A Fast Meshing Algorithm for Molecular Surfaces

Chaoyu Quan, Benjamin Stamm

A Fast Meshing Algorithm for Molecular Surfaces

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

[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*

SUMMARY

We derive a fast meshing algorithm for molecular surfaces that is based on patch-wise meshing using an advancing-front method adapted to the particular case of molecular surfaces encountered in implicit solvation models in computational chemistry. 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 that allows to describe and characterize all singularities explicitly. The presented algorithm thus avoids completely the problem of self-intersection of the SES.

KEY WORDS: Molecular Surface, Solvent Excluded Surface, Molecular Visualization, Meshing, Advancing-front Method

1. INTRODUCTION

The majority of chemically relevant reactions take place in the liquid phase and the effect of the environment (solvent) is important and should be considered in in silico computations. 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, see [15] for a review article and references therein. A second field where the notion of molecular surface is important is simply the visualization of molecules. To model a molecular surface of the solute, each constituting atom is idealized by a simple sphere with its Van der Waals (VdW) radius. Three kinds of molecular surfaces are commonly used in solvation models or in molecular visualization: the VdW surface, the Solvent Accessible Surface (SAS) and the Solvent Excluded Surface (SES).

1.1. Previous Works

The definitions of the SAS and the SES were first introduced by Lee & Richards [7, 12] in the 1970s. The SES is also called the "smooth molecular surfaces" or "Connolly's surfaces" due to Connolly's fundamental calculation on it [3]. In 1996, Michel Sanner proposed his MSMS (Michel Sanner's Molecular Surface) algorithm for meshing molecular surfaces [13] which is now one of the most widely-used packages for molecular visualization. Besides, there are many other contributions on the visualization of molecular surfaces [6, 9] or the calculation of molecular areas and volumes [16, 2]. However, a computable implicit representation of the SES and a complete characterization of the SES-singularities remained unsolved until a recent paper by us [10].

1.2. Contribution

We develop a fast meshing algorithm for molecular surfaces, especially the SES, based on the analytical characterization of the SES including its singularities presented in [10]. The explicit

*Correspondence to: Benjamin Stamm, Sorbonne Universités, UPMC Univ Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France. E-mail: stamm@ann.jussieu.fr

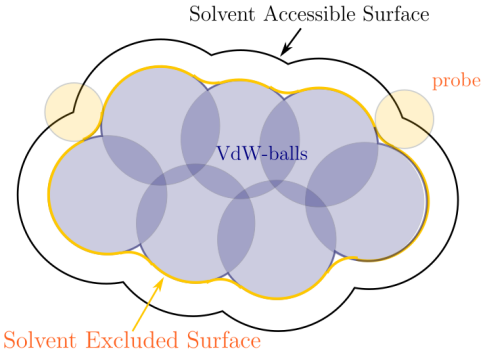

Figure 1. 2D-schematics of the Solvent Accessible Surface and the Solvent Excluded Surface, both defined by a spherical probe in orange rolling over the molecular VdW-atoms in dark blue.

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, improves greatly the possibilities of meshing the surface.

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, a fast meshing algorithm for molecular surfaces is developed including two sub-algorithms for meshing respectively a (convex or concave) spherical patch and a toroidal patch in the fourth section. Then, we give the illustrations of some artificial as well as realistic molecular surfaces in the fifth section. Finally, a conclusion is presented in the last section.

2. MOLECULAR SURFACES

A mathematical analysis and calculation of the SAS and the SES have been presented in our recent work [10]. In this section, we recall some results including a mathematical definition of the surfaces using their implicit representations.

2.1. Definitions

In quantum chemistry, atoms of a molecule are represented by VdW-balls with VdW-radii which are experimentally fitted, given the underlying chemical element [11]. In consequence and mathematically speaking, 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 the center of an idealized spherical probe rolling over the solute VdW-atoms, 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 solute VdW-atoms, 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 1 for a graphical illustration.

