Exact calculation of the overlap volume of spheres and mesh elements

Severin Strobl[a,*], Arno Formella[b], Thorsten Pöschel[a]

[a] *Institute for Multiscale Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Nägelsbachstr. 49 b, 91052 Erlangen, Germany*
[b] *Department of Computer Science, Universidade de Vigo, Ourense, Spain*

Abstract

An algorithm for the exact calculation of the overlap volume of a sphere and a tetrahedron, wedge, or hexahedron is described. The method can be used to determine the exact local solid fractions for a system of spherical, non-overlapping particles contained in a complex mesh, a question of significant relevance for the numerical solution of many fluid-solid interaction problems. While challenging due to the limited machine precision, a numerically robust version of the calculation maintaining high computational efficiency is devised. The method is evaluated with respect to the numerical precision and computational cost. It is shown that the exact calculation is only limited by the machine precision and can be applied to a wide range of size ratios, contrary to previously published methods. Eliminating this constraint enables the usage of meshes with higher resolution near the system boundaries for coupled CFD–DEM simulations. The numerical robustness is further illustrated by applying the method to highly deformed mesh elements. The full source code of the reference implementation is made available under an open-source license.

Keywords: overlap volume, local solid fraction, numerical robustness, sphere-mesh overlap

1. Introduction

Spherical objects are still the most commonly used primitives in a wide range of numerical simulations. Despite their alleged simplicity, particle methods often rely on spherical particles benefiting from an efficient contact detection and the availability of physically realistic force models with a relatively low numerical cost.

Due to its importance for many industrial processes, one highly active area of research is the simulation of fluidized beds [1–5]. Here, the dynamics of the individual particles can be modeled using a Discrete Element Method (DEM) which is coupled to a Computational Fluid Dynamics (CFD) simulation. The coupling between the solid and fluid phase is performed via the transfer of momentum employing a drag force [6, 7] acting on the particles and a sink term in the momentum equations of the gas phase. This two-way coupling requires knowledge of the local solid volume fraction, i.e. the fraction of a certain control volume occupied by solid particles. In the simplest case, this control volume defined by the CFD discretization scheme has the shape of a cuboid. For many real world applications the CFD discretization has to use a more complex grid in order to capture the complex geometry of the system boundary, for instance of an industrial device like a Wurster-coater [8]. This then necessitates the usage of tetrahedral, wedge-shaped or hexahedral elements in the CFD mesh. The influence of the method used for the calculation of the volume fraction on the overall simulation result has been shown recently and especially the smoothness of the solution as particles enter and leave cells was identified as a crucial factor [9, 10].

A different application requiring the exact calculation of the overlap volume between spheres and mesh elements is the calculation of density fields in the context of granular media. There, the exact dynamics of the individual particles is often of inferior interest, especially for systems involving millions of particles. Instead, macroscopic properties like the average density or solid fraction become the characteristic parameters describing the packing of particles in a system [11]. Often coarse graining formulations featuring a Gaussian function as a smoothing kernel are used [12]. For packings in complex geometries, however, special care has to be taken to avoid anomalies near the walls [13].

*Corresponding author
Email address: severin.strobl@fau.de (Severin Strobl)

A calculation of the density using a mesh of tetrahedral or hexahedral elements perfectly aligned with the system boundaries avoids those difficulties.

Also for microscopic simulations, the calculation of the common volume of spherical particles and some mesh elements is of relevance. For the modeling of a polymer suspended in a stochastic hard-sphere fluid the exact free volume, that is, the volume not occupied by the polymer, should be taken into account [14]. This is required as the collision rate of the particles forming the fluid phase depends directly on the free volume.

These examples illustrate that the calculation of the common volume or overlap between a sphere and various typical mesh elements is an important step in different numerical simulation schemes. As shown in the next sections, however, this allegedly simple problem proves to be quite challenging in the general case. In Sec. 2, the previous work in this field is discussed, covering both exact solutions and approximations. In Sec. 3, a numerically robust approach for the exact calculation of the overlap volume between a sphere and a tetrahedron, wedge, or hexahedron is introduced. The challenges of implementing this method due to the limited floating point precision of real world computers and ways to overcome those are presented in Sec. 4. Based on this, a thorough validation of the method as well as numerical benchmarks are performed in Sec. 5.

2. Previous work

A wide range of techniques have been developed over the years to determine the spatially resolved solid fraction, ranging from smoothing methods over approximations to exact calculations. While it is beyond the scope of this work to give a detailed description of all the approaches available in the literature, a selection of the most relevant in this context will be presented. The simplest estimation of the solid fraction can be achieved independent of the shape of the mesh element by only taking the center of the sphere into account and allocating the full volume of the sphere to the corresponding element. The error of this method, however, becomes large even for moderate size ratios between the spheres and the mesh elements. Additionally, the values of the solid fraction are not smooth in dynamic systems due to jumps generated by spheres entering and leaving the individual mesh elements. A modification of this scheme is the additional smoothing between neighboring mesh elements by computing the average solid fraction for a given element and all its direct neighbors [15]. Yet it was shown that these two methods can lead to a reduced rate of convergence [10] as well as non-physical effects in some cases [9]. Thus these schemes will not be considered further and the focus is put on more advanced solutions.

In the interest of a clear presentation, first only the case of the overlap of a sphere and regular cells, that is, cuboids will be discussed. In one of the earliest works on coupled CFD–DEM simulations, the solid fraction in three dimensions was determined from the exactly calculated value for a two-dimensional system by scaling with a factor derived from comparing regular packings in two and three dimensions [3]. Another commonly used approach is the approximation of the sphere as a cube and the subsequent summation of the fractions of the cube contained in the single cuboidal mesh elements [16, 17]. As shown later, this method leads to a maximum error of 20%, which can be reduced to less than 2.5% when using a third order fit achieved by numerical integration of some of the partial overlaps between the sphere and the regular mesh [18]. A fully analytical solution without any approximations for the calculation of the common volume of a sphere and a cuboid can be derived [19]. It uses a decomposition of the overlap volume in sections defined by the faces, edges, and corners of the cuboid (see Fig. 1). Some alternatives are more complex numerical integration schemes like the one proposed by Bnà et al. [20], that can also handle implicitly defined surfaces. While those methods are more general in some respect due to the support for non-spherical shapes, they rely on numerical integration instead of a direct analytical solution and are typically restricted to regular mesh elements.

For the case of more complex mesh elements, that is, tetrahedra, wedges, or hexahedra, often employed in modern CFD simulations in order to represent the complex boundaries of the system, a number of methods have been proposed as well. An approach independent of the shape of the mesh element is numerical integration, either on a regular grid or using Monte Carlo integration. The former corresponds to dividing the sphere into a number of cuboids and counting the number of overlaps of the center points with each individual mesh element. To reduce the computational cost, the number of discretization steps is often quite small, for example 10–20 cuboids per sphere [8]. The exact solution for this problem is quite involved and numerically and computationally challenging. The exact overlapping volume of a sphere and a tetrahedron with a common center is derived by Richards [21]. The practically more relevant case of a sphere and a tetrahedron with arbitrary relative position and size is treated in [22]. This description contains some

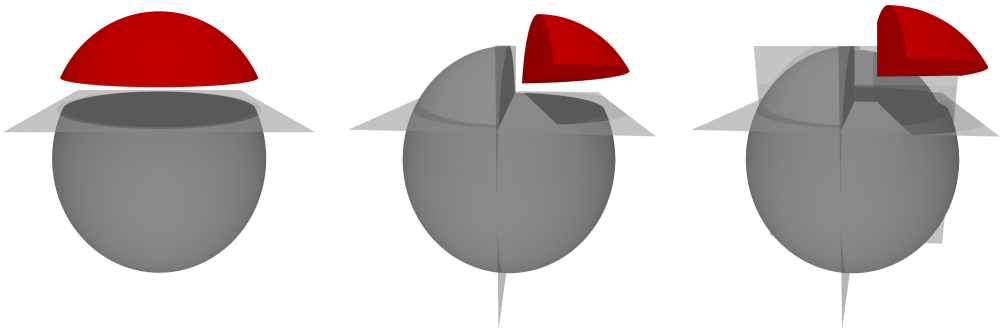


Figure 1: Intersection between one, two, and three half-spaces formed by the sides of a cuboid and a sphere. This corresponds to the intersection of a face (a), an edge (b), and a corner (c) with the sphere. The volume of interest which is the portion of the sphere inside of the cuboid is highlighted.

inaccuracies and does not take into account the influence of the limited floating point precision on the robustness of the calculation. Using a similar approach, other authors [23] devised a method to calculate the exact overlap volume of a sphere and a tetrahedron, wedge, or hexahedron. This method, however, is limited with respect to the size ratio between the sphere and the mesh elements: Due to the construction of the solution, the sphere must not overlap more than one corner of any mesh element at a time, meaning the maximum diameter of all spheres in the system has to be smaller than the shortest edge of all elements in the mesh.

In general, this is an acceptable restriction for coupled CFD–DEM simulations as the ratio between the resolution of the CFD mesh and the particle diameter should be larger than ≈ 1.63 [10]. This constraint may still lead to problems especially for complex geometries, where it might be required to locally refine the CFD grid, resulting in mesh elements smaller than the particle size. Also with the other applications described in the introduction in mind, a general method independent of the size ratio is highly desirable.

