Meshing molecular surfaces based on analytical implicit representation

Chaoyu Quan, Benjamin Stamm

Meshing Molecular Surfaces Based on Analytical Implicit Representation

Chaoyu Quan[1,2] and Benjamin Stamm*[3,4]

[1]Sorbonne Universités, UPMC Univ Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France
[2]CNRS, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France
[3]Center for Computational Engineering Science, RWTH Aachen University, Aachen, Germany
[4]Computational Biomedicine, Institute for Advanced Simulation IAS-5 and Institute of Neuroscience and Medicine INM-9, Forschungszentrum Jülich, Germany

Abstract

We develop an algorithm for meshing molecular surfaces that is based on patch-wise meshing using an advancing-front method adapted to the particular case of molecular surfaces. We focus on the solvent accessible surface (SAS) and the solvent excluded surface (SES). The essential ingredient is a newly developed analysis of such surfaces in *Ch. Quan, B. Stamm, Mathematical Analysis and Calculation of Molecular Surfaces, Journal of Computational Physics, 322:760-782, 2016* that allows to describe all SES-singularities a priori and therefore a complete characterization of the SES. In addition, an algorithm for filling molecular inner holes is proposed based on the pre-computed data structures of molecular surfaces.

Keywords: Molecular Surface, Solvent Excluded Surface, Molecular Visualization, Implicit Representation, Advancing-front Method, Meshing Algorithm

1 Introduction

Many fields of research like chemistry, biochemistry, physics and biomedicine work with molecular surfaces. For example, the majority of (bio-)chemically relevant reactions take place in the liquid phase and the effect of the environment (solvent) is important and should be considered in one way or the other in the model. As an alternative to the way of taking solvent molecules explicitly into account, various implicit solvation models have been proposed in which the molecular surface of the solute is a part of the model and constitutes the interface of the atomistic and the continuum model [1]. A second field where the notion of molecular surface is important is simply the visualization of molecules.

In the simplest model of a molecular surface of the solute or a molecule in general, each constituting atom is idealized by a simple sphere with its Van der Waals (VdW) radius. The boundary of the union of these VdW spheres is the so-called VdW-surface.

*Corresponding Author. E-mail address: best@mathcces.rwth-aachen.de (B. Stamm)
Submitted to Journal of Molecular Graphics and Modelling, 23 September 2016

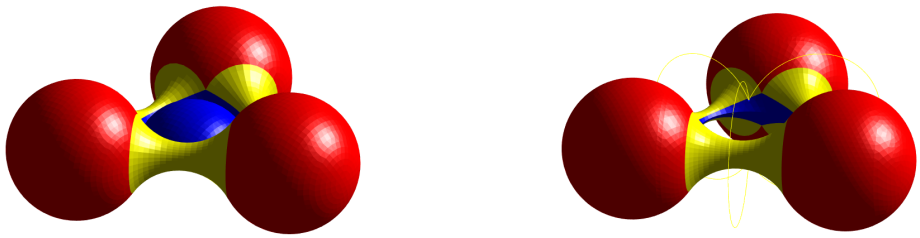

Figure 1: The SES of a three atomic system with the self-intersection problem between two concave spherical patches (left, triangulation provided by the MSMS-algorithm) and the SES with a singular circle on the concave spherical patches (right, triangulation provided by our algorithm).

Besides the VdW-surface, two other kinds of molecular surfaces are commonly used in solvation models or in molecular visualization: the Solvent Accessible Surface (SAS) and the Solvent Excluded Surface (SES) [2]. Since the VdW-surface and the SAS are both the topological boundary of the union of spheres, their geometric features are therefore easier to understand. However, the SES, which sometimes performs better in a chemical calculation [3] or is more suitable for applications like docking [4], is more complicated.

1.1 Previous Works

The definitions of the SAS and the SES were first introduced by Lee & Richards [5, 6] in the 1970s. The SES is also called the "smooth molecular surfaces" or "Connolly's surfaces" due to Connolly's fundamental calculation on it [2]. Indeed, the SES can be considered to be the prototype for the computational study of molecular surfaces. This surface model has been applied to a very large variety of problems and has also been used to compute solvation energies with continuum solvation models [1, 3, 7].

Latter, Michel Sanner developed the reduced surfaces and proposed the MSMS (Michel Sanner's Molecular Surface) algorithm for computing an (in fact approximately) analytical representation of the SES [8]. The MSMS algorithm can also provide a triangulation of the SES with a user-specified density of vertices. Although it can not deal with all self-intersections between different patches of the SES (see Figure 1 for an example where self-intersection occurs), the MSMS algorithm is now one of the most widely-used packages of molecular surfaces.

Besides, there are many other contributions on the molecular visualization [9, 10, 11, 12, 13], high-quality meshing [14] or the calculation of molecular areas and volumes [15, 16, 17]. However, a computable analytical implicit representation of the SES and a complete characterization of all SES-singularities remained unsolved until a recently-published paper by us [18].

1.2 Contribution

We give a detailed strategy for constructing the data structures of molecular surfaces, based on the analytical characterization of the SES including its singularities presented in [18]. Then, a meshing algorithm for molecular surfaces, especially the SES, is developed, combining an advancing-front algorithm with the pre-computed data structures. The explicit characterization of all singularities resolves the issue of self-intersection that is experienced due to singularities as they can be computed

prior to the meshing of the surface. This, in turn, allows the possibilities of meshing the SES exactly, in the sense that each vertex of the mesh lies exactly on the surface. We want to emphasize once again that this is only possible due to the newly developed analysis of the molecular surfaces and it is not the case for the existing meshing algorithms. In addition, we propose an algorithm for filling molecular inner holes with virtual atoms for the reason that the appearance of these inner holes is not always justified in the solute molecular cavity of the continuum solvation models, as this would mean that the solvent is present in these inner holes.

1.3 Outline

In the next section, we give the definitions and the implicit representations of different molecular surfaces. In the third section, the construction of molecular surfaces, which is defined by different patches and their connectivity, is proposed. Based on this pre-computed data, an algorithm for filling molecular inner holes is proposed in the fourth section. Further, a meshing algorithm for molecular surfaces including two sub-algorithms for meshing respectively a (convex or concave) spherical patch and a toroidal patch is developed in the fifth section where we also present some illustrations of artificial as well as realistic molecular surfaces. Finally, a conclusion is presented in the last section.

2 Molecular Surfaces

A mathematical analysis and calculation of the SAS and the SES has been presented in our recent work [18]. In this section, we recall some results including a mathematical definition of the surfaces, their implicit representations and the complete characterization of the SES.

2.1 Definitions

As already emphasized, atoms of a molecule can be represented by VdW-balls with VdW-radii which are experimentally fitted, given the underlying chemical element [19]. In consequence, the VdW-surface is defined as the topological boundary of the union of all VdW-balls. Further, the SAS of a solute molecule is defined by rolling the center of an idealized spherical probe over the solute molecule, that is, the surface enclosing the region in which the center of a spherical probe can not enter. Finally, the SES is defined by the same spherical probe rolling over the molecule, but now we consider the surface enclosing the region in which a spherical probe can not access. In other words, the SES is the boundary of the union of all spherical probes that do not intersect the VdW-balls of the solute molecule, see Figure 2 for a graphical illustration.

The definition of VdW-surface is based on the model that each atom has a specific radius around the atom center. However, the definition of the VdW-surface has ignored the size and shape of the surrounding solvent molecules in solvation models. The definition of SAS has taken this into account by modeling them by idealized spherical probes with a certain probe radius. The definition of the SES is different from the SAS in the sense that not the probe center traces out the desired surface, but the surface of the probe. In the application of the above-mentioned docking [4], the SES will not lead to the overlapping of neighboring surfaces since the SES does not inflate the atom radii but the SAS will.

Sometimes, the SAS can be non-connected: it can be composed of several separate surfaces. We call the outmost surface as the exterior Solvent Accessible Surface (eSAS) and the union of all

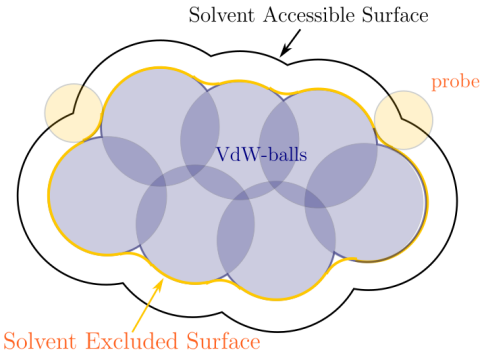

Figure 2: 2-dimension (2D) schematics of the Solvent Accessible Surface and the Solvent Excluded Surface, both defined by a spherical probe in orange rolling over the molecular VdW-atoms in dark blue.

separated surfaces as the complete Solvent Accessible Surface (cSAS), see Figure 3 for an example and [18] for details. Correspondingly, we also propose the concept of the complete Solvent Excluded Surface (cSES) and the exterior Solvent Excluded Surface (eSES). We make a convention that the SAS refers to both the cSAS and the eSAS in a general context, and the SES refers in the same spirit to both the cSES and the eSES.

Both the VdW-surface and the SAS are composed of three parts: open spherical patches, open circular arcs (or circles) and intersection points (formed by the intersection of three or more spheres). The SES can be divided into three corresponding types of patches [2]: convex spherical patches, toroidal patches and concave spherical patches, see Figure 4 for a 3D illustration. As showed in [18], any point on a convex spherical patch of the SES has a closest point to the SAS on a spherical patch. Similarly, any point on a toroidal patch of the SES has a closest point to the SAS on a circular arc, and any point on a concave patch has a closest point to the SAS which is an intersection point.

