P., editors, *Advances in X-Ray Tomography for Geomaterials*, pages 95–115. ISTE.

[182] Owen, D. R. J., Feng, Y. T., de Souza Neto, E. A., Cottrell, M. G., Wang, F., Andrade Pires, F. M., and Yu, J. (2004). The modelling of multi-fracturing solids and particulate media. *International Journal for Numerical Methods in Engineering*, 60(1):317–339.

[183] Ozaki, K., Ogita, T., and Oishi, S. (2012). A robust algorithm for geometric predicate by error-free determinant transformation. *Information and Computation*, 216:3–13.

[184] Pabst, H.-F., Springer, J. P., Schollmeyer, A., Lenhardt, R., Lessig, C., and Froehlich, B. (2006). Ray casting of trimmed NURBS surfaces on the GPU. In *Proceedings of IEEE Symposium on Interactive Ray Tracing*, pages 151–160.

[185] Pan, X. D. and Reed, M. B. (1991). A coupled distinct element—finite element method for large deformation analysis of rock masses. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, 28(1):93–99.

[186] Parker, E. and O'Brien, J. (2009). Real-time deformation and fracture in a game environment. In *Computer Animation, Conference Proceedings*, pages 165–175.

[187] Pathan, M., Tagarielli, V., Patsias, S., and Baiz-Villafranca, P. (2017). A new algorithm to generate representative volume elements of composites with cylindrical or spherical fillers. *Composites Part B: Engineering*, 110:267–278.

[188] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C. (1987). Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72(2):449–466.

[189] Perkins, E. and Williams, J. R. (2001). A fast contact detection algorithm insensitive to object sizes. *Engineering Computations*, 18(1/2):48–62.

[190] Peters, J. F., Kala, R., and Maier, R. S. (2009). A hierarchical search algorithm for discrete element method of greatly differing particle sizes. *Engineering Computations; Bradford*, 26(6):621–634.

[191] Piegl, L. A. and Tiller, W. (1997). *The NURBS Book*. Springer, 2nd edition.

[192] Piegl, L. A. and Tiller, W. (2001). Parametrization for surface fitting in reverse engineering. *Computer-Aided Design*, 33(8):593–603.

[193] Proudhon, H., Buffière, J.-Y., and Fouvry, S. (2007). Three-dimensional study of a fretting crack using synchrotron X-ray micro-tomography. *Engineering Fracture Mechanics*, 74(5):782–793.

[194] Quey, R. (2017). Personal communication.

[195] Quey, R., Dawson, P., and Barbe, F. (2011). Large-scale 3D random polycrystals for the finite element method: Generation, meshing and remeshing. *Computer Methods in Applied Mechanics and Engineering*, 200(17-20):1729–1745.

[196] Quey, R. and Renversade, L. (2018). Optimal polyhedral description of 3D polycrystals: Method and application to statistical and synchrotron X-ray diffraction data. *Computer Methods in Applied Mechanics and Engineering*, 330(Supplement C):308–333.

[197] Rabbitz, R. (1994). Fast collision detection of moving convex polyhedra. *Graphics Gems IV*, pages 83–109.

[198] Redenbach, C. (2009). Microstructure models for cellular materials. *Computational Materials Science*, 44(4):1397–1407.

[199] Redon, S., Kheddar, A., and Coquillart, S. (2002). Fast Continuous Collision Detection between Rigid Bodies. *Computer Graphics Forum*, 21(3):279–287.

[200] Salciarini, D. and Tamagnini, C. (2009). A hypoplastic macroelement model for shallow foundations under monotonic and cyclic loads. *Acta Geotechnica*, 4(3):163–176.

[201] Sankaran, S. and Zabaras, N. (2007). Computing property variability of polycrystals induced by grain size and orientation uncertainties. *Acta Materialia*, 55(7):2279–2290.

[202] Santasusana, M., Irazábal, J., Oñate, E., and Carbonell, J. M. (2016). The Double Hierarchy Method. A parallel 3D contact method for the interaction of spherical particles with rigid FE boundaries using the DEM. *Computational Particle Mechanics*, 3(3):407–428.

[203] Sarma, G. B. and Dawson, P. R. (1996). Effects of interactions among crystals on the inhomogeneous deformations of polycrystals. *Acta Materialia*, 44(5):1937–1953.

[204] Sato, Y., Hirata, M., Maruyama, T., and Arita, Y. (1996). Efficient collision detection using fast distance-calculation algorithms for convex and non-convex objects. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 1, pages 771–778.

[205] Schoenberg, I. J. (1988). Contributions to the problem of approximation of equidistant data by analytic functions. In *I. J. Schoenberg Selected Papers*, Contemporary Mathematicians, pages 58–87. Birkhäuser, Boston, MA.

[206] Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., and Hanrahan, P. (2008). Larrabee: A many-core x86 architecture for visual computing. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 18:1–18:15, New York, NY, USA. ACM.

[207] Sellgren, U., Björklund, S., and Andersson, S. (2003). A finite element-based model of normal contact between rough surfaces. *Wear*, 254(11):1180–1188.