In conclusion, it can be said that the available solutions for the calculation of the overlapping volume of a sphere and a mesh element of common shape are either inherently inaccurate as they are only approximations, limited in their applicability, or lack numerical robustness. Thus, an algorithm which provides an exact result while being general enough and numerically robust is required.

3. Exact overlap calculation using decomposition

In this section, the approach to calculate the overlap volume of a sphere and an arbitrary tetrahedron, wedge, or hexahedron is described. It is possible to use the same method for tetrahedra, wedges, and convex hexahedra due to the common geometrical features, namely planar faces and vertices formed by exactly three intersecting faces. The basic entities are the same, only the numbers of vertices, edges and faces differ. As the tetrahedron is the simplest of the three geometric shapes, the basic method will be described using a sphere and a tetrahedron in order to simplify the presentation. First, the basic idea is described and the involved geometric shapes are defined. Then the full algorithm and the necessary building blocks for the exact calculation are presented. Those building blocks are then examined in detail where necessary in the following sections.

3.1. Basic idea

The overlapping volume of a sphere and a tetrahedron cannot be calculated analytically directly. Rather the volume has to be decomposed into multiple basic parts. This approach is similar to the one described for regular cuboids in Sec. 2 (see also Fig. 1) and is originally presented in an appendix [22]. This original presentation by Bernardeau and van de Weygaert [22], however, contains some inconsistencies and only describes a part of the solution with the required level of detail. The influence of the limited numeric precision of real world computer systems is not taken into account, neither for the classification of the required sub-volumes of the original overlapping volume, nor the

calculation of the volumes of those simpler geometric shapes. The classification and recombination of these sub-volumes constitutes the major part of the method presented here, as a single instance of an erroneous decomposition leads to a completely wrong result rather than a slight deviation in the final volume. Thus, any implementation following the original method [22] directly will lack numerical robustness, in contrast to the presented method.

The overlap calculation is trivial for two cases, namely full overlap and no overlap. Full overlap occurs either if the sphere is completely inside the tetrahedron, that is, the center of the sphere lies within the tetrahedron and no face of the tetrahedron is intersecting the sphere, or if all vertices of the tetrahedron lie within the sphere. The resulting overlap volume is the full volume of the sphere or the tetrahedron respectively. If this is not the case and no face of the tetrahedron intersects the sphere, the overlap volume is zero.

The case of partial overlap is more complicated and requires a decomposition of the overlapping volume. Independent of the relative size or position of the sphere, the initial overlap volume is always set to the full volume of the sphere. Using this as the starting value, the parts of the sphere that are outside of the tetrahedron are then subtracted subsequently by iterating over the faces of the tetrahedron. The idea is to subtract the volume of the caps of the sphere cut off by the planes forming the sides of the tetrahedron (compare Fig. 2a). Obviously, during this process certain parts of the sphere are accounted for multiple times in case the sphere intersects any edges or vertices of the tetrahedron, as shown in Fig. 2b and c.

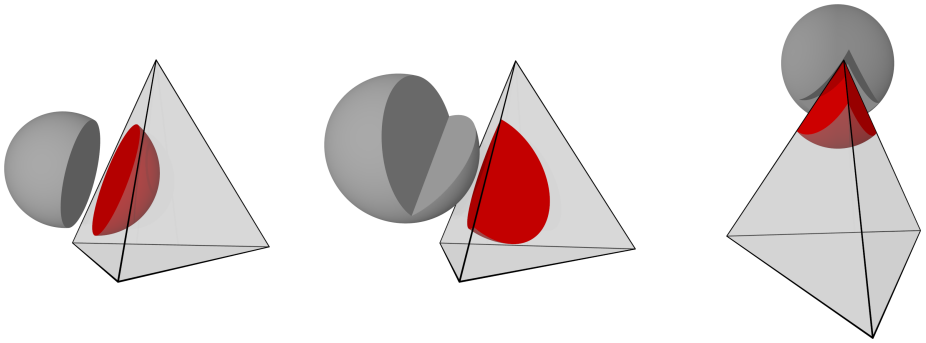


Figure 2: Intersections between the different primitives of a tetrahedron and a sphere: intersection of a face (a), an edge (b), and a corner or vertex with the sphere (c). The resulting overlap volume is marked in red and has the shape of a spherical cap, a general spherical wedge, or a special cone.

For an edge, this volume erroneously accounted for twice has the shape of a general spherical wedge. While a common spherical wedge is defined by two intersecting planes containing the center of the spheres, that is, two great circles, a general spherical wedge on the contrary does not necessarily touch the center of the sphere, see e.g. the configuration sketched in Fig. 3. When considering the overlapping volume between a sphere and a cuboid, the planes forming an edge are always perpendicular (Fig. 1b). This is not the case for the edges of a tetrahedron, here the angle between the two planes defining an edge is arbitrary in the range $(0, \pi)$. There is no general closed expression for the volume of such a shape, but the volume can be decomposed into two sub-volumes which in turn have exact analytical solutions. For the sake of simplicity, the term *regularized general wedge* will be used for the sub-volumes. The decomposition of a general spherical wedge into two regularized spherical wedges is covered in Sec. 4.2, and the calculation of the volume of one of those volumes in Sec. 4.3.

For a vertex of the tetrahedron lying inside of the sphere (Fig. 2c), the situation is further complicated. As at each vertex three edges formed between the three faces are intersecting, for each vertex inside of the sphere one has to correct for the erroneously added volume of the spherical caps using three general spherical wedges. Those wedges themselves overlap one another, however, so the simple correction is not sufficient. Fig. 4 sketches the three spherical caps and the corresponding general spherical wedges defined by the pairwise intersection of the caps. The common volume of the spherical wedges is the volume outside of the tetrahedron cut out of the sphere by the three faces forming the vertex. For clarity and following the notation in previous work [22], we will refer to this shape used for the correction at vertices located inside of the sphere as a *cone* from here on forward. This volume in turn can again

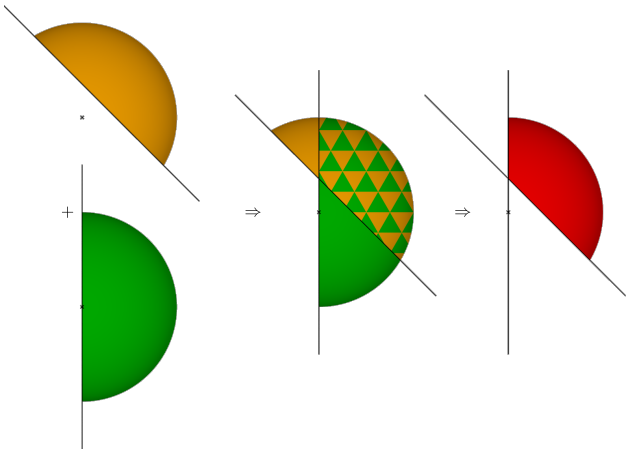


Figure 3: The Boolean operation union applied to two spherical caps (green and orange) cut by planes from a single sphere forms a *general spherical wedge*, that is, a spherical wedge where the intersection line of the two planes does not necessarily contain the center of the sphere. The common volume of the two caps marked by the triangular pattern in the center corresponds to the intersection or overlap volume of the caps. This volume again has the shape of a general spherical wedge and is highlighted in red on the right side.

be decomposed into two parts: a tetrahedron defined by the vertex and the intersection points of the edges and the sphere (points $\vec{v}$, $\vec{i}_{0,1}$, $\vec{i}_{0,2}$, and $\vec{i}_{1,2}$ in Fig. 5) and an additional part of a spherical cap, limited by the planes defining the sides of the tetrahedron. The base plane of this spherical cap is given by the three intersection points. The three parts to be excluded from this volume are again general spherical wedges formed by the base plane and the three faces f_0, f_1, and f_2 (Fig. 5). So the total volume required for the correction at a vertex is the volume of the cone:

$$V_{\text{cone}} = V_{\text{tetra}} + V_{\text{cap}} - \sum_{\text{sides}} V_{\text{wedge}} \,. \tag{1}$$

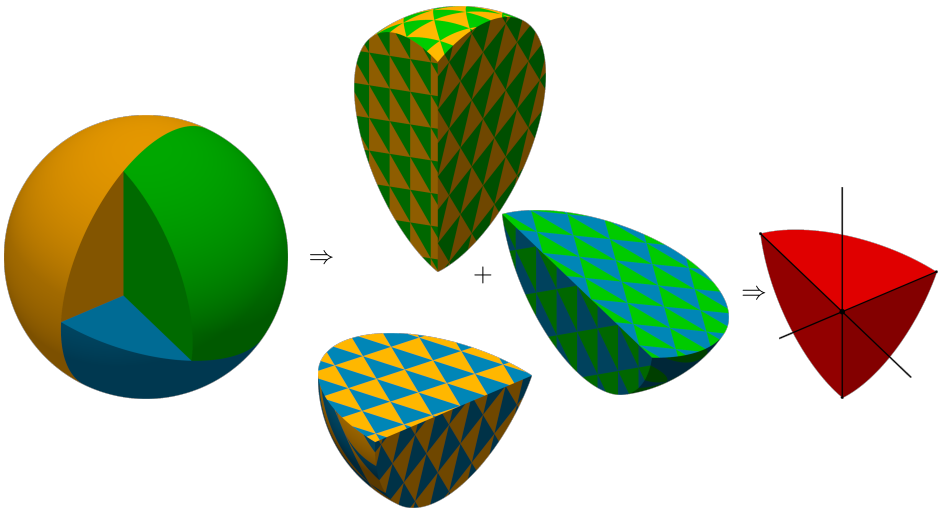


Figure 4: The pairwise intersection of three spherical caps formed by the faces of a tetrahedron meeting at a vertex leads to three general spherical wedges (compare Fig. 3). The patterns in the central part of the figure indicate the two caps from the left forming the wedges. The common volume of those wedges highlighted in red on the right is a complex shape bounded by the sphere and the planes corresponding to the faces.