2.2 Implicit Representations

We denote by M the number of atoms in a solute molecule, by $c_i \in \mathbb{R}^3$ and $r_i \in \mathbb{R}^+$ the center and the radius of the i-th VdW atom. The open ball with center c_i and radius r_i is called the i-th VdW-ball. The VdW-surface can consequently be represented as an implicit surface $\widetilde{f}_{\text{vdw}}^{-1}(0) = \{p \in \mathbb{R}^3 | \widetilde{f}_{\text{vdw}}(p) = 0\}$ with the following implicit function:

$$\widetilde{f}_{\text{vdw}}(p) = \min_{i=1,\dots,M} \{\|p - c_i\|_2 - r_i\}, \quad \forall p \in \mathbb{R}^3. \tag{2.1}$$

Similarly, the open ball with center c_i and radius $r_i + r_{\text{p}}$ is called the i-th SAS-ball denoted by B_i, where r_{p} is the radius of the idealized spherical probe. Furthermore, we denote by S_i the i-th SAS-sphere corresponding to B_i, that is, $S_i = \partial B_i$. Similar to the VdW-surface, the SAS can be represented as an implicit surface $\widetilde{f}_{\text{sas}}^{-1}(0)$ with the following implicit function:

$$\widetilde{f}_{\text{sas}}(p) = f_{\text{vdw}}(p) - r_{\text{p}} = \min_{i=1,\dots,M} \{\|p - c_i\|_2 - r_i - r_{\text{p}}\}, \quad \forall p \in \mathbb{R}^3. \tag{2.2}$$

We notice that the above implicit function of the SAS is simple to compute. It seems nevertheless hopeless for us to further obtain an implicit function of the SES for the reason that $\widetilde{f}_{\text{sas}}(p)$ is not

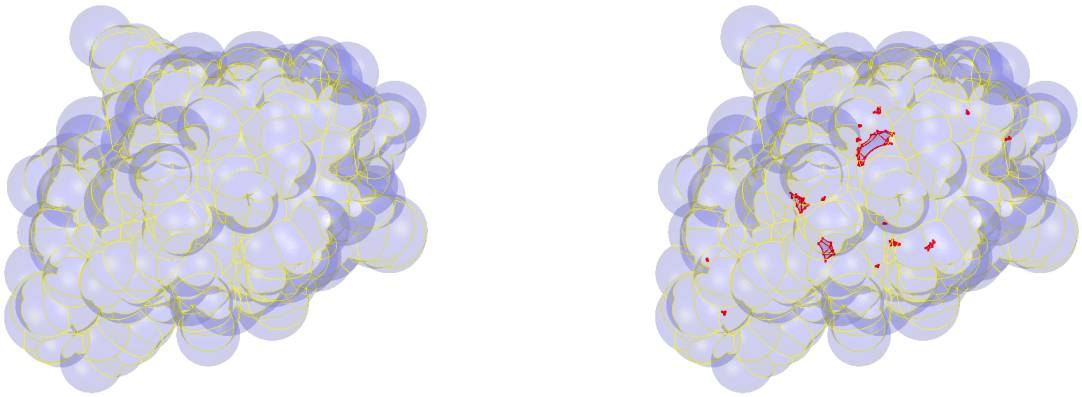

Figure 3: The transparent eSAS (left) and the transparent cSAS (right) of 1B17 with 485 atoms where the probe radius $r_{\mathrm{p}} = 1$Å. The boundary of each exterior spherical patch is constituted by circular arcs in yellow while the boundary of each interior spherical patch is constituted by circular arcs in red.

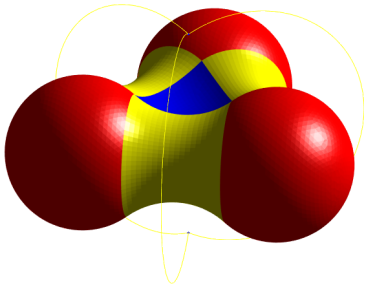

Figure 4: 3D schematic of the SES illustrating the convex spherical patches (red), the toroidal patches (yellow) and the concave spherical patches (blue).

a distance function to the SAS. On the other hand, having the signed distance function at hand would allow the construction of an implicit function for the SES due to the geometrical relationship between the SAS and the SES, i.e., they are separated by the fixed distance r_{p}.

In [18], we calculated the signed distance function to the SAS. Indeed, since the SAS is a closed surface, there exists a closest point on the SAS to any given point $p \in \mathbb{R}^3$, which is denoted by x^p_{sas} depending on p. We emphasize that there might exist more than one closest point to p and in this case, x^p_{sas} is chosen to be one of these closest points. The signed distance function $f_{\mathrm{sas}}(p)$ can then be easily written as:

$$f_{\mathrm{sas}}(p) = \begin{cases} -\|p - x^p_{\mathrm{sas}}\| & \text{if } p \text{ lies inside the SAS,} \\ \|p - x^p_{\mathrm{sas}}\| & \text{if } p \text{ lies outside the SAS.} \end{cases} \tag{2.3}$$

The difficulty is however to find efficiently one closest point x^p_{sas}. In the Chapter 6 & 7 of [20], one can find a very detailed and general discussion about the properties of the signed (or more generally speaking, oriented) distance function.

According to the fact that any point on the SES has signed distance $-r_{\mathrm{p}}$ to the SAS, an implicit function of the SES is obtained directly as:

$$f_{\mathrm{ses}}(p) = f_{\mathrm{sas}}(p) + r_{\mathrm{p}}, \tag{2.4}$$

which motivates the choice of using the signed distance function denoted by $f_{\mathrm{sas}}(p)$ to represent the SAS. From the above formula, the SES can be represented by a level set $f^{-1}_{\mathrm{sas}}(-r_{\mathrm{p}})$, associated with the signed distance function f_{sas} to the SAS. Therefore, the key point becomes how to compute the signed distance $f_{\mathrm{sas}}(p)$ from a point $p \in \mathbb{R}^3$ to the SAS. Generally speaking, given a general surface $S \subset \mathbb{R}^3$ and any arbitrary point $p \in \mathbb{R}^3$, it is difficult or expensive to compute the signed distance from p to S. The fast marching method [21] and the fast sweeping method [22] are two famous methods for computing such signed distance. However, considering that the SAS is a special surface composed of spherical patches, this computation can be done analytically [18].

Further, the region enclosed by the VdW-surface is called the VdW-cavity, that is, any point p in the VdW-cavity satisfies $f_{\mathrm{vdw}}(p) \leq 0$. More generally, the region enclosed by a molecular surface is called by its corresponding molecular cavity. In consequence, the region enclosed by the SAS is called the SAS-cavity, and the region enclosed by the SES is called the SES-cavity. Similarly, any point p in the SAS-cavity satisfies $f_{\mathrm{sas}}(p) \leq 0$, and any point p in the SES-cavity satisfies $f_{\mathrm{ses}}(p) \leq 0$.

2.3 Complete Characterization of the SES

In this subsection, we present the main results about the complete characterization of the SES in [18]. For any point x_{sas} on the SAS, we define a mapping by

$$\mathcal{R}(x_{\mathrm{sas}}) = \left\{ p \in \overline{\Omega} \mid \mathrm{dist}(p, \Gamma_{\mathrm{sas}}) = |p - x_{\mathrm{sas}}| \right\},$$

where $\overline{\Omega}$ is the SAS-cavity, Γ_{sas} is the SAS and $\mathrm{dist}(p, \Gamma_{\mathrm{sas}})$ denotes the distance from p to Γ_{sas}. Consequently, we have the following theorem according to [18]:

Theorem 2.1. *The following equivalence statements hold:*

[1] *if x_{sas} lies on a spherical patch of the SAS, part of the sphere S_i, then $\mathcal{R}(x_{\mathrm{sas}}) = [c_i, x_{\mathrm{sas}}]$ is a line segment.*

[2] *if $x_{\rm sas}$ lies on a circular arc of the SAS, part of the intersection circle $S_i \bigcap S_j$, then $\mathcal{R}(x_{\rm sas}) = \triangle x_{\rm sas} c_i c_j$ is a triangle.*

[3] *if $x_{\rm sas}$ is an intersection point of the SAS, then $\mathcal{R}(x_{\rm sas})$ is a polyhedron.*

Remark 2.1. *The first equivalence in the above theorem states that in the case where $x_{\rm sas}$ lies on a spherical patch of the SAS associated with S_i, $x_{\rm sas}$ is a closest point to an arbitrary point $p \in \overline{\Omega}$ if and only if p lies on the line segment with endpoints c_i and $x_{\rm sas}$. The second equivalence states that in the case where $x_{\rm sas}$ lies on a circular arc of the SAS associated with S_i and S_j, $x_{\rm sas}$ is a closest point to an arbitrary point $p \in \overline{\Omega}$ if and only if p lies on the triangle with three vertices c_i, c_j and $x_{\rm sas}$.*

We can generalize the mapping $\mathcal{R}$ by

$$\mathcal{R}(X) = \bigcup_{x \in X} \mathcal{R}(x), \quad \forall X \subseteq \Gamma_{\rm sas}.$$

In consequence, for a spherical SAS patch denoted by P_{i_1} with index i_1, its corresponding convex SES patch P_+ can be written as $P_+ = \mathcal{R}(P_{i_1}) \bigcap \Gamma_{\rm ses}$ where $\Gamma_{\rm ses}$ is the SES; for a circular SAS arc denoted by l_{i_2} with index i_2, its corresponding toroidal SES patch P_t can be written as $P_t = \mathcal{R}(l_{i_2}) \bigcap \Gamma_{\rm ses}$; for an SAS intersection point denoted by x_{i_3} with index i_3, its corresponding concave SES patch P_- can be written as $P_- = \mathcal{R}(x_{i_3}) \bigcap \Gamma_{\rm ses}$. In Figure 4, for instance, P_+ is a red convex patch, P_t is a yellow toroidal patch and P_- is a blue concave patch. Further, according to Theorem 2.1, we know that $\mathcal{R}(P_{i_1})$ is a spherical sector, $\mathcal{R}(l_{i_2})$ is a double-cone region and $\mathcal{R}(x_{i_3})$ is a polyhedron.