[208] Sevilla, R., Fernández-Méndez, S., and Huerta, A. (2008). NURBS-enhanced finite element method for Euler equations. *International Journal for Numerical Methods in Fluids*, 57(9):1051–1069.

[209] Shewchuk, J. R. (1997). Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–363.

[210] Shi, G.-H. (1992). Manifold Method of Material Analysis,. Technical report.

[211] Shrivastava, P. and Das, S. (2013). Fast area of contact computation for collision detection of a deformable object using FEM. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–4.

[212] Sonon, B., François, B., and Massart, T. J. (2015). An advanced approach for the generation of complex cellular material representative volume elements using distance fields and level sets. *Computational Mechanics*, 56(2):221–242.

[213] Steuben, J. C., Iliopoulos, A. P., and Michopoulos, J. G. (2016). Discrete element modeling of particle-based additive manufacturing processes. *Computer Methods in Applied Mechanics and Engineering*, 305(Supplement C):537–561.

[214] Stewart, P. J. and Buttolo, P. (2007). United States Patent: 7191104 - Method of real-time collision detection between solid geometric models.

[215] Sugihara, K., Iri, M., Inagaki, H., and Imai, T. (2000). Topology-oriented implementation - An approach to robust geometric algorithms. *Algorithmica (New York)*, 27(1):5–20.

[216] Sukumar, N., Moës, N., Moran, B., and Belytschko, T. (2000). Extended finite element method for three-dimensional crack modelling. *International Journal for Numerical Methods in Engineering*, 48(11):1549–1570.

[217] Tasora, A. and Anitescu, M. (2011). A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics. *Computer Methods in Applied Mechanics and Engineering*, 200(5-8):439–453.

[218] Tereshchenko, V., Chevokin, S., and Fisunenko, A. (2013). Algorithm for finding the domain intersection of a set of polytopes. *Procedia Computer Science*, 18:459–464.

[219] Tian, Y., Cassidy, M. J., and Gaudin, C. (2010). Advancing pipe–soil interaction models in calcareous sand. *Applied Ocean Research*, 32(3):284–297.

[220] Trias, D., Costa, J., Turon, A., and Hurtado, J. E. (2006). Determination of the critical size of a statistical representative volume element (SRVE) for carbon reinforced polymers. *Acta Materialia*, 54(13):3471–3484.

[221] Turnbull, C. and Cameron, S. (1998). Computing distances between NURBS-defined convex objects. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3685–3690.

[222] Turner, A. K., Kim, F. H., Penumadu, D., and Herbold, E. B. (2016). Meso-scale framework for modeling granular material using computed tomography. *Computers and Geotechnics*, 76:140–146.

[223] Udupa, S. (1977). Collision detection and avoidance in computer controlled manipulators. In *5th International Joint Conference of Artificial Lntelligence*, pages 737–748, Cambridge.

[224] van den Bergen, G. (1999). A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25.

[225] van den Bergen, G. (2003). *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann.

[226] van den Bergen, G. (2017). Solid3: SOLID - Software Library for Interference Detection.

[227] van der Linden, J. H., Narsilio, G. A., and Tordesillas, A. (2016). Machine learning framework for analysis of transport through complex networks in porous, granular media: A focus on permeability. *Physical Review E*, 94(2):022904.

[228] Van Welbergen, H., Van Basten, B. J. H., Egges, A., Ruttkay, Z. M., and Overmars, M. H. (2010). Real time animation of virtual humans: A trade-off between naturalness and control. *Computer Graphics Forum*, 29(8):2530–2554.

[229] Vecchio, I., Redenbach, C., and Schladitz, K. (2014). Angles in Laguerre tessellation models for solid foams. *Computational Materials Science*, 83:171–184.

[230] Vlahinić, I., Kawamoto, R., Andò, E., Viggiani, G., and Andrade, J. E. (2017). From computed tomography to mechanics of granular materials via level set bridge. *Acta Geotechnica*, 12(1):85–95.

[231] Voivret, C., Radjaï, F., Delenne, J.-Y., and El Youssoufi, M. S. (2009). Multiscale force networks in highly polydisperse granular media. *Physical Review Letters*, 102(17):178001.

[232] Wachs, A., Girolami, L., Vinay, G., and Ferrer, G. (2012). Grains3D, a flexible DEM approach for particles of arbitrary convex shape — Part I: Numerical model and validations. *Powder Technology*, 224:374–389.

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

[234] Walizer, L. E. and Peters, J. F. (2011). A bounding box search algorithm for DEM simulation. *Computer Physics Communications*, 182(2):281–288.

[235] Wang, F., Cheng, J., and Yao, Z. (2001). FFS contact searching algorithm for dynamic finite element analysis. *International Journal for Numerical Methods in Engineering*, 52(7):655–672.

[236] Wang, F.-J., Wang, L.-P., Cheng, J.-G., and Yao, Z.-H. (2007). Contact force algorithm in explicit transient analysis using finite-element method. *Finite Elem. Anal. Des.*, 43(6-7):580–587.