Using this expression, the full overlap volume between the sphere and the tetrahedron is given by:

$$V_{\text{overlap}} = V_{\text{sphere}} - \sum_{\text{faces}} V_{\text{cap}} + \sum_{\text{edges}} V_{\text{wedge}} - \sum_{\text{vertices}} V_{\text{cone}} \,, \tag{2}$$

where the sums are only over the primitives of the element actually intersecting the sphere.

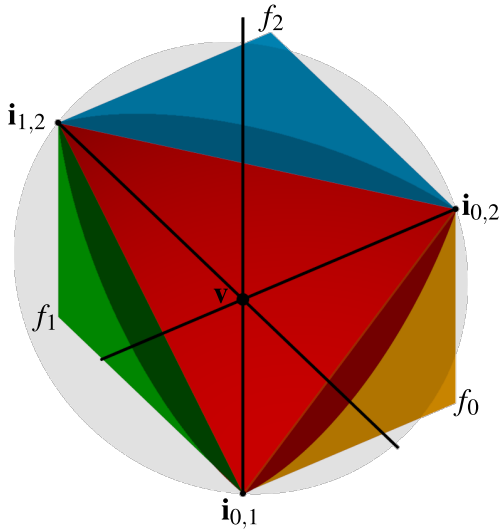


Figure 5: Decomposition of the volume defined by the three intersecting faces f_0, f_1, and f_2 and the sphere according to Fig. 4 into a simple tetrahedron and a spherical cap from which three spherical wedges are cut off (highlighted in red). The tetrahedron is defined by the intersection points of the edges between the planes and the sphere ($\vec{i}_{0,1}$, $\vec{i}_{0,2}$, $\vec{i}_{1,2}$) and the vertex $\vec{v}$ itself. The spherical wedges are bounded by the permutations of the side planes and the base plane containing the intersection points. The colors correspond to the planes defining the spherical caps in Fig. 4.

All partial volumes contained in Eqs. 1 and 2 except for the general spherical wedge possess simple analytical solutions. The main difficulty is the numerically stable detection of intersections and the resulting decomposition into the various sub-volumes, which is the focus of Sec. 4.1.

3.2. Algorithm

The steps required for an implementation of Eq. 2 are outlined in Algorithm 1, using as input the specification of the sphere and the mesh element. In order to optimize the computational performance of the calculation, a computationally cheap check using the axes-aligned bounding boxes (AABBs) of the sphere and the element should be added to exclude non-intersecting objects early on. Of course, this can be directly integrated into a broad-phase collision detection scheme commonly used to handle various geometric objects of different sizes.

The case of the mesh element lying fully inside of the sphere is detected via the call to `contains` on line 3. This check can be implemented efficiently by testing whether all vertices of the element are contained within the sphere. As mentioned above the detection of the intersecting primitives (vertices, edges, faces) is challenging due to the limited numerical precision available and thus performed in a dedicated routine `intersectPrimitives` (line 6) detailed in Sec. 4.1. This function returns three sets, $\mathcal{M}_{\text{vertices}}$, $\mathcal{M}_{\text{edges}}$, and $\mathcal{M}_{\text{faces}}$, marking the primitives of the element which intersect the sphere. It is crucial that those sets are consistent, e.g., if a vertex is marked, so have to be all the edges and faces intersecting at this vertex. Those sets also allow for the handling of two special cases: Containment of the sphere within the element (line 7, together with a check whether the center of the sphere is inside of the element) and no intersection (line 9). The remainder of Algorithm 1 determines the overlap volume for the case of partial overlap, following Eq. 2. First, in lines 13–15 the volumes of the spherical caps cut from the sphere by the faces of the mesh element are subtracted. The next two loops perform the adjustment of the overlap volume for the parts accounted for twice at the edges (lines 16–19) and the correction for those general spherical wedges in turn at the vertices (lines 20–23). Both steps make heavy use of the function `sphericalWedge` calculating the volume of a general spherical wedge

Algorithm 1 Calculate the overlap volume of a sphere and a mesh element (tetrahedron, wedge, hexahedron).

procedure OVERLAPVOLUME(sphere, element)
  // Optimization: Perform coarse collision detection using AABBs.
  if CONTAINS(sphere, element) **then**
    return VOLUME(element)
  end if
  $\mathcal{M}_{\text{vertices}}, \mathcal{M}_{\text{edges}}, \mathcal{M}_{\text{faces}} \leftarrow$INTERSECTPRIMITIVES(sphere, element)
  if $\mathcal{M}_{\text{faces}} = \varnothing \wedge$ CONTAINS(element, CENTER(sphere)) **then**
    return VOLUME(sphere)
  else if $\mathcal{M}_{\text{vertices}} = \varnothing \wedge \mathcal{M}_{\text{edges}} = \varnothing \wedge \mathcal{M}_{\text{faces}} = \varnothing$ **then**
    return 0
  end if
  $V_{\text{overlap}} \leftarrow$ VOLUME(sphere)
  for $f \in \mathcal{M}_{\text{faces}}$ **do**
    $V_{\text{overlap}} \leftarrow V_{\text{overlap}} -$ CAPVOLUME(sphere, f)
  end for
  for $e \in \mathcal{M}_{\text{edges}}$ **do**
    $f_0, f_1 \leftarrow$ FACES(e)
    $V_{\text{overlap}} \leftarrow V_{\text{overlap}} +$ SPHERICALWEDGE(sphere, f_0, f_1)
  end for
  for $v \in \mathcal{M}_{\text{vertices}}$ **do**
    $f_0, f_1, f_2 \leftarrow$ FACES(v)
    $V_{\text{overlap}} \leftarrow V_{\text{overlap}} -$ CONE(sphere, f_0, f_1, f_2)
  end for
  return V_{overlap}
end procedure

defined by two planes. For the edges this function is called directly, but also during the calculation of the cone volume in `cone` (compare Eq. 1 and Fig. 5) three calls to `sphericalWedge` are performed. An efficient and numerically robust implementation of this method is discussed in Sec. 4.2.

4. Numerical implementation

While in principle Algorithm 1 is relatively simple to implement, the real challenge arises from the limited numerical precision of common floating point implementations. Special care has to be taken along the implementation of every single step of the algorithm as numerical errors do not only accumulate, but can also lead to catastrophic failure of the calculation, as described in the next section. Hence, the individual functions used in Algorithm 1 have to be tuned to be numerically robust while limiting the impact on computational performance, which is the focus of the following sub-sections.

4.1. Stable intersection detection

For the implementation of Eq. 2, it is necessary to first identify the different entities of the mesh element intersecting the sphere. This means consistent sets for the intersecting vertices, edges, and faces have to be built. While for exact arithmetics this is trivial, the limited machine precision of real computer systems poses a challenge. As an example, the configuration where a sphere just touches a single vertex of a tetrahedron will be considered. A naive implementation would test all vertices, edges and faces of the element for a possible overlap with the sphere. Depending on the exact implementation, one possible outcome is that an intersection is detected for the vertex and two faces, but not the edge between the faces. This could result from the usage of different algorithms for the geometric primitives:

- vertex/sphere $\Rightarrow$ point/sphere;

- edge/sphere ⇒ point/cylinder;
- face/sphere ⇒ point/prism, 3 × point/cylinder, 3 × point/sphere.

As a consequence, Eq. 2 would lead to a wrong result, as the volume of the spherical wedge formed by the two faces sharing the edge would not be added back, so the correction by the cone-shaped volume would lead to an invalid result. This is just one case, where a inherently inconsistent set of primitives leads to a wrong result.

As a possible solution for this problem, one might try to introduce a certain threshold ε which has to be surpassed in order for an intersection to be reported. This approach, however, just shifts the problem but does not ultimately solve it, as for any value of ε a case can be constructed that leads to the original erroneous behavior. Thus, a different approach is taken in the present implementation. Instead of testing each primitive by itself leading to duplicate tests, only the minimum number of intersection tests is performed and all other information is derived from them.