There is no difficulty to compute the convex SES patches or the toroidal SES patches given all spherical SAS patches and circular SAS arcs, see details in [2]. However, the self-intersection problem might occur among the concave SES patches. For an SAS intersection point $x_{i_3} \in S_{j_1} \bigcap S_{j_2} \bigcap S_{j_3}$, denote by $P_\triangle$ the concave spherical triangle corresponding to x_{i_3} (see the left of Figure 5 for a schematic), that is,

$$P_\triangle = \partial B_{r_{\rm p}}(x_{i_3}) \bigcap \left\{ p : p = x_{i_3} + \sum_{k=1}^{3} \lambda_k (c_{j_k} - x_{i_3}),\ \forall \lambda_k \geq 0 \right\},$$

where c_{j_k} is the center of the sphere S_{j_k} associated with x_{i_3}, $k = 1, 2, 3$, and $B_r(c)$ denotes the open ball with center c and radius r. Notice that $P_\triangle$ is a triangle-shaped patch generated by the sphere $\partial B_{r_{\rm p}}(x_{i_3})$ and three planes, see the left of Figure 5 for an illustration. Then, we have the following result from [18] for computing the concave SES patch P_- corresponding to x_{i_3}.

Theorem 2.2. *$P_- = P_\triangle \setminus \bigcup_{x \in K} B_{r_{\rm p}}(x)$ where $K = \{x \in I : \|x - x_{i_3}\| < 2r_{\rm p},\ x \neq x_{i_3}\}$ and I denotes the set of all SAS intersection points.*

If P_- does not coincide with $P_\triangle$, singular arcs occur on the concave SES patch and constitute part of its boundary ∂P_-, see the right of Figure 5 for an illustration. In fact, Theorem 2.2 says that an arbitrary concave SES patch P_- can be characterized as $P_\triangle$ excluding all spherical probes centered at the "nearby" intersection points in K.

Remark 2.2. *Theoretically speaking, the geometry of a concave SES patch can be as complicated as possible. But Theorem 2.2 provides an analytical representation of each concave SES, which finally gives a complete characterization of the SES together with the analytical representation of each convex SES patch and each toroidal SES patch.*

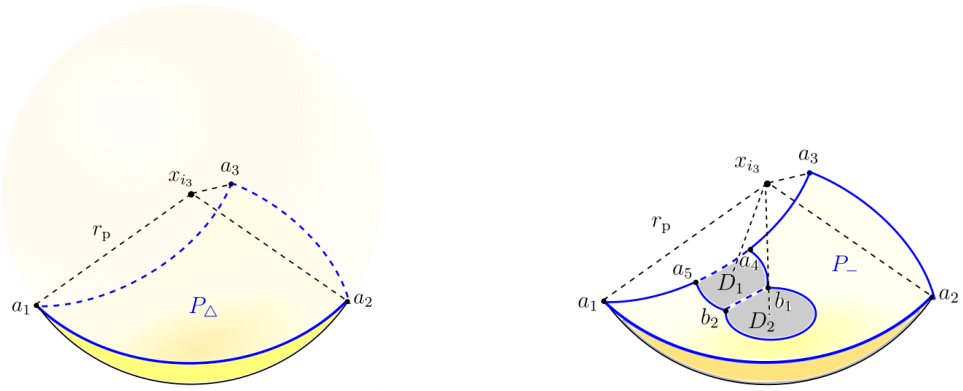

Figure 5: On the left, the concave spherical triangle $P_\triangle$ with vertices (a_1, a_2, a_3) corresponds to an intersection point x_I. In the case where P_- coincide with $P_\triangle$, there will be no singularity on the concave spherical patch. On the right, the concave spherical patch P_- does not coincide with $P_\triangle$ and there are singular circular arcs as parts of its boundary. The vertices of P_- are $(a_1, a_2, a_3, a_4, b_1, b_2, a_5)$. The two flat grey regions D_1 and D_2 are formed by the intersection of $P_\triangle$ with two other nearby spherical probes. These two regions have the boundaries composed of line segments and circular arcs.

3 Construction of Molecular Surfaces

This section will focus on the construction of molecular surfaces by calculating different components of them. We only consider the SAS and the SES since the construction of the VdW-surface is the same as the SAS. Furthermore, we assume that any SAS-ball is not included in another one (otherwise, the inner SAS-ball can be ignored).

3.1 Data Structures of the SAS

We first need to compute an intersection matrix in which the i-th row records the neighboring SAS-spheres intersected with the i-th SAS-sphere. To retrieve all neighboring SAS-spheres, we can use a data structure called a binary spatial division tree proposed by Barnes and Hut with the average complexity $O(\log M)$ [23, 8] where M is the number of atoms. In practice, the maximum number of intersected SAS-spheres for a given SAS-sphere is bounded by a constant $k_{\max}$. In consequence, the intersection matrix is defined of size $M \times k_{\max}$ and each row reports the indices of the neighboring spheres. Based on this intersection matrix, we can establish data structures of the components of both the SAS and then the SES.

An intersection point on the SAS can in theory be formed by the intersection of more than three SAS-spheres. This can appear quite often due to symmetries. In this case, the intersection can however be divided into multiple triplets of SAS-spheres for simplicity. Therefore, we assume that any intersection point is formed by the intersection of three SAS-spheres. On each SAS-sphere, we calculate the intersection points formed by the intersection with any two neighboring SAS-spheres. After calculating the intersection points on each SAS-sphere, we obtain the set of all intersection points I. An intersection point has the following data structure:

SAS Intersection Point

- (x, y, z): coordinate of the point
- (i, j, k): indices of three corresponding SAS-spheres

For each pair of neighboring SAS-spheres, we can calculate all circular arcs on the intersection circle of two intersecting SAS-spheres since all intersection points on this circle are known. To represent each circular arc, we record the starting and ending point, its center, radius, radian and the pair of SAS-spheres with the following data structure:

Circular SAS Arc

- (i_1, i_2): index of the starting and ending intersection point
- (x, y, z): coordinate of the center
- r: radius
- β: radian
- (i, j): indices of two corresponding SAS-spheres

On each SAS-sphere, there are loops composed of circular arcs, which also form the boundaries of spherical patches. In consequence, we can represent each loop by a set of constituting circular arcs and then each spherical patch by a set of loops forming the boundary. The following data structures are used:

SAS Loop

- $l_1, l_2, \ldots, l_{n_1}$: the consisting circular arcs
- i: index of the SAS-sphere on which the loop lies

Spherical SAS Patch

- $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{n_2}$: the loops forming the boundary of the patch
- i: index of the SAS-sphere on which the patch lies

3.2 Assembling Spherical Patches

The crucial problem in the above data structures is to associate a spherical patch with a set of loops forming its boundary. This is tightly connected with determining whether two loops lie on the boundary of a common spherical patch or not, given all loops on a sphere. This motivates us to propose a method based on a binary tree structure. To do so, we define the "interior" and "exterior" of a loop on a sphere. More precisely, a loop divides the sphere into the "interior", the part which is not hidden by any intersected sphere forming this loop, and the "exterior", the remaining part. Denote by $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ the list of loops on the sphere. For any two loops

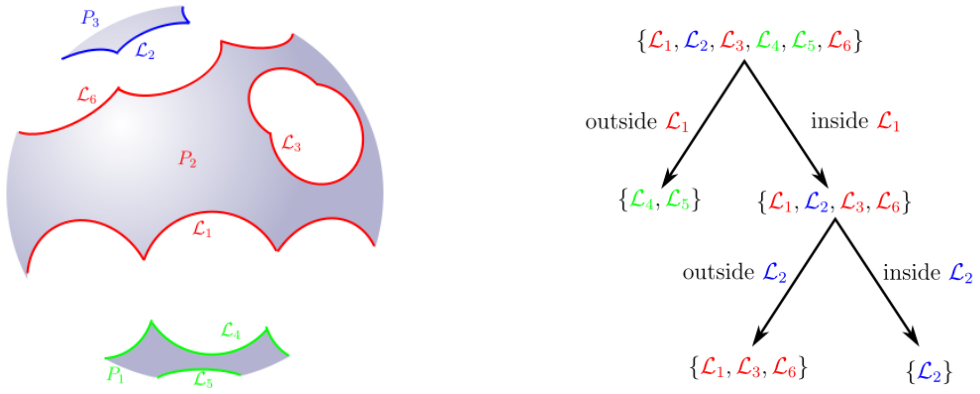

Figure 6: On the left is a brief schematic of an SAS-sphere and the loops on it. There are six loops on the SAS-sphere $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_5, \mathcal{L}_6, \mathcal{L}_6\}$, and three spherical patches with the boundaries formed by two loops in green $\{\mathcal{L}_4, \mathcal{L}_5\}$, three loops in red $\{\mathcal{L}_1, \mathcal{L}_3, \mathcal{L}_6\}$ and one loop in blue $\{\mathcal{L}_2\}$ respectively. The tree on the right illustrates the corresponding binary tree whose leaves identify the boundaries of three different spherical patches.