[237] Wang, S.-H., Jiang, J., and Stack, M. M. (2015). Methodology development for investigation of slurry abrasion corrosion by integrating an electrochemical cell to a miller tester. *Journal of Bio- and Tribo-Corrosion*, 1(2):9.

[238] Wang, S. P. and Nakamachi, E. (1997). The inside–outside contact search algorithm for finite element analysis. *International Journal for Numerical Methods in Engineering*, 40(19):3665–3685.

[239] Wang, Z. and Li, P. (2017). Voronoi cell finite element modelling of the intergranular fracture mechanism in polycrystalline alumina. *Ceramics International*, 43(9):6967–6975.

[240] Webster, R. (1994). *Convexity*. Oxford University Press, Oxford; UK.

[241] Wejrzanowski, T., Skibinski, J., Szumbarski, J., and Kurzydlowski, K. J. (2013). Structure of foams modeled by Laguerre–Voronoi tessellations. *Computational Materials Science*, 67:216–221.

[242] Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science*, Lecture Notes in Computer Science, pages 359–370. Springer, Berlin, Heidelberg.

[243] Westman, A. E. R. and Hugill, H. R. (1930). The packing of particles. *Journal of the American Ceramic Society*, 13(10):767–779.

[244] Williams, J. R., Perkins, E., and Cook, B. (2004). A contact algorithm for partitioning N arbitrary sized objects. *Engineering Computations; Bradford*, 21(2-4):235–248.

[245] Wouterse, A., Luding, S., and Philipse, A. P. (2009). On contact numbers in random rod packings. *Granular Matter*, 11(3):169–177.

[246] Wriggers, P. (1995). Finite element algorithms for contact problems. *Archives of Computational Methods in Engineering*, 2(4):1–49.

[247] Wriggers, P. (2006). *Computational Contact Mechanics*. Computational Contact Mechanics. Springer.

[248] Wu, H., Guo, N., and Zhao, J. (2017). Multiscale modeling and analysis of compaction bands in high-porosity sandstones. *Acta Geotechnica*, pages 1–25.

[249] Xiang, J., Munjiza, A., and Latham, J.-P. (2009). Finite strain, finite rotation quadratic tetrahedral element for the combined finite–discrete element method. *International Journal for Numerical Methods in Engineering*, 79(8):946–978.

[250] Yoon, S.-e., Kim, Y. J., and Harada, T. (2010). Recent advances in real-time collision and proximity computations for games and simulations. In *ACM SIGGRAPH ASIA 2010 Courses*, SA '10, pages 22:1–22:110, New York, NY, USA. ACM.

[251] Zhang, L., Xu, W., Liu, C., Ma, X., and Long, J. (2017). Quantitative analysis of surface roughness evolution in FCC polycrystalline metal during uniaxial tension. *Computational Materials Science*, 132:19–29.

[252] Zhao, B. and Wang, J. (2016). 3D quantitative shape analysis on form, roundness, and compactness with μCT. *Powder Technology*, 291(Supplement C):262–275.

[253] Zhao, B., Wang, J., Coop, M., Viggiani, G., and Jiang, M. (2015). An investigation of single sand particle fracture using x-ray micro-tomography. *Geotechnique*, 65(8):625–641.

[254] Zhao, D., Nezami, E., Hashash, Y., and Ghaboussi, J. (2006). Three-dimensional discrete element simulation for granular materials. *Engineering Computations (Swansea, Wales)*, 23(7):749–770.

[255] Zhao, Z., Kuchnicki, S., Radovitzky, R., and Cuitiño, A. (2007). Influence of in-grain mesh resolution on the prediction of deformation textures in fcc polycrystals by crystal plasticity FEM. *Acta Materialia*, 55(7):2361–2373.

[256] Zheng, F., Jiao, Y.-Y., Gardner, M., and Sitar, N. (2017). A fast direct search algorithm for contact detection of convex polygonal or polyhedral particles. *Computers and Geotechnics*, 87(Supplement C):76–85.

[257] Zheng, Y., Chew, C. M., and Adiwahono, A. H. (2011). A GJK-based approach to contact force feasibility and distribution for multi-contact robots. *Robotics and Autonomous Systems*, 59(3–4):194–207.

[258] Zheng, Y. and Yamane, K. (2013). Ray-shooting algorithms for robotics. *IEEE Transactions on Automation Science and Engineering*, 10(4):862–874.

[259] Zhi-Hua, Z. and Nilsson, L. (1989). A contact searching algorithm for general contact problems. *Computers & Structures*, 33(1):197–209.

[260] Zhigang, F., Jianxun, J., Jie, X., and Xiaochi, W. (2010). Efficient collision detection using bounding volume hierarchies of OBB-AABBs and its application. In *2010 International Conference on Computer Design and Applications (ICCDA)*, volume 5, pages V5–242–V5–246.

[261] Zhong, Z.-H. and Nilsson, L. (1994). Automatic contact searching algorithm for dynamic finite element analysis. *Computers & Structures*, 52(2):187–197.

[262] Zienkiewicz, O. and Taylor, R. (2005). *The Finite Element Method Set*. The Finite Element Method Set. Elsevier, 6th edition.