Algorithm 2 Determine the sets of intersecting primitives (vertices, edges, faces) between a sphere and a mesh element (tetrahedron, wedge, hexahedron).

```
1:  procedure INTERSECTPRIMITIVES(sphere, element)
2:      M_vertices, M_edges, M_faces ← {}
3:      // Iterate over all the edges:
4:      for e ∈ EDGES(element) do
5:          i_0, i_1 ← LINESPHEREINTERSECTION(e, sphere)
6:          // Check for no intersection or sphere touching edge:
7:          if i_0 > 1 − ε ∨ i_1 < ε ∨ (i_0 > 0 ∧ i_1 < 1 ∧ i_1 − i_0 < ε) then
8:              continue
9:          else
10:             if i_0 < 0 ∧ i_1 > 1 then
11:                 M_vertices ← M_vertices ∪ VERTICES(e)
12:             else
13:                 M_vertices ← M_vertices ∪ CLOSEST(sphere, VERTICES(e))
14:             end if
15:         end if
16:         M_edges ← M_edges ∪ {e}
17:         M_faces ← M_faces ∪ FACES(e)
18:     end for
19:     // Check set of marked vertices for consistency:
20:     for v ∈ M_vertices do
21:         if EDGES(v) ⊈ M_edges then
22:             M_vertices ← M_vertices \ {v}
23:         end if
24:     end for
25:     // Iterate over all the faces:
26:     for f ∈ FACES(e) do
27:         if POLYGONSPHEREINTERSECTION(f, sphere) ≠ ∅ then
28:             M_faces ← M_faces ∪ {f}
29:         end if
30:     end for
31:     return M_vertices, M_edges, M_faces
32: end procedure
```

A possible realization is presented in Algorithm 2, taking as input the description of the sphere and the mesh element (tetrahedron, wedge, hexahedron). The procedure starts with empty sets for the intersecting or marked vertices $\mathcal{M}_{\text{vertices}}$, edges $\mathcal{M}_{\text{edges}}$, and faces $\mathcal{M}_{\text{faces}}$. As a first step an iteration over all the edges of the element is performed, calculating the intersection points between the edge and the sphere. For this, the edge is expressed in terms of the two

vertices $\vec{v}_0$ and $\vec{v}_1$ forming it as $\vec{v}_0 + t \cdot (\vec{v}_1 - \vec{v}_0)$. Depending on the exact configuration zero, one, or two intersection points as solutions for t may be found. Initially, the case of the edge not intersecting the sphere at all or the sphere only touching the edge is treated and the edge is ignored. In the other case, either one or both vertices defining the edge are intersecting the sphere. If both intersection points lie on the edge, both vertices are marked. If the edge contains only one of the intersection points, the vertex closest to the center of the sphere is marked on line 13. If any vertex is marked at all, so is the edge itself. And if the edge is marked, so are the faces forming it (line 17). The next section of the algorithm performs a consistency check on the marked vertices: A vertex must only be marked if the three edges intersecting at this position are marked as well. This step is required to detect and eliminate spuriously marked vertices. Finally, the intersections between the sphere and the faces of the element have to be detected using the function `polygonSphereIntersection`. To comply with the previously stated goal of eliminating redundant intersection tests, only the inside of the polygon bounding the face but not the edges or corners must be tested. This concludes the algorithm to detect the intersecting primitives and the three sets of the marked entities are returned.

4.2. Volume of a general spherical wedge

As described in Sec. 3, the term general spherical wedge is used here to describe the volume cut out of a sphere by two intersecting planes forming two half-spaces, where the intersection line cuts the sphere. The planes are denoted p_0 and p_1 and defined in three-dimensional space by the points $\vec{c}_0$ and $\vec{c}_1$ contained in the planes and the normal vectors $\vec{n}_0$ and $\vec{n}_1$, respectively. For any such configuration of two intersecting planes, a third plane p_S can be found which is perpendicular to both p_0 and p_1 and contains the center point $\vec{c}_S$ of the sphere. The decomposition of the general spherical wedge into two sub-volumes will be described using the projection of the sphere and the planes p_0 and p_1 onto this plane (Fig. 6). The location of the points $\vec{c}_0$ and $\vec{c}_1$ relative to the intersection line along with the direction of the normals defines the general spherical wedge uniquely.

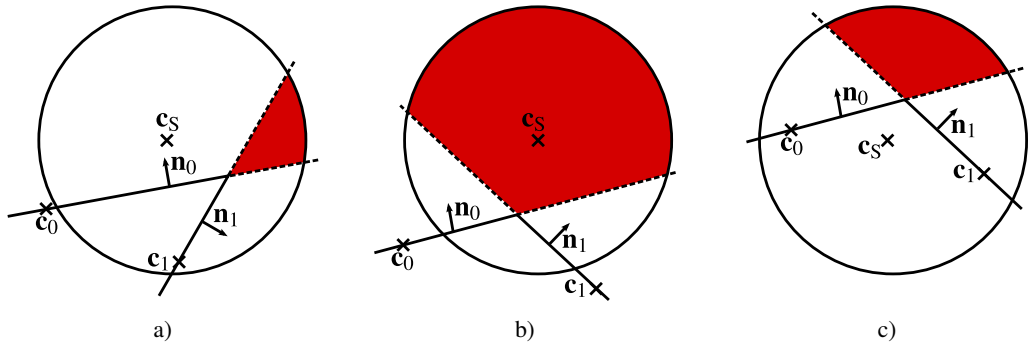


Figure 6: Three different cases of two-dimensional projections of the intersecting planes p_0 and p_1, each defined by a point in the plane and a normal vector, onto a plane p_S perpendicular to the intersection line containing the center $\vec{c}_S$ of the sphere. The marked section of the sphere is the projection of the general spherical wedge.

Depending on the relative position of the planes with respect to the center of the sphere, three different cases have to be identified and treated separately (Fig. 6 a–c). The volume of the general spherical wedge can be decomposed into a combination of simpler sub-volumes according to Fig. 7. The efficient identification of the correct case for this decomposition based on the geometrical configuration is presented in Algorithm 3.

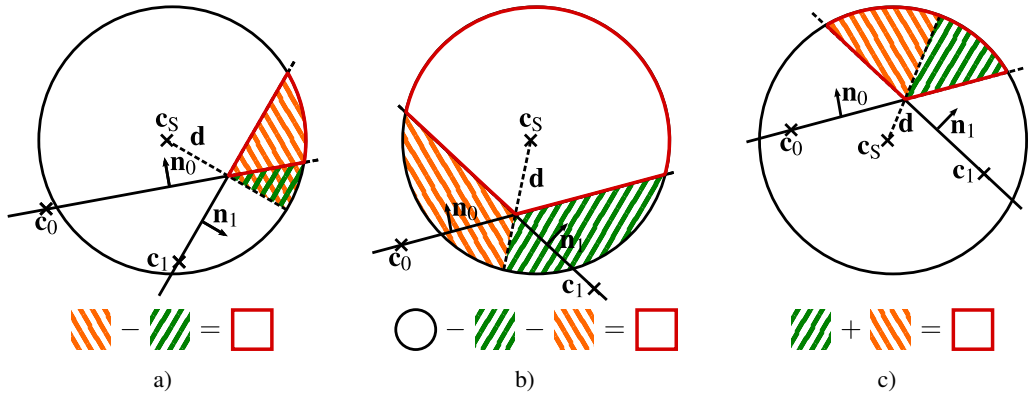


Figure 7: Sketch of the reduction of the volume calculation of a general spherical wedge to the calculation of the volumes of two regularized spherical wedges each defined by the radius r, the distance to the center d and an angle.

Algorithm 3 Calculation of the volume of a general spherical wedge defined by two planes using a decomposition into sub-volumes according to Fig. 7.