$\mathcal{L}_i$ and $\mathcal{L}_j$ belonging to the boundary of a common spherical patch, we then have $\mathcal{L}_i \subset \mathcal{L}_j^\circ$ and $\mathcal{L}_j \subset \mathcal{L}_i^\circ$, where $\mathcal{L}_i^\circ$ and $\mathcal{L}_j^\circ$ represent respectively the "interior" of $\mathcal{L}_i^\circ$ and $\mathcal{L}_j^\circ$. In consequence, each spherical patch on the sphere has a boundary composed of loops which are "inside" each other. Here, one loop "inside" another means that the loop is in the "interior" of the other and further, a loop is said to be "inside" itself.

The problem is to classify all loops into different classes such that each loop in one class is "inside" another one in the same class. We first divide the set of loops on a sphere into two subsets by checking whether an element of the set is "inside" a given loop or not. Then, we can look at each of the subsets (loops) and for each subset, we check again if each loop of this subset is "inside" the first loop of this subset. If yes, we continue to check for the next loop of the subset until we find one loop that is "outside" another and then we build two new subsets for this subset. Otherwise, if each loop of the subset is "inside" all the others, we leave this subset as a leaf of the binary tree representing the boundary of a spherical patch. Given an initial loop, we can therefore derive a binary tree whose leaves identify the boundaries of different spherical patches. See Figure 6 for a schematic of this process.

This method is also suitable for assembling a concave spherical SES patch denoted by P_- corresponding to an SAS intersection point x_{i_3} using the formula in Theorem 2.2. We first calculate all loops forming the boundary of P_- each having a data structure as an SAS loop. Then, we classify the set of loops into different subsets each identifying the boundary of a sub-patch using the above method which is based on a binary tree.

3.3 Assembling Surfaces

a. SAS

With the above data structures of different molecular components, we are ready to construct complete and exterior molecular surfaces, i.e., assemble these data structures. The cSAS is composed

Figure 7: Schematics of a rectangle-shaped toroidal patch (left, yellow) and a double-triangle-shaped toroidal patch (right, yellow) corresponding to two circular SAS circles respectively with radius larger than and smaller than r_{p}.

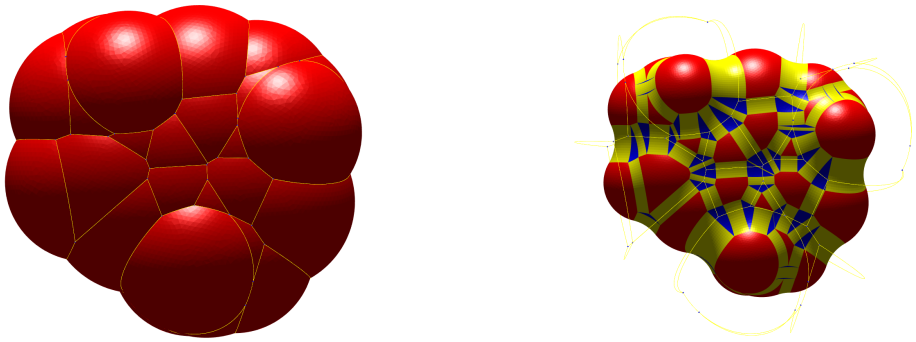

Figure 8: The SAS and the SES of caffeine with probe radius $r_{\mathrm{p}} = 1.5$Å. On the left, the SAS is composed of spherical patches in blue, circular arcs in yellow and intersection points in red. On the right, the patches in red (resp. in yellow or in blue) are convex spherical patches (resp. toroidal patches or concave spherical patches) on the SES, each corresponding to a spherical patch (resp. a circular arc or an intersection point) on the SAS.

of all spherical patches on each SAS-sphere, which have been calculated in Section 3.2. We map a faraway point from the molecule onto a spherical SAS patch and then use this patch as the first element of the eSAS. Two spherical SAS patches are neighbors if they have a common circular arc or circle on their boundaries. By adding neighboring spherical patches one by one, we finally obtain all spherical patches on the eSAS. Since the data structure of each patch contains all necessary information about its neighbors, this is straightforward.

b. SES

As already mentioned, a spherical SAS patch (cSAS or eSAS) corresponds to a similar convex spherical SES patch (respectively cSES or eSES), a circular SAS arc corresponds to a (rectangle-shaped or double-triangle-shaped, see Figure 7) toroidal SES patch and an SAS intersection point corresponds to a concave spherical SES patch obtained from Theorem 2.2. A graphical illustration is presented in Figure 8 for the caffeine molecule. With this geometrical relationship, the construction of the cSES and the eSES can be done directly based on the construction (i.e., the assembling of the data structures) of the cSAS and the eSAS, where the data structures of different SES patches are established as follows:

Rectangle-shaped Toroidal Patch

- i_l: index of the corresponding circular SAS arc
- $\mathcal{L}_1^t$ or $(\mathcal{L}_1^t, \mathcal{L}_2^t)$: one loop composed of four circular arcs or two circles forming the boundary

Double-triangle-shaped Toroidal Patch

- i_l: index of the corresponding circular SAS arc
- $(\mathcal{L}_1^t, \mathcal{L}_2^t)$: two loops each composed of up to three circular arcs forming the boundary

Convex Spherical SES Patch

- i_P: index of the corresponding spherical SAS patch
- $\mathcal{L}_1^+, \mathcal{L}_2^+, \ldots, \mathcal{L}_{m_1}^+$: loops forming the boundary of the patch

Concave Spherical SES Patch (or Subpatch)

- i_I: index of the corresponding SAS intersection point
- $\mathcal{L}_1^-, \mathcal{L}_2^-, \ldots, \mathcal{L}_{m_2}^-$: loops forming the boundary of the patch

In the above data structures of the SES, a loop ($\mathcal{L}_i^t$) on a toroidal patch contains only the corresponding circular arcs (or circles), each having the same structure as a circular SAS arc introduced in the previous subsection. A loop ($\mathcal{L}_i^+$ or $\mathcal{L}_i^-$) on a spherical SES patch has the same structure as an SAS loop containing both the corresponding circular arcs (or circles) and the index of the sphere on which the loop lies.

4 Molecular Inner Holes

As already mentioned, there might exist inner holes in the cSAS-cavity (or the VdW-cavity). In implicit solvation models, one might be interested in filling these inner holes since it is unphysical that the solvent is present in these holes.One possibility is to construct virtual atoms to fill these inner holes, which simultaneously doesn't influence the construction of the eSAS. In particular, these virtual atoms can be treated as completely artificial with the purpose to fill these inner holes. One has however to consider that these virtual balls should not intersect the exterior region of the SAS-cavity.

Denote the exterior SAS by Γ^{e} and the separate inner subsurfaces of the cSAS by Γ_j^{i}, $1 \leq j \leq n$. Further, denote the set of intersection points on Γ^{e} by I^{e} and the set of intersection points on Γ_j^{i} by I_j^{i}. With the above notations, we state the following lemma:

Lemma 4.1. *The distance between two subsurfaces of the cSAS can be characterized by*

$$\mathrm{dist}(\Gamma^{\mathrm{e}}, \Gamma_j^{\mathrm{i}}) = \min_{x \in I^{\mathrm{e}}, y \in I_j^{\mathrm{i}}} \|x - y\|, \quad \forall 1 \leq j \leq n,$$

and

$$\mathrm{dist}(\Gamma_j^{\mathrm{i}}, \Gamma_k^{\mathrm{i}}) = \min_{x \in I_j^{\mathrm{i}}, y \in I_k^{\mathrm{i}}} \|x - y\|, \quad \forall 1 \leq j, k \leq n,$$

where dist(X, Y) *denotes the minimum distance between two sets X and Y defined by* $\mathrm{dist}(X, Y) = \min_{x \in X, y \in Y} \|x - y\|$ *and* $\|\cdot\|$ *denotes the Euclidean norm.*

Proof. Consider any point $p \in \Gamma^{\mathrm{i}}_j$. If it has a closest point $x^p_{\mathrm{sas}} \in \Gamma^{\mathrm{e}}$ lying on a spherical patch or a circular SAS arc, then p must lie on the corresponding line segment or the corresponding triangle according to Theorem 2.1 (and Remark 2.1). However, notice that p is not included by the union of all SAS-spheres. It implies that p can not lie on such a line segment or triangle which is covered by the union of SAS-spheres. This is a contradiction! In consequence, p can only have a closest point to Γ^{e} in the set of intersection points I^{e}. On the other hand, we have the same conclusion that for any point on Γ^{e}, it can only have a closest point to Γ^{i}_j in I^{i}_j. Therefore, the distance between Γ^{e} and Γ^{i}_j is equal to the distance between the two sets of intersection points I^{e} and I^{i}_j. That is to say,

$$\mathrm{dist}(\Gamma^{\mathrm{e}}, \Gamma^{\mathrm{i}}_j) = \min_{x \in \Gamma^{\mathrm{e}}, y \in \Gamma^{\mathrm{i}}_j} \|x - y\| = \min_{x \in I^{\mathrm{e}}, y \in \Gamma^{\mathrm{i}}_j} \|x - y\| = \min_{x \in I^{\mathrm{e}}, y \in I^{\mathrm{i}}_j} \|x - y\|.$$

Further, we have the same result for Γ^{i}_j and Γ^{i}_k, i.e., the distance between Γ^{i}_j and Γ^{i}_k is equal to the distance between the two sets I^{i}_j and I^{i}_k. □

The above lemma allows a fast computation of the distance from an inner cavity to the exterior SAS or to another inner cavity since the right-hand sides only compute the distances among the finite number of intersection points. With the distances between each inner subsurface and the exterior subsurface Γ^e of the cSAS (i.e. the eSAS), we can add virtual spheres with small enough radii so that each virtual sphere doesn't intersect Γ^{e}. This prevents the added virtual spheres affecting the geometry of Γ^{e}.