procedure GENERALWEDGE($\vec{c}_S$, r, $\vec{c}_0$, $\vec{n}_0$, $\vec{c}_1$, $\vec{n}_1$, $\vec{d}$)
Require: $r > 0$, $|\vec{d}| < r$, $\vec{c}_0 \neq \vec{c}_1$, $|\vec{n}_0| = |\vec{n}_1| = 1$
  if $|\vec{d}| < \varepsilon$ **then**
    $\alpha \leftarrow \pi - \text{ANGLE}(\vec{n}_0, \vec{n}_1)$
    return $\frac{2}{3}\alpha\, r^3$
  end if
  $s_0 \leftarrow \vec{d} \cdot \vec{n}_0$
  $s_1 \leftarrow \vec{d} \cdot \vec{n}_1$
  if $|s_0| < \varepsilon \vee |s_1| < \varepsilon$ **then**
    $\alpha \leftarrow \pi - \text{ANGLE}(\vec{n}_0, \vec{n}_1)$
    return REGULARIZEDWEDGE(r, $|\vec{d}|$, α, $|s_0| > |s_1|\ ?\ s_0 : s_1$)
  end if
  $\hat{\vec{d}} \leftarrow \vec{d}/|\vec{d}|$
  $d_0 \leftarrow \left(\vec{c}_S + \vec{d} - \vec{c}_0\right) \cdot \hat{\vec{d}}$
  $d_1 \leftarrow \left(\vec{c}_S + \vec{d} - \vec{c}_1\right) \cdot \hat{\vec{d}}$
  if $s_0 \geq 0 \wedge s_1 \geq 0$ **then** // Fig. 7c
    $\alpha_0 \leftarrow \frac{\pi}{2} - \text{COPYSIGN}(\text{ANGLE}(\vec{n}_0, \hat{\vec{d}}), d_0)$
    $\alpha_1 \leftarrow \frac{\pi}{2} - \text{COPYSIGN}(\text{ANGLE}(\vec{n}_1, \hat{\vec{d}}), d_1)$
    return REGULARIZEDWEDGE(r, $|\vec{d}|$, α_0, s_0) + REGULARIZEDWEDGE(r, $|\vec{d}|$, α_1, s_1)
  else if $s_0 < 0 \wedge s_1 < 0$ **then** // Fig. 7b
    $\alpha_0 \leftarrow \frac{\pi}{2} + \text{COPYSIGN}(1, d_0) \times [\text{ANGLE}(\vec{n}_0, \hat{\vec{d}}) - \pi]$
    $\alpha_1 \leftarrow \frac{\pi}{2} + \text{COPYSIGN}(1, d_1) \times [\text{ANGLE}(\vec{n}_1, \hat{\vec{d}}) - \pi]$
    return $\frac{4}{3}\pi\, r^3$ − [REGULARIZEDWEDGE(r, $|\vec{d}|$, α_0, $-s_0$) + REGULARIZEDWEDGE(r, $|\vec{d}|$, α_1, $-s_1$)]
  else // Fig. 7a
    $\alpha_0 \leftarrow \frac{\pi}{2} - \text{COPYSIGN}(1, d_0 \times s_0) \times [\text{ANGLE}(\vec{n}_0, \hat{\vec{d}}) - (s_0 < 0\ ?\ \pi : 0)]$
    $\alpha_1 \leftarrow \frac{\pi}{2} - \text{COPYSIGN}(1, d_1 \times s_1) \times [\text{ANGLE}(\vec{n}_1, \hat{\vec{d}}) - (s_1 < 0\ ?\ \pi : 0)]$
    $v_0 \leftarrow \text{REGULARIZEDWEDGE}(r, |\vec{d}|, \alpha_0, |s_0|)$
    $v_1 \leftarrow \text{REGULARIZEDWEDGE}(r, |\vec{d}|, \alpha_1, |s_1|)$
    return MAX(v_0, v_1) − MIN(v_0, v_1)
  end if
end procedure

The function `generalWedge` requires the center and radius of the sphere, the description of the two planes and a vector $\vec{d}$ as input. This vector $\vec{d}$ is the separation vector from the center of the sphere to the intersection point between the common edge of the intersecting planes p_0 and p_1 and the plane containing the center of the sphere while being perpendicular to this edge. This intersection point ($\vec{c}_S + \vec{d}$) is easily accessible as it lies just in the middle between the two intersection points of the line defined by the edge and the sphere. At the very beginning of the algorithm, the special case of $|\vec{d}| < \varepsilon$ is handled, where ε corresponds to the floating point precision of the machine. This special configuration can be treated as a common spherical wedge with the volume only depending on the radius of the sphere and the angle between the two planes. The algorithm makes use of three special functions, `angle`, `copysign`, and `regularizedWedge`. The first function performs a numerically robust calculation of the angle between two normalized vectors. The standard function `copysign` returns the absolute value of the first argument with the sign of the second argument. The last function `regularizedWedge` is used to process the simpler sub-volumes (Fig. 7) and is described in detail in the next section. Another noteworthy special case handled in lines 8–11 occurs when one of the planes touches the center of the sphere. This can be treated using a single call to `regularizedWedge`, no further subdivision of the volume is required.

4.3. Volume of a regularized spherical wedge

The volume of a general spherical wedge can be calculated in a single step, whenever at least one of the planes forming the wedge contains the center of the sphere (see Fig. 8). We term this geometric object a *regularized* spherical wedge and it is fully defined by the radius r of the sphere, the length of the separation vector from the center of the sphere d and the angle α.

Several analytical expressions for the volume of such regularized spherical wedges are available in the literature under various names [23–25]. They are often used in the context of calculating the intersection volume of overlapping spheres. As the accurate determination of the volume of a regularized spherical wedge lies at the very core of the whole overlap calculation, both computational efficiency and numerical stability are crucial. The previously proposed methods are not optimal in this regard. Some of the solutions [24, 25] lack numerical stability for values of α close to the limits of the allowed range, the version proposed by Wu et al. [23] is not optimized with respect to the computational cost. For the present implementation based on the work of Avis et al. [25], a different expression is derived by integrating the area of the projection of the spherical wedge. First, we consider the projection on the plane p_s, that is the plane perpendicular to the two planes defining the regularized spherical wedge that also contains the center of the sphere. The projection is sketched in Fig. 8, where the area of interest is marked. This area of interest can be

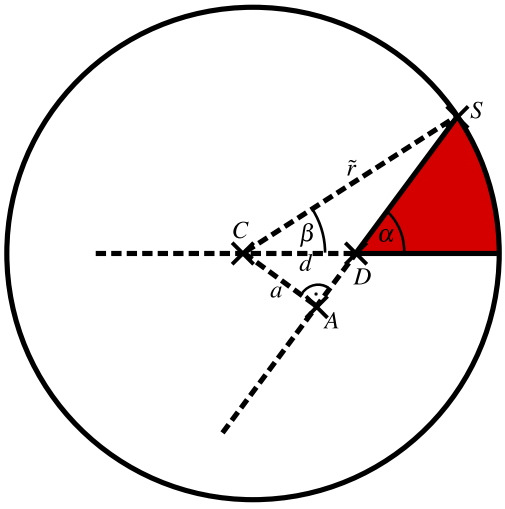


Figure 8: Projection of a regularized spherical wedge onto the plane perpendicular to the two planes defining the wedge and containing the center of the sphere. The highlighted area $A_{\lhd}(\tilde{r}, d, \alpha)$ has the shape of a generalized circular sector.

considered as a generalization of a circular sector, that is, a sector that does not touch the center of the circle. The

area $A_{\triangleleft}(\tilde{r}, d, \alpha)$ of this sector can be calculated by subtracting the area A_{CDS} of the triangle defined by the points C, D, and S from the area $A_{\tilde{r},\beta}$ of the sector defined by $\tilde{r}$ and the angle β:

$$A_{\triangleleft}(\tilde{r}, d, \alpha) = \frac{1}{2}\left[\alpha - \arcsin\left(\frac{a}{\tilde{r}}\right)\right]\tilde{r}^2 - \frac{a}{2}\left(\sqrt{\tilde{r}^2 - a^2} - \sqrt{d^2 - a^2}\right), \quad \text{with } a = d\,\sin(\alpha)\,. \tag{3}$$

In order to get the volume $V_{\triangleleft}(r, d, \alpha)$ of the regularized spherical wedge, the area $A_{\triangleleft}(\tilde{r}, d, \alpha)$ can now be integrated over the height of the sphere, that is, in the direction perpendicular to the projection plane used in Fig. 8. It should be noted that $\tilde{r}$ in Eq. 3 then becomes a function of the radius of the spherical wedge r and the height h i.e. the distance of the projection plane from the center of the sphere:

$$\begin{aligned}
V_{\triangleleft}(r, d, \alpha) &= \int_{-\sqrt{r^2-d^2}}^{\sqrt{r^2-d^2}} A_{\triangleleft}\left(\sqrt{r^2 - h^2}, d, \alpha\right) dh \\
&= \int_{0}^{\sqrt{r^2-d^2}} \left[\alpha - \arcsin\left(\frac{a}{\sqrt{r^2 - h^2}}\right)\right]\left(r^2 - h^2\right) - \frac{a}{2}\left(\sqrt{r^2 - h^2 - a^2} - \sqrt{d^2 - a^2}\right) dh \\
&= \frac{1}{3}\,a\,\sqrt{r^2 - d^2}\,\sqrt{d^2\left[1 - \sin^2(\alpha)\right]} + a\left(\frac{1}{3}\,a^2 - r^2\right)\arctan\left(\frac{\sqrt{r^2 - d^2}}{\sqrt{d^2 - a^2}}\right) + \\
&\quad \frac{2}{3}\,r^3 \arctan\left(\frac{a\,\sqrt{r^2 - d^2}}{r\,\sqrt{d^2 - a^2}}\right), \quad \text{with } a = d\,\sin(\alpha)\,.
\end{aligned} \tag{4}$$

By combining common terms and choosing mathematically equivalent but numerically more robust expressions for some terms, the volume of the spherical wedge can be calculated as:

$$\begin{aligned}
V_{\triangleleft}(r, d, \alpha) &= \frac{1}{3}\,a\,b\,c + a\left(\frac{1}{3}\,a^2 - r^2\right)\arctan\left(\frac{b}{c}\right) + \frac{2}{3}\,r^3 \arctan\left(\frac{b\,\sin(\alpha)}{r\,\cos(\alpha)}\right), \\
&\text{with } a = d\,\sin(\alpha), \quad b = \sqrt{|r^2 - d^2|}, \quad c = d\,\cos(\alpha)\,.
\end{aligned} \tag{5}$$

This expression was tuned for numerical robustness, so it not necessary to handle edge cases like $d \to 0$ explicitly, as long as the standard `atan2` function providing an increased numerical stability compared to `atan`, especially for angles close to $\frac{\pi}{2}$, is used. In contrast to the original equation, Eq. 5 requires the evaluatation of both the sine and cosine of α, however this can be done efficiently using the commonly available math function `sincos` and is automatically optimized by many compilers. As Eq. 5 is only valid for the case of $\alpha \in [0, \frac{\pi}{2}]$, for values of $\alpha \in (\frac{\pi}{2}, \pi]$ the volume has to be calculated as the volume of a spherical cap with height $h = r - d\,\sin(\alpha)$ minus the volume of a regularized wedge with an angle $\alpha' = \pi - \alpha$.

4.4. Handling of edge cases

Certain sections of Algorithm 1 are more susceptible for numerical errors than others. An especially critical step in the algorithm is the precise calculation of the volume of the "cone" formed by the edges intersecting at a vertex lying within the sphere (Figs. 4, 5). If the vertex is close to the surface of the sphere, the cone can degenerate in multiple ways. In case of the intersection points of the edges collapsing to a single point, the cone volume can be neglected altogether. More difficult to handle is the case of two of the three intersection points coinciding numerically. The resulting volume has the shape of a general spherical wedge, a case which has to be detected and handled correctly explicitly. Besides those extreme cases, especially for larger size ratios between the sphere and the mesh element, the distances from the vertex to the three intersection points of the edges and the sphere can be of different magnitudes, leading to a large numerical error in calculating the normal of the triangle formed by the intersection points. As the exact direction of this normal has a grave influence on the total numerical error, special care has to be taken. To limit the loss of precision in calculating this normal vector, those calculations are not performed using the standard 64 bit of double precision floating point variables. Instead an efficient software emulation of quadruple precision (128 bit) is employed [26]. As the numerically robust calculation of the normal vector requires only floating point additions and multiplications, both operations extremely cheap on modern CPUs, the overhead of this modification on the total volume calculation can be neglected.

4.5. Non-planar faces of mesh elements

While Algorithm 1 can be used not only for tetrahedra but also for wedges and convex hexahedra, one problem remains: Especially for wedges or hexahedra generated by common meshing tools, the faces formed by four vertices might not be perfectly planar. This means, these vertices might not be contained in a single plane by a tiny margin. While this is not problematic in most cases, for the overlap calculation even these very small differences may become significant. This is due to the fact that while some parts of the algorithm use the faces and the corresponding normals, other parts use the edges between them. Therefore, the calculations involving the spherical caps, the general spherical wedges, and the "cones" use slightly different geometries, leading to errors which become significant for larger ratios between the sizes of the mesh element and the sphere. In case such imperfect elements are to be handled, it may be advisable to subdivide the individual wedges or hexahedra into the corresponding tetrahedra [27] and perform sphere-tetrahedron overlap calculations.

5. Validation and benchmarking

With all the building blocks necessary to solve Eq. 2 at hand, the quality of the solution both in terms of numerical precision and computational efficiency is evaluated. Due to the nature of the solution using a decomposition into many sub-volumes, it is vital to test as many edge cases as possible to identify potential catastrophic failures of the algorithm. At the same time the computational performance should not be neglected, as any potential use case of the method requires a large number of calls to the overlap calculation routine, possibly in every time step.

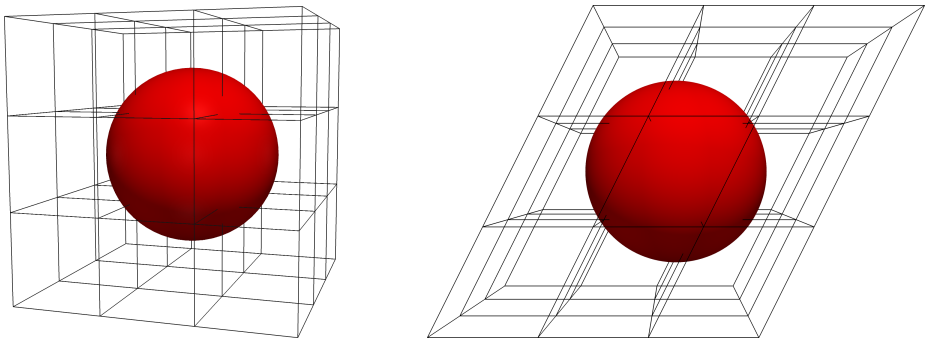
5.1. Validation

The first set of tests is used to validate the implementation of Algorithm 1 for hexahedra, wedges and tetrahedra. As it is only possible to calculate the overlap volume directly for very simple configurations involving carefully aligned regular mesh elements, the correct result is unknown a priori for less trivial setups. Thus a different approach is taken, using multiple mesh elements and a single, randomly placed sphere with varying radius.

The initial mesh consists of equally sized regular cuboids with an edge length of unity centered at the origin. The number of mesh elements is determined by the radius r of the sphere with $r \in [5 \times 10^{-2}, 2 \times 10^{1}]$. The radius of the sphere is varied within those limits using 40 logarithmically equally spaced intervals. For each realization, a mesh is generated covering at least the volume $[-(0.5 + r), 0.5 + r]^3$, leading to a minimum of 27 and a maximum of 68 921 hexahedra. The position of the sphere center is randomly chosen inside of the central element. An example configuration is depicted in Fig. 9a. For each value of the radius 100 000 sample points are generated, leading to a total of 4.1×10^6 test configurations. The chosen setup ensures that the sphere is always fully contained within the generated mesh. This allows the comparison of the combined overlap of the sphere with all elements of the mesh with the exact result, the full volume of the sphere. While this method does not provide a way to validate the overlap volume calculated for the individual mesh elements, due to the random placement and complexity of the involved calculations it is assumed that potentially occurring errors for single mesh elements do not cancel out, but lead to an overall erroneous result. It should be noted that for small radii there is a relatively high probability for the sphere to be fully contained within the central mesh element(s). The whole setup, however, corresponds to what would typically be encountered in a simulation, so no attempt is made to artificially modify the probability to encounter one of those special cases.

In order to also test the overlap calculation for wedges and tetrahedra, each element in the hexahedral mesh is divided into multiple sub-elements. For the mesh containing only wedge-shaped elements, each hexahedron is divided into two wedges. To achieve a maximum of test coverage created by different relative positions of the geometric objects, multiple tetrahedral meshes are constructed. The first two are simple decompositions of the hexahedral elements into five or six tetrahedra each, while preserving the vertex positions [27], resulting in 135 to 344 605 and 162 to 413 526 elements. To further stress the implementation, two additional refined meshes are generated by splitting each tetrahedron into four smaller tetrahedra. This gives a total of six meshes covering an identical volume with different individual elements.

For each test configuration and each mesh, the sum of the overlapping volume between the single mesh elements and the sphere is calculated and compared to the exact result, the full volume of the sphere. As the goal is to validate



a) b)

Figure 9: Sample meshes consisting of 27 hexahedra each with a unit sphere centered at the origin. The mesh depicted in sub-figure b is derived from the one in sub-figure a by shearing along one axis.

For the randomized tests the center of the sphere is chosen uniformly within $[-0.5, 0.5]^3$ and the number of mesh elements is adapted so the mesh always contains the sphere completely.

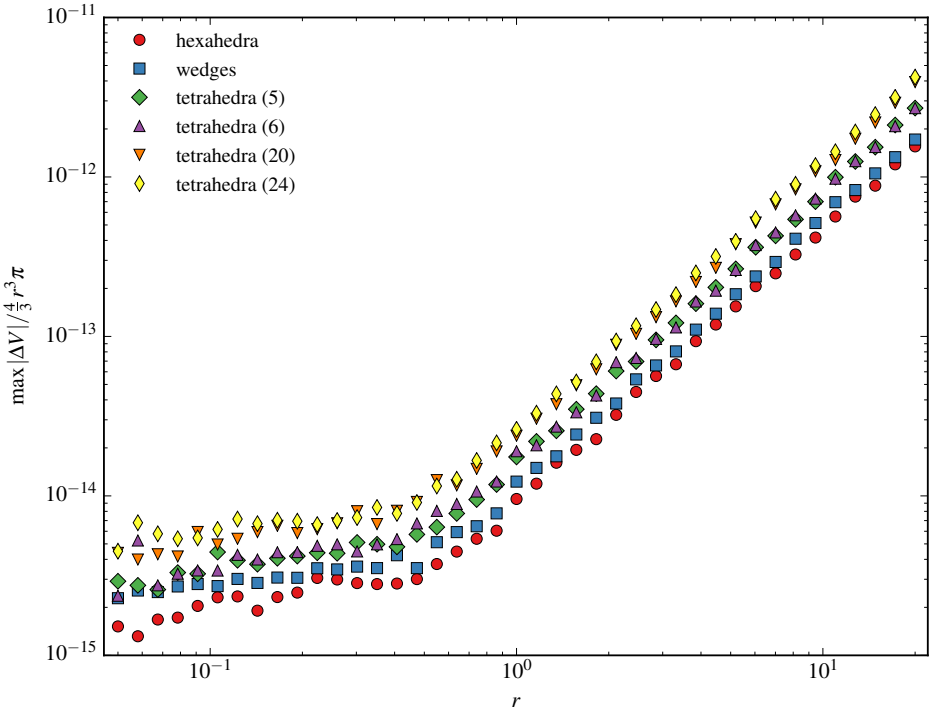


Figure 10: The maximum error in the total overlapping volume between a regular mesh composed of hexahedra, wedges, or tetrahedra and a sphere using the configuration depicted in Fig. 9a relative to the volume of the sphere. The individual elements of the hexahedral mesh are decomposed into 2 wedges or 5, 6, 20, or 24 tetrahedra to generate the other meshes.