Given an inner subsurface Γ^{i}_j, we want to fill the inner hole enclosed by it with virtual spheres. We propose the following three steps to do this:

- First, we fix the radii of virtual atoms as $\delta_j = \dfrac{1}{2}\mathrm{dist}(\Gamma^{\mathrm{e}}, \Gamma^{\mathrm{i}}_j)$.
- Second, we take a small rectangular box containing the inner hole and take a set of virtual spheres with radius δ_j which cover this box completely, implying that the inner hole is also covered.
- Third, we remove all virtual spheres whose centers lie outside the inner hole and at the same time have distance to Γ^{i}_j greater than δ_j. In other words, we remove those virtual spheres which don't intersect the inner hole and therefore don't contribute to fill the inner hole.

In consequence, the remaining virtual spheres (atoms) covers the inner hole enclosed by Γ^{i}_j while do not intersect the exterior region of the SAS-cavity. By repeating the above steps for each inner cavity with the boundary Γ^{i}_j, $\forall 1 \leq j \leq n$, we can finally fill all inner holes with virtual spheres. Figure 9 illustrates the added virtual spheres for filling four inner holes of molecule 1ETN, one sphere for each hole.

5 Meshing

The SAS is composed of spherical patches like the VdW-surface but with increased radii $r_i + r_{\mathrm{p}}$. In consequence, meshing the SAS or the VdW-surface can be reduced to developing a meshing algorithm for an arbitrary spherical patch given its SAS-center, its SAS-radius and its boundary information obtained from the above data structures. This algorithm can also be applied to mesh a convex or concave SES patch, since its center, its radius and its boundary information are known. In this section, we propose a meshing algorithm of molecular surface consisting of two sub-algorithms respectively for meshing a spherical patch and meshing a toroidal patch.

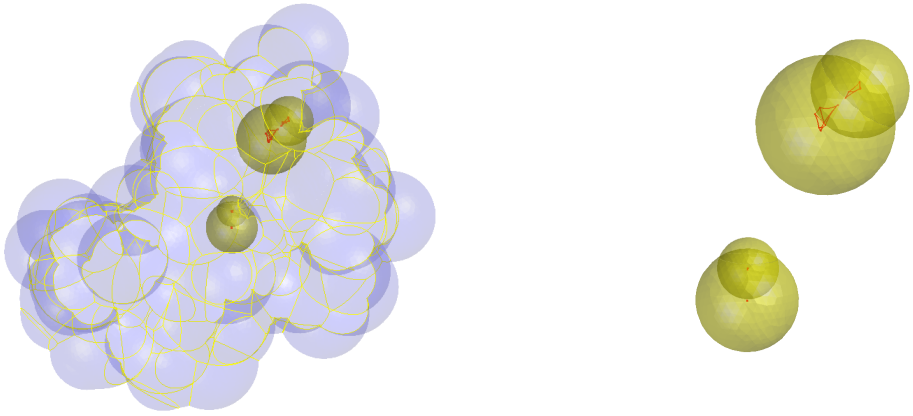

Figure 9: The left figure shows the SAS (blue) of 1ETN with $r_{\mathrm{p}} = 0.7$Å and the added virtual spheres (yellow) filling the inner holes. On the right, the inner holes and the added virtual spheres are magnified for a better view.

5.1 Boundary Division

We first give a strategy for dividing the boundary of each (toroidal or spherical) patch on the SAS or the SES, which ensures that the meshes of two neighboring patches match on their interface, i.e., that the final global mesh will be conforming.

To divide the boundary of a toroidal or a spherical patch, we set initially the triangle size (the approximate length of a triangle edge) to d, which should be relatively small compared to the radius of the spherical patch. Since the boundary of a patch consists of loops which are composed of circular arcs, we make a uniform division of each circular arc on the boundary. The radius and the radian of a circular arc l_m are denoted by r_{l_m} and θ_{l_m}. At the same time, we set a maximum allowed angle variation between two neighboring division points to α_0 (in our codes, we use $\alpha_0 = 60^\circ$) in the case where the radius of the circular arc r_{l_m} is small compared to d. Then, the number of elements of the discretization of this circular arc denoted by N_{l_m} is set as follows:

$$N_{l_m} = \max\left\{ \left\lfloor \frac{r_{l_m}\theta_{l_m}}{d} \right\rfloor + 1, \ \left\lfloor \frac{\theta_{l_m}}{\alpha_0} \right\rfloor + 1 \right\}, \tag{5.5}$$

where $\lfloor \cdot \rfloor$ is the floor function which maps a real number to its largest smaller integer. In consequence, this ensures that the distance and the angle variation between two neighboring division points are respectively smaller than d and α_0.

5.2 Spherical Patches

In this subsection, we use the basic advancing-front method, see [24, 25, 26, 27, 28] for an overview of this technique, for meshing a (convex or concave, SAS or SES) spherical patch uniformly, with its center, its radius and its boundary information known. We only present the brief scheme of this method working on a sphere. However, we emphasize that any other suitable meshing algorithm

Figure 10: Meshing two spherical patches.

can be applied to the spherical patch as long as they conserve the given boundary partition. For instance, the marching cubes algorithm [29, 30] or the Delaunay refinement algorithm [31, 32], can be taken into account. Therefore, the following algorithm can be replaced with another algorithm of choice.

The process of any advancing-front method can be summarized as follows:

(1) Initialization of the front.

(2) Creation of an internal element

- determination of the departure zone;
- analysis of the entity and creation of (an) internal point(s) and (an) internal element(s);
- update the front.

(3) Repeat the creation of elements as long as the front is not empty.

For a given spherical patch, the initial front is chosen naturally to be its boundary which has been divided in Section 5.1. Then, a departure edge in the front is analyzed from which one or several new internal triangles are created. The front is updated and the process repeated until the front is empty, that is, when the front has merged and the spherical patch is entirely meshed.

To keep the completeness, we present in the Appendix our implementation of the advancing-front algorithm. Also, we illustrate the mesh of a spherical patch that is obtained using this advancing-front algorithm in Figure 10.

Remark 5.1. *Considering that the goal is to visualize molecular surfaces, the quality of mesh will not be focused on. If one aims to high-quality meshes, a remeshing process might be helpful or one can also use a different advancing-front algorithm.*

5.3 Toroidal Patches

To mesh a toroidal SES patch, we should distinguish the cases of a rectangle-shaped patch and a double-triangle-shaped patch. We can parameterize a rectangle-shaped patch by defining a mapping from the toroidal patch to a rectangle having the same side length and the same boundary division as this patch. In consequence, given the boundary division of the rectangle, we first mesh

Figure 11: Schematics of meshing a rectangle (left) and a isosceles triangle (right) with the boundary division given.

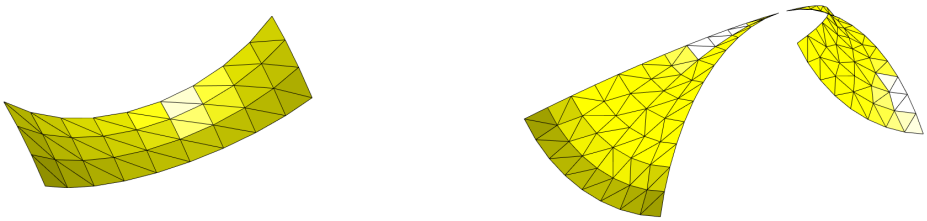

Figure 12: Meshing a rectangle-shaped patch (left) and a double-triangle-shaped patch (right).

it in a trivial way (see the left of Figure 11 for a schematic) and then map the vertices of this mesh back to the patch to obtain the mesh of the rectangle-shaped patch. Similarly, we can parameterize a double-triangle-shaped patch by defining a mapping from this patch to two isosceles triangles respectively having the same side lengths and the same boundary divisions. We first mesh the two isosceles triangles (see the right of Figure 11 for a schematic) and then map the vertices of this mesh back to the patch to obtain the mesh of the double-triangle-shaped patch. Since the toroidal SES patch is part of a torus generated by rotating the spherical probe around an axis, these two mappings can be explicitly given using the parametric representation of a torus [33].