the correctness of our method, the error over all realizations for each radius is recorded, corresponding to the worst-case behavior. The maximum of the absolute value of this error between the calculated overlap volume and the correct value relative to the volume of the sphere is depicted in Fig. 10. The error for all meshes clearly depends strongly on the radius, which is understandable from the way the overlap volume for the individual mesh elements is calculated: Following the algorithms detailed in Sec. 3, the overlap is determined by combining various sub-volumes of the sphere

outside of the mesh element. So, for combinations of a mesh element and a sphere with a size ratio strongly differing from unity, geometric entities of very different dimensions have to be handled. This leads to calculations involving values from a rather wide interval of floating point numbers, resulting in a general loss of precision. For small spheres, the maximum error forms a plateau determined by the machine precision ($\approx 10^{-16}$ for values near unity), while for larger radii, the effects of numerical errors become clearly visible. Nevertheless, the maximum relative error in the calculation is well below 10^{-11}, even for spheres 40 times as large as the mesh elements, showing the high quality of the described solution and implementation. For practical applications, the region of a size ratio of unity is most relevant, where the maximum error is of the order of 10^{-14}.

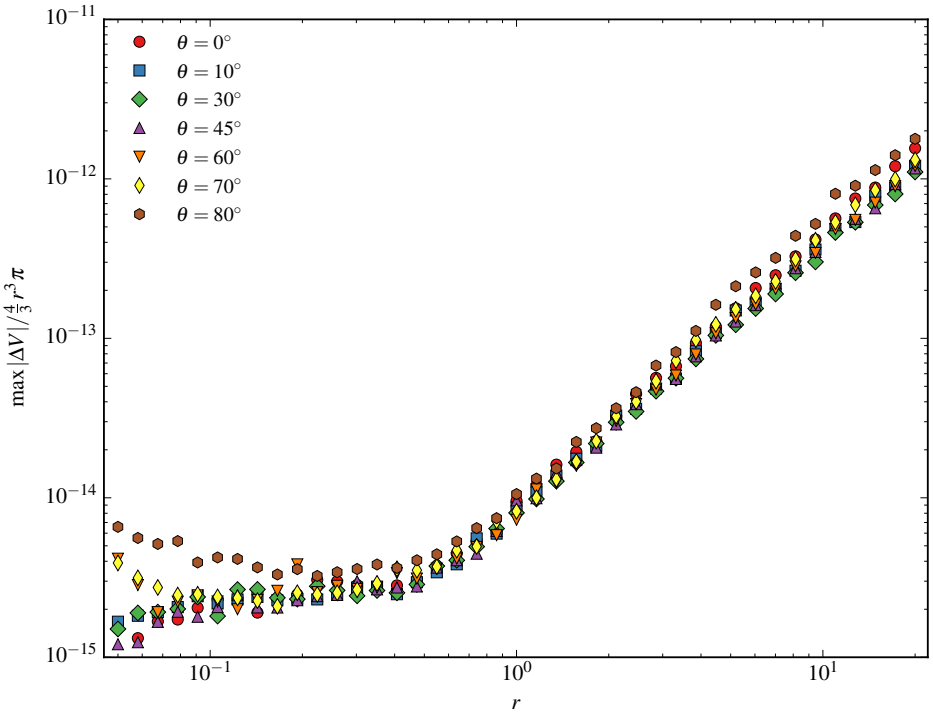


Figure 11: The maximum error in the total overlapping volume between a sheared mesh composed of hexahedra and a sphere using the configuration depicted in Fig. 9b relative to the volume of the sphere. The shearing angle θ is varied between 0 and $\frac{\pi}{2}$, showing the negligible dependency of the error on the angle especially for size ratios of the order of unity. Only a subset of the values for the examined angles $\theta \in \left(0, \frac{\pi}{4}\right]$ are plotted, as the remaining lines collapse for these cases.

The second test setup uses a configuration almost identical to the one described above with the difference of an additional shearing of the original hexahedral mesh in one direction (Fig. 9b). This leads to angles between the sides of a single hexahedron other than $\frac{\pi}{2}$, aiming to uncover any problems with acute and obtuse angles in the handling of hexahedra, while for the other mesh types the more extreme angles occurring serve as an additional stress test. A set of 10 different shearing angles $\theta \in \left[0, \frac{\pi}{2}\right)$ are tested, with $\theta = 0$ corresponding to no shearing, that is, the case examined in the previous test configuration. The angle is incremented in steps of 10°, and the special case of $\theta = \frac{\pi}{4}$ is added to the set of tested parameters.

The results for the maximum error for all six meshes in this test case are comparable to the ones obtained for the undeformed meshes. The qualitative behavior is the same for all types of mesh elements, so in Fig. 11 only the results for the sheared hexahedral mesh are shown. For small size ratios, that is, $r \in [5 \times 10^{-2}, 2 \times 10^{-1}]$ and large shearing angles ($\theta > \frac{\pi}{4}$), the maximum error is larger but of the same order of magnitude, whereas no effect is observable for smaller angles. For arbitrary angles above the relative radius of $r = 2 \times 10^{-1}$ the deviation is not significant. So a quantitative difference can only be identified for small size ratios and large shearing angles. From this we conclude that the additional shearing does not lead to a significant increase in the error for any of the examined

meshes. With this, the presented implementation is considered numerically robust independent of the exact shape of the elements. Of course, degenerated mesh elements can still lead to larger errors, but any simulation is vulnerable to such configurations and a lot of effort is typically put into avoiding those situations in the first place.

5.2. Benchmarking

After verifying the basic correctness of the implementation in the previous section, the computational performance of the reference implementation of Algorithm 1 is examined next. All benchmark runs are preformed on a typical workstation with an Intel Core i7-4771 CPU running at a base frequency of 3.5 GHz. The code is compiled using the GNU C++ compiler in version 4.9.3.

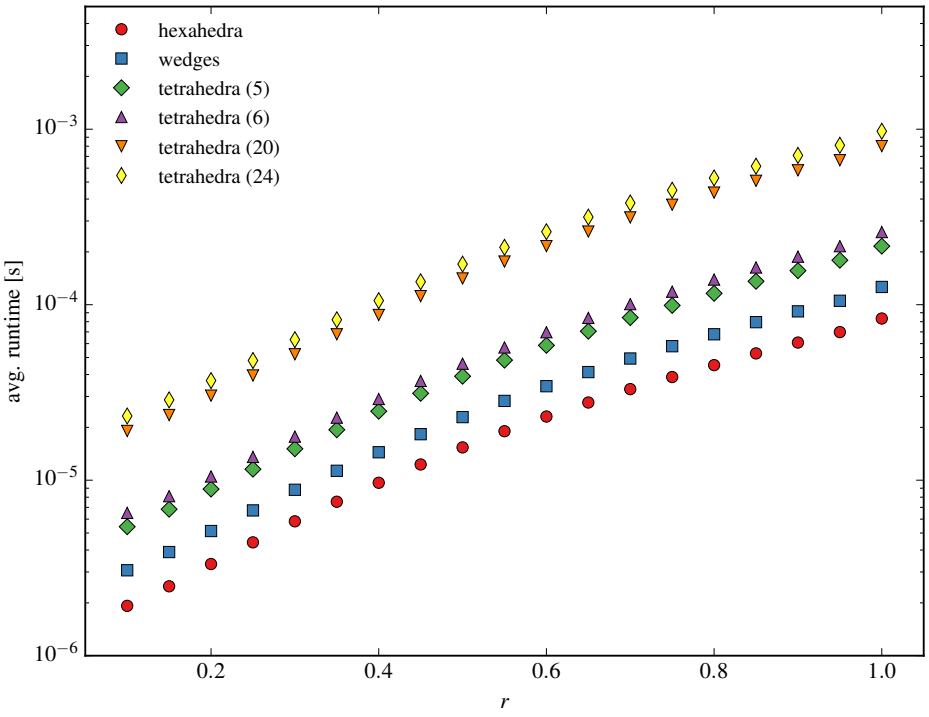


Figure 12: The influence of the relative size ratio on the average runtime to calculate the overlap volume between a sphere of radius r and a regular mesh of hexahedra, wedges, or tetrahedra using the same configuration as in Fig. 10. The values given are the averaged wall-clock times as measured on the test system, an Intel Core i7-4771 CPU.

The setup used for the benchmarks is identical to the configuration used for the validation in Sec. 5.1. Again, a sphere of varying radius $r \in [0.1, 1.0]$ is placed in the central unit hexahedron and the overlapping volume with all elements in the mesh is calculated. The time for the evaluation of the total overlap volume is recorded and averaged over 10 000 samples. This configuration is representative for systems where the size of the spherical particles is below the average size of the mesh elements. The average run time for the different mesh types (hexahedra, wedges, and tetrahedra) over the size ratio is plotted in Fig. 12. Of course, both the radius of the sphere and the specific mesh have a significant influence on the average runtime. Both effects are easily understood when considering the number of partial overlaps for each configuration: The relative size ratio determines how many elements are intersecting the sphere. For very small spheres, it is likely that the sphere is fully contained within a single element, allowing the handling of this special case in Algorithm 1 to dramatically reduce the computation time. This case, however, becomes highly unlikely or even impossible as the sphere grows relative to the size of the mesh elements, leading to the strong increase in the average runtime. It should be noted that the differences in runtime for different meshes at equal size ratio are not an effect of the element geometry, but rather the fact that for the meshes generated by decomposition a larger number of mesh elements overlap the sphere.

The timing results show that the performance of the exact method makes it a viable candidate for typical application scenarios. For a ratio between the side length of the hexahedra and the sphere diameter of $\frac{1}{0.6} \approx 1.7$, the calculation of the overlap volume for a single sphere takes about 5.8×10^{-6} seconds on average on our test system. This is near the lower limit of the size ratio as recommended for typical CFD–DEM coupling schemes [10]. Assuming a system size of 45 000 particles, a figure common for coupled CFD–DEM simulations [8], this would lead to an average runtime of the exact volume calculation of less than 0.3 seconds on a single core of the CPU used for our benchmarks. This is clearly an acceptable value for most coupling schemes and can be further reduced by exploiting the inherent parallelism of the problem.

6. Conclusions

Extending previously published works [22, 23] an algorithm is described to calculate the exact overlap volume between a sphere and a mesh element with the shape of a tetrahedron, wedge, or hexahedron. The presented approach is specifically tuned for numerical robustness and common pitfalls and ways to avoid those in an implementation are discussed. A thorough validation using randomized configurations is performed, showing the high quality of the solution over a wide range of size ratios. Benchmarks of customary configurations highlight the benefits of the exact solution, providing a precise result while maintaining the high computational performance.

The full C++ source code for the exact calculation of the overlap volume of a sphere and a mesh element is available under the terms of the GNU General Public License version 3 at https://github.com/severinstrobl/overlap.

Acknowledgements

The authors would like to acknowledge the funding of the Deutsche Forschungsgemeinschaft (DFG) through the Cluster of Excellence Engineering of Advanced Materials. The work was partly executed at the *Centro de Investigación, Transferencia e Innovación* (CITI), Ourense, Spain.

[1] J. A. M. Kuipers, K. van Duin, F. van Beckum, W. van Swaaij, A numerical model of gas-fluidized beds, Chemical Engineering Science 47 (8) (1992) 1913–1924. doi:10.1016/0009-2509(92)80309-Z.

[2] Y. Tsuji, T. Kawaguchi, T. Tanaka, Discrete particle simulation of two-dimensional fluidized bed, Powder Technology 77 (1) (1993) 79–87. doi:10.1016/0032-5910(93)85010-7.

[3] B. P. B. Hoomans, J. A. M. Kuipers, W. J. Briels, W. P. M. van Swaaij, Discrete particle simulation of bubble and slug formation in a two-dimensional gas-fluidised bed: A hard-sphere approach, Chemical Engineering Science 51 (1) (1996) 99–118. doi:10.1016/0009-2509(95)00271-5.

[4] N. G. Deen, M. Van Sint Annaland, M. A. Van der Hoef, J. A. M. Kuipers, Review of discrete particle modeling of fluidized beds, Chemical Engineering Science 62 (12) (2007) 28–44. doi:10.1016/j.ces.2006.08.014.

[5] C. L. Wu, A. S. Berrouk, K. Nandakumar, Three-dimensional discrete particle model for gas–solid fluidized beds on unstructured mesh, Chemical Engineering Journal 152 (23) (2009) 514–529. doi:10.1016/j.cej.2009.05.024.

[6] S. Ergun, Fluid flow through packed columns, Chemical Engineering Progress 48 (1952) 89–94.

[7] C. Y. Wen, Y. H. Yu, Mechanics of fluidization, Chemical Engineering Progress Symposium Series 62 (1966) 100–111.

[8] L. Fries, S. Antonyuk, S. Heinrich, S. Palzer, DEM–CFD modeling of a fluidized bed spray granulator, Chemical Engineering Science 66 (11) (2011) 2340–2355. doi:10.1016/j.ces.2011.02.038.

[9] C. M. Boyce, D. J. Holland, S. A. Scott, J. S. Dennis, Novel fluid grid and voidage calculation techniques for a discrete element model of a 3D cylindrical fluidized bed, Computers & Chemical Engineering 65 (0) (2014) 18–27. doi:10.1016/j.compchemeng.2014.02.019.

[10] Z. Peng, E. Doroodchi, C. Luo, B. Moghtaderi, Influence of void fraction calculation on fidelity of CFD-DEM simulation of gas-solid bubbling fluidized beds, AIChE Journal 60 (6) (2014) 2000–2018. doi:10.1002/aic.14421.

[11] P. Müller, A. Formella, T. Pöschel, Granular jet impact: probing the ideal fluid description, Journal of Fluid Mechanics 751 (2014) 601–626. doi:10.1017/jfm.2014.210.

[12] I. Goldhirsch, Stress, stress asymmetry and couple stress: from discrete particles to continuous fields, Granular Matter 12 (3) (2010) 239–252. doi:10.1007/s10035-010-0181-z.

[13] A. Ries, L. Brendel, D. E. Wolf, Coarse graining strategies at walls, Computational Particle Mechanics 1 (2) (2014) 177–190. doi:10.1007/s40571-014-0023-6.

[14] A. Donev, A. L. Garcia, B. J. Alder, Stochastic event-driven molecular dynamics, J. Comput. Phys. 227 (4) (2008) 2644–2665. doi:10.1016/j.jcp.2007.11.010.

[15] E. W. C. Lim, C.-H. Wang, A.-B. Yu, Discrete element simulation for pneumatic conveying of granular material, AIChE Journal 52 (2) (2006) 496–509. doi:10.1002/aic.10645.

[16] A. Tomiyama, I. Žun, H. Higaki, Y. Makino, T. Sakaguchi, A three-dimensional particle tracking method for bubbly flow simulation, Nuclear Engineering and Design 175 (12) (1997) 77–86. doi:10.1016/S0029-5493(97)00164-7.

[17] D. Darmana, N. G. Deen, J. A. M. Kuipers, Detailed modeling of hydrodynamics, mass transfer and chemical reactions in a bubble column using a discrete bubble model, Chemical Engineering Science 60 (12) (2005) 3383–3404. doi:10.1016/j.ces.2005.01.025.
[18] H. A. Khawaja, S. A. Scott, M. S. Virk, M. Moatamedi, Quantitative analysis of accuracy of voidage computations in CFD-DEM simulations, The Journal of Computational Multiphase Flows 4 (2) (2012) 183–192. doi:10.1260/1757-482X.4.2.183.
[19] B. Freireich, M. Kodam, C. Wassgren, An exact method for determining local solid fractions in discrete element method simulations, AIChE Journal 56 (12) (2010) 3036–3048. doi:10.1002/aic.12223.
[20] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Numerical integration of implicit functions for the initialization of the VOF function, Computers & Fluids 113 (2015) 42–52. doi:10.1016/j.compfluid.2014.04.010.
[21] W. A. Richards, The common content of a sphere and a regular tetrahedron having a common centre, International Journal of Mathematical Education in Science and Technology 26 (2) (1995) 257–266. doi:10.1080/0020739950260211.
[22] F. Bernardeau, R. van de Weygaert, A new method for accurate estimation of velocity field statistics, Monthly Notices of the Royal Astronomical Society 279 (2) (1996) 693–711. doi:10.1093/mnras/279.2.693.
[23] C. L. Wu, J. M. Zhan, Y. S. Li, K. S. Lam, A. S. Berrouk, Accurate void fraction calculation for three-dimensional discrete particle model on unstructured mesh, Chemical Engineering Science 64 (6) (2009) 1260–1266. doi:10.1016/j.ces.2008.11.014.
[24] K. D. Gibson, H. A. Scheraga, Exact calculation of the volume and surface area of fused hard-sphere molecules with unequal atomic radii, Molecular Physics 62 (5) (1987) 1247–1265. doi:10.1080/00268978700102951.
[25] D. Avis, B. K. Bhattacharya, H. Imai, Computing the volume of the union of spheres, The Visual Computer 3 (6) (1988) 323–328. doi:10.1007/BF01901190.
[26] T. J. Dekker, A floating-point technique for extending the available precision, Numerische Mathematik 18 (3) (1971) 224–242. doi:10.1007/BF01397083.
[27] J. Dompierre, P. Labbé, M.-G. Vallet, R. Camarero, How to subdivide pyramids, prisms, and hexahedra into tetrahedra, in: Proceedings of the 8th International Meshing Roundtable, South Lake Tahoe, CA, U.S.A., 1999, pp. 195–204.