We illustrate the mesh of a rectangle-shaped toroidal patch on the left of Figure 12 and the mesh of a double-triangle-shaped toroidal patch on the right.

5.4 Mesh Refinement

Once the mesh of a molecular patch is established, it is easy to refine it uniformly. Indeed, we can bisect each edge of the mesh and map its middle point to the closest point on the patch which can be computed given the data structure of the patch. Then, each triangle is replaced with four smaller triangles formed by the three vertices of this triangle and three closest points to the middle points of the three edges. In consequence, the refined mesh consists of these smaller triangles. This process of refinement is quite efficient with the complexity proportional to the number of triangles in the mesh.

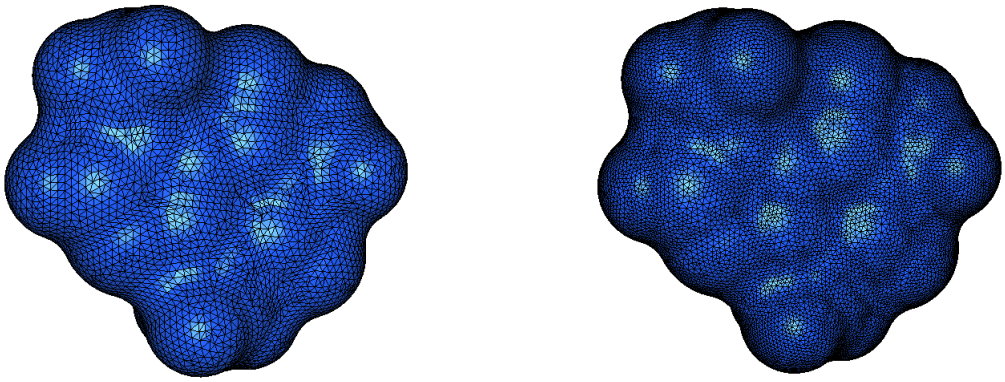

Figure 13: The mesh (right) of caffeine before remeshing with $r_{\mathrm{p}} = 1$Å and the mesh (left) after remeshing with the MMGS tool.

5.5 Remeshing

In many cases, high-quality meshes of molecular surfaces are required. One way to further improve the meshes generated by the above advancing-front algorithm is to use surface remeshing tools. With a remeshing tool, one can usually set some quality requirements and obtain an output mesh approximating well the input mesh. Figure 13 illustrates the mesh of caffeine using the remeshing tool MMGS developed by Dapogny et al. [34] as an example. One should notice that in this case, the remeshing can not ensure that all vertices lie exactly on the molecular surfaces anymore but it can keep the initial vertices of the input mesh.

5.6 Illustrations

We visualize in Figure 14 the meshes of the eSES of some artificial molecules and in Figure 15 and Figure 16 the meshes of eSES of some non-artificial molecules. In both cases, they are generated by the above-proposed meshing algorithm. In these figures, the green curves are the boundaries of all singular concave SES patches which are computed according to Theorem 2.2. On these singular patches, some singular arcs of the surface appear which might cause the self-intersection problem encountered in previous implementations. Generally speaking, the existence of singularities is quite often and the larger the molecule is, the more singularities exist.

In addition, we should mention that the techniques of molecular visualization (for example, [9, 10, 11, 12]) can also be combined with the molecular data structures constructed in Section 3, taking into account the complete characterization of the SES.

5.7 Computational Cost

To get a first idea about the efficiency of the proposed algorithm, we present the total run time with respect to the number of atoms of different molecules. This program was run on a laptop with 2.5GHz quad-core Intel Core i7 processor in Matlab. Figure 17 demonstrates the relationship

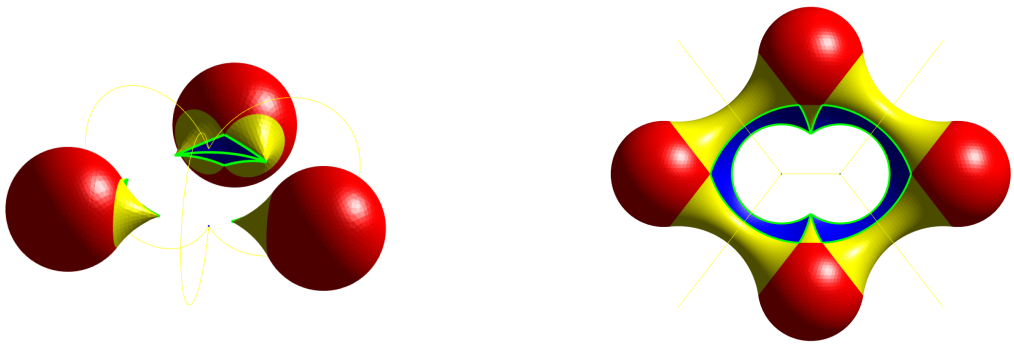

Figure 14: The eSES of three artificial spheres (left) and the eSES of four artificial spheres (right) where the green curves are the boundaries of singular concave patches.

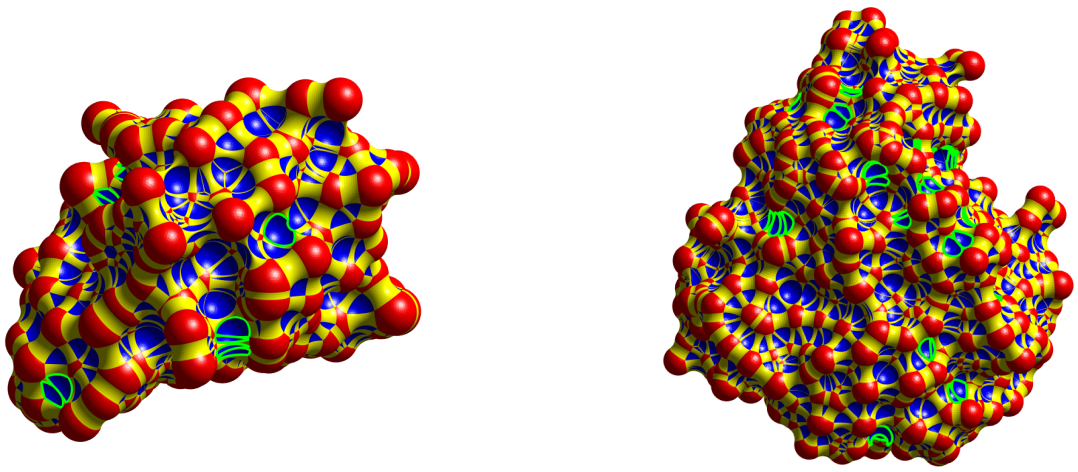

Figure 15: The eSES of molecule 1B17 with 485 atoms (left) and the eSES of molecule 101M with 1414 atoms (right) where $r_{\mathrm{p}} = 1.5$Å. The green curves are the boundaries of singular concave patches.

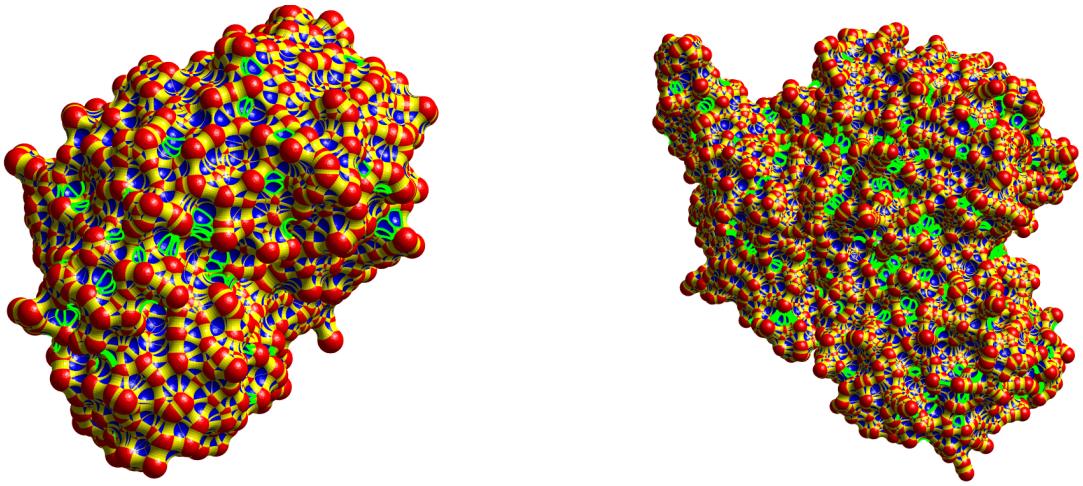

Figure 16: The eSES of molecule 4S19 with 3333 atoms (left) and the eSES of molecule 1IA0 with 9606 atoms (right) where $r_{\mathrm{p}} = 1.5$Å. The green curves are the boundaries of singular concave patches.

between the total run time and the size of molecule, where we observe almost a linear relationship. In order to lower the pre-constant of the linear scaling, one has to refer to a proper and more professional implementation in a better performing language. We therefore expect a better performance of the proposed method in fortran or C++.

6 Conclusion

In this article, we presented the construction of data structures for different molecular surfaces containing all information of their components. At the heart of our method is the recently developed singularity analysis of the SES which avoids the problem of self-intersection. This allows us to develop a meshing algorithm by meshing separately each patch, which includes two sub-algorithms respectively for meshing a (convex or concave, SAS or SES) spherical patch with an advancing-front method and for meshing a toroidal (SES) patch. It is also worth mentioning that each vertex of the created mesh lies exactly on the molecular surface. In addition, we propose an algorithm for filling molecular inner holes with virtual spheres since in some cases, the presence of these inner holes is unphysical, in particular in the context of solvation model. We provide therefore a way to treat these inner holes for more accurate chemical computation. In the future, we will focus on the computation of solvation energy in the polarizable continuum solvation model (PCM) using the SES-cavity, based on the complete characterization of the SES, together with the proposed algorithm for filling inner holes.

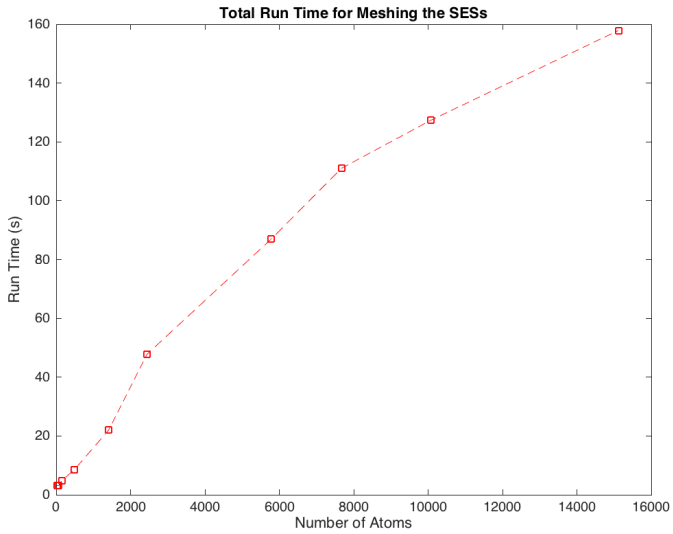

Figure 17: Total run time of the proposed algorithm for meshing the SESs of molecules with various sizes where the approximate triangle size d is set to be 0.5Å and the probe radius $r_{\mathrm{p}} = 1.5$Å.

References

[1] Jacopo Tomasi, Benedetta Mennucci, and Roberto Cammi. Quantum mechanical continuum solvation models. *Chemical reviews*, 105(8):2999–3094, 2005.

[2] M. L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16(5):548–558, Oct 1983.

[3] Tanya M Raschke, Jerry Tsai, and Michael Levitt. Quantification of the hydrophobic interaction by simulations of the aggregation of small hydrophobic solutes in water. *Proceedings of the National Academy of Sciences*, 98(11):5965–5969, 2001.

[4] Anne Lopes, Sophie Sacquin-Mora, Viktoriya Dimitrova, Elodie Laine, Yann Ponty, and Alessandra Carbone. Protein-protein interactions in a crowded environment: an analysis via cross-docking simulations and evolutionary information. *PLoS Comput Biol*, 9(12):e1003369, 2013.

[5] Byungkook Lee and Frederic M Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology*, 55(3):379–IN4, 1971.

[6] Frederic M Richards. Areas, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering*, 6:151–176, 1977.

[7] Richard M Jackson and Michael JE Sternberg. A continuum model for protein–protein interactions: application to the docking problem. *Journal of molecular Biology*, 250(2):258–275, 1995.

[8] Michel F Sanner, Arthur J Olson, and Jean-Claude Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, 1996.

[9] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

[10] Daniel Kauker, Michael Krone, Alexandros Panagiotidis, Guido Reina, and Thomas Ertl. Rendering molecular surfaces using order-independent transparency. In *EGPGV*, pages 33–40, 2013.

[11] Michael Krone, Katrin Bidmon, and Thomas Ertl. Interactive visualization of molecular surface dynamics. *IEEE transactions on visualization and computer graphics*, 15(6):1391–1398, 2009.

[12] Roman A Laskowski. Surfnet: a program for visualizing molecular surfaces, cavities, and intermolecular interactions. *Journal of molecular graphics*, 13(5):323–330, 1995.

[13] Julius Parulek and Ivan Viola. Implicit representation of molecular surfaces. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE*, pages 217–224. IEEE, 2012.

[14] Patrick Laug and Houman Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18(3):199–210, 2002.

[15] Oleg V Tsodikov, M Thomas Record, and Yuri V Sergeev. Novel computer program for fast exact calculation of accessible and molecular surface areas and average surface curvature. *Journal of computational chemistry*, 23(6):600–609, 2002.

[16] Neil R Voss and Mark Gerstein. 3v: cavity, channel and cleft volume calculator and extractor. *Nucleic acids research*, page gkq395, 2010.

[17] Felipe A Bulat, Alejandro Toro-Labbé, Tore Brinck, Jane S Murray, and Peter Politzer. Quantitative analysis of molecular surfaces: areas, volumes, electrostatic potentials and average local ionization energies. *Journal of molecular modeling*, 16(11):1679–1691, 2010.

[18] Chaoyu Quan and Benjamin Stamm. Mathematical analysis and calculation of molecular surfaces. *Journal of Computational Physics*, 322:760 – 782, 2016.

[19] Anthony K Rappé, Carla J Casewit, KS Colwell, WA Goddard Iii, and WM Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American Chemical Society*, 114(25):10024–10035, 1992.

[20] Michel C Delfour and J-P Zolésio. *Shapes and geometries: metrics, analysis, differential calculus, and optimization*, volume 22. Siam, 2011.

[21] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[22] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005.

[23] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324:446–449, December 1986.

[24] Paul Louis George and Éric Seveno. The advancing-front mesh generation method revisited. *International Journal for Numerical Methods in Engineering*, 37(21):3605–3619, 1994.

[25] Peter Möller and Peter Hansbo. On advancing front mesh generation in three dimensions. *International Journal for Numerical Methods in Engineering*, 38(21):3551–3569, 1995.

[26] Rainald Löhner. Progress in grid generation via the advancing front technique. *Engineering with computers*, 12(3-4):186–210, 1996.

[27] Joachim Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science*, 1(1):41–52, 1997.

[28] Pascal J Frey, Houman Borouchaki, and Paul-Louis George. 3d delaunay mesh generation coupled with an advancing-front approach. *Computer methods in applied mechanics and engineering*, 157(1):115–131, 1998.

[29] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[30] Pascal Jean Frey and Paul-Louis George. *Mesh Generation: Application to Finite Elements*. ISTE, 2007.

[31] James C Caendish, David A Field, and William H Frey. An apporach to automatic three-dimensional finite element mesh generation. *International journal for numerical methods in engineering*, 21(2):329–347, 1985.

[32] William H Frey. Selective refinement: a new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24(11):2183–2200, 1987.

[33] Wikipedia. Torus — wikipedia, the free encyclopedia, 2016. [Online; accessed 3-November-2016].

[34] Charles Dapogny, Cécile Dobrzynski, and Pascal Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358–378, 2014.

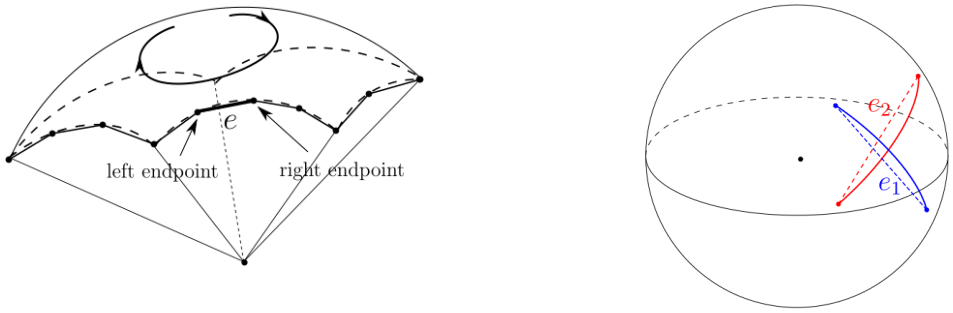

Figure 18: On the left, the left endpoint and the right endpoint of edge e are determined by the orientation of the loop. On the right, two dashed edges e_1 (in blue) and e_2 (in red) "intersect" in the sense that their two projected chords in form of circular arcs intersect on the sphere.

A Appendix: Advancing-front Algorithm

A.1 Data Structure

Denote by $P \in \mathbb{R}^{N_p \times 3}$ the array of coordinates of the N_p points in the mesh, initialized to be the coordinates of all points of the boundary division. We define the orientation of any loop on the spherical patch satisfying that the interior of this patch is always on the right-hand side if one goes along the loop. In consequence, each edge on a loop is endowed with an orientation, implying that we can classify its two endpoints to the right endpoint (or the starting endpoint) and the left endpoint (or the ending endpoint) viewing from the outside of the sphere where the patch lies, see the left of Figure 18 for a schematic.

Since the front might consist of several loops, we choose one of them as the active loop. Any edge on this active loop is called an active edge and any point on the active loop is called an active point. All active edges are sorted by the orientation of the active loop. We always choose the first active edge as the departure edge mentioned in Subsection 5.2 and create a new triangle having the active edge as one side. Then, we update the set of active edges and go to the next active edge.

At each step, the set of active edges is represented by a matrix A_e of size $N_{ae} \times 2$ where N_{ae} is the number of active edges. Any active edge in A_e is represented by the indices of its two endpoints in P. The set of triangles of the mesh at each step are represented by a matrix T of size $N_t \times 3$ recording the indices of the three vertices of a triangle where N_t is the number of triangles in the mesh.

A.2 Check Front Points

For a given active edge denoted by e with the right endpoint P_1 and the left endpoint P_2, we want to construct a point P_0 for creating a new triangle with the edge e and the opposite point P_0. The unit normal vectors at P_1 and P_2 on the spherical patch are denoted respectively by $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$. Then, we define a unit vector $\overrightarrow{n_e}$ at the middle point of e by

$$\overrightarrow{n_e} = \frac{\overrightarrow{P_1P_2}}{|\overrightarrow{P_1P_2}|} \times \frac{\overrightarrow{n_1} + \overrightarrow{n_2}}{2},$$

which is perpendicular to the edge e and is tangent to the spherical patch at the same time. Geometrically speaking, $\overrightarrow{n_e}$ is a tangential unit vector to the sphere pointing towards the unmeshed region of the patch.

In the following, the notion of point-edge distance between a point and an edge is introduced and defined as the sum of the Euclidean distances between the point and two endpoints of the edge. Notice that each edge in the mesh is a chord of the corresponding sphere. Note that the projection of the chord onto the sphere is a circular arc which lies on the plane generated by the chord and the spherical center. We say that two edges "intersect" if their projected chords in form of circular arcs intersect on the sphere, see the right of Figure 18 for a schematic of two "intersecting" edges.

For the edge e, the neighboring active edge with P_1 as a common endpoint is called the right neighboring active edge denoted by e_1. The one with P_2 as a common endpoint is called the left neighboring active edge denoted by e_2. The endpoint of e_1, other than P_1, is denoted by P_r and the endpoint of e_2, other than P_2, is similarly denoted by P_l. By projecting the vectors $\overrightarrow{P_1P_l}$ and $\overrightarrow{P_1P_2}$ to the tangent plane to the sphere at point P_1, we obtain two vectors $\overrightarrow{\tau_{1l}}$ and $\overrightarrow{\tau_{12}}$. The angle between e and e_1, denoted by α_1, is then defined to be the angle between vectors $\overrightarrow{\tau_{1l}}$ and $\overrightarrow{\tau_{12}}$ if $\det(\overrightarrow{\tau_{1l}}, \overrightarrow{\tau_{12}}, \overrightarrow{n_1}) \geq 0$ and to be 2π minus the angle between vectors $\overrightarrow{\tau_{1l}}$ and $\overrightarrow{\tau_{12}}$ if $\det(\overrightarrow{\tau_{1l}}, \overrightarrow{\tau_{12}}, \overrightarrow{n_1}) < 0$ where $\det(\cdot)$ denotes the determinant of a matrix. The angle between e and e_2, denoted α_2, is defined similarly to α_1.

We first check if there exists possible points among the front points for the creation of a new triangle and collect all candidates in a set S_f. This means that we first try to create a new triangle with the existing front points without creating a new point. We propose two criterions for adding front points to S_f as follows:

- Angle Criterion

  Set a minimal angle between two neighboring active edges to ϵ_α. If $\alpha_1 < \epsilon_\alpha$, add the left neighboring active point to S_f, see the left of Figure 19 for a shematic. If $\alpha_2 < \epsilon_\alpha$, add the right neighboring active point to S_f.

- Point-edge Criterion

  Set a point-edge distance tolerance between a front point and a front edge to ϵ_e. For a front point P_f from the set of points of all loops, other than P_1 or P_2, we check if it has a distance to e smaller than $|e| + \epsilon_e$ where $|e|$ is the length of e, see the right of Figure 19. Further, we check if the scalar product between $\overrightarrow{n_e}$ and the vector from the middle point of e to P_f is positive, and simultaneously if both edges P_fP_1 and P_fP_2 do not "intersect" any other front edge. If all conditions are satisfied, add P_f to S_f.

The angle criterion is used to check if the angle between e and one of its neighboring edges is small and the point-edge criterion is used to check if there exists any front point close to the edge e. In consequence, S_f is a list of front points P_f that are possibly suited for creating a new triangle with the edge e and the opposite point P_f.

If S_f is not empty, we sort it with respect to the point-edge distance to the edge e and choose the one denoted by P_0 that has the minimal distance to e, i.e.,

$$P_0 = \text{argmin}_{p \in S_f} \text{dist}(p, e), \tag{1.6}$$

where $\text{dist}(p, e)$ is the point-edge distance between a point p and an edge e. This ensures that any other point in S_f does not lie in the triangle $\triangle P_0P_1P_2$.

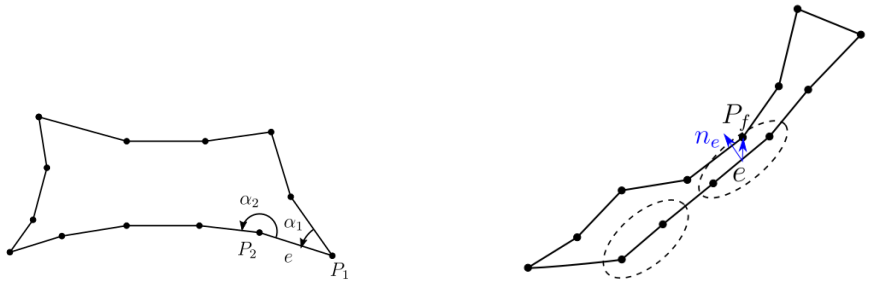

Figure 19: Planar schematics of two criterions for searching a possibly front point: angle criterion (left) and point-edge distance criterion (right).

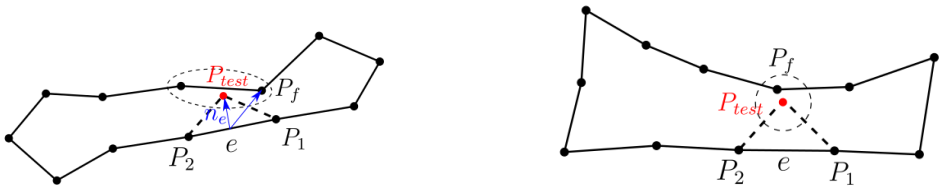

Figure 20: Planar schematics of two criterions for checking if P_{test} is suited as a new vertex of the mesh: point-edge criterion (left) and distance criterion (right).

A.3 Create a New Point

In the above subsection, we first scan the set of front points that can be used to create a new triangle for a given edge e. Nevertheless, if S_f is empty, we should consider to create a new testing point P_{test} on the spherical patch. This testing point is constructed such that the edges $P_{test}P_1$ and $P_{test}P_2$ have the same length and that $\triangle P_1P_2P_{test}$ has a fixed height h (we take $h = \frac{\sqrt{3}}{2}d$). In the case where $|e| = d$, the triangle $\triangle P_{test}P_1P_2$ is equilateral. Then, we check if P_{test} is suited as a new vertex of the mesh with the following two criterions:

- Point-edge Criterion

  Check first if there exists a front edge e_f having a point-edge distance to P_{test} smaller than $|e_f|+\epsilon_e$, see the left of Figure 20. If yes, we further check for each endpoint of e_f (still denoted by P_f) if the scalar product between $\overrightarrow{n_e}$ and the vector from the middle point of e_f to P_f is positive and simultaneously the edges P_fP_1 and P_fP_2 do not "intersect" any other front edge. If all conditions are satisfied for the edge e_f and the endpoint P_f, we add P_f to S_f.

- Distance Criterion

  Set a distance tolerance between a testing point and a front point to ϵ_d. If there exists a front point P_f with distance to P_{test} smaller than ϵ_d (see the right of Figure 20) and both edges P_fP_1 and P_fP_2 do not "intersect" any other front edge, then we add P_f to S_f.

If S_f is not empty now, we still select the point P_0 using formula (1.6) with this S_f. If S_f is empty, we determine P_0 as P_{test}.

A.4 Update Front

The previous section was devoted to determine the point P_0 given an edge e. Then, a new triangle can be created by connecting the endpoints of e and the point P_0.

After the creation of the triangle, we update the active loop including A_e and N_{ae}, the set of vertices of the mesh including P and N_p, as well the set of triangles T and N_t. However, we should pay attention to two special cases of updating the active loop. If the point P_0 is a front point on the active loop but not a neighboring active point of e, the active loop is divided into two parts and we chose one of them to be the new active loop. If the point P_0 is a front point but not on the active loop, we add the loop on which P_0 lies to the active loop to form an active larger loop. After updating, we go to the next active edge on the active loop and repeat the process until the front has merged.

To obtain a mesh that is as uniform as possible, it is also necessary to control the length of each newly created edge. From the boundary division of circular arcs in Section 5.1, the length of any edge on the initial front is (in most cases, slightly) smaller than d. After the initialization, we bisect the newly created edge whenever its length is larger than a given tolerance d_0 and then map its middle point to the closest point on the sphere in order to obtain two new shorter edges. This technique ensures that each edge of the mesh will not become too large. Like this, we control the maximal diameter of each triangle.

In the advancing-front algorithm for meshing a spherical patch, we generate a surface mesh on the sphere which is essentially as difficult as generating a mesh for a planar domain in 2D. In this process, a new triangle can always be created by determining an optimal point P_0 and therefore the front will finally merge. This implies that a dead lock [27] of the front will not appear. From another point of view, we can first transform the spherical patch to a planar region, mesh it using a 2D mesh generator such as the NetGen [27] and transform the mesh back to the original spherical patch. In consequence, the robustness of the proposed advancing-front algorithm can be guaranteed.