Sometimes, the SAS can be non-connected: it can be composed of several nonintersecting separate surfaces. We call the outmost surface as the exterior Solvent Accessible Surface (eSAS) and the union of all separated surfaces as the complete Solvent Accessible Surface (cSAS), see Figure 2 for an example and [10] 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 to both the cSES and the eSES.

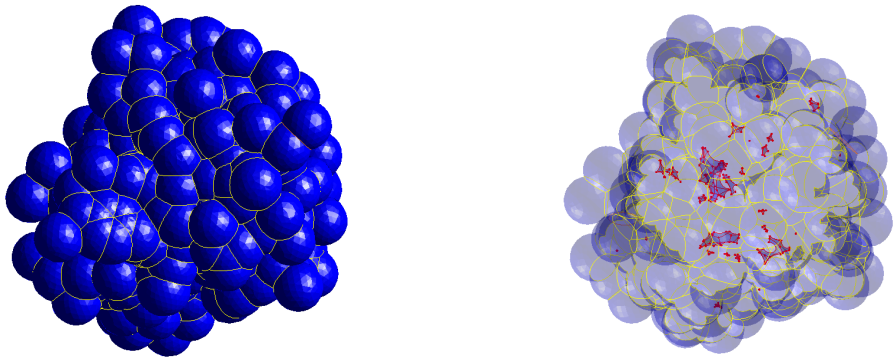

Figure 2. The eSAS (left) and the transparent cSAS (right) of 1B17 with 485 atoms.

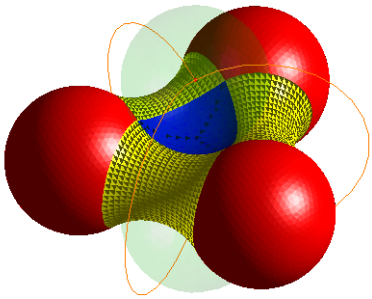

Figure 3. 3D schematic of the SES illustrating the convex spherical patches (red), the toroidal patches (yellow) and the concave spherical patches (blue).

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 [3]: convex spherical patches, toroidal patches and concave spherical patches, see Figure 3 for a 3D illustration. As shown in [10], any point on a convex spherical patch of the SES has a closest point to the SAS lying on a spherical patch. Similarly, any point on a toroidal patch of the SES has a closest point to the SAS lying 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 $f_{vdw}^{-1}(0)$ with the following implicit function:

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

Similarly, the open ball with center c_i and radius $r_i + r_p$ 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 $\widetilde{f}_{sas}^{-1}(0)$ with the following implicit function:

$$\widetilde{f}_{sas}(p) = f_{vdw}(p) - r_p = \min_{i=1,\dots,M} \{\|p - c_i\|_2 - r_i - r_p\}, \quad \forall p \in \mathbb{R}^3. \tag{2}$$

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 $\widetilde{f}_{sas}(p)$ is not 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 [10], 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. The signed distance function $f_{sas}(p)$ can then be easily written as:

$$f_{sas}(p) = \begin{cases} -\|p - x^p_{sas}\| & \text{if } p \text{ lies inside the SAS,} \\ \|p - x^p_{sas}\| & \text{if } p \text{ lies outside the SAS.} \end{cases} \tag{3}$$

The difficulty is however to find efficiently the closest point x^p_{sas}.

According to the fact that any point on the SES has signed distance $-r_p$ to the SAS, an implicit function of the SES is obtained directly as:

$$f_{ses}(p) = f_{sas}(p) + r_p, \tag{4}$$

which motivates the choice of using the signed distance function denoted by $f_{sas}(p)$ to represent the SAS. From the above formula, the SES can be represented by a level set $f^{-1}_{sas}(-r_p)$, associated with the signed distance function f_{sas} to the SAS. Therefore, the key point becomes how to compute the signed distance $f_{sas}(p)$ 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 to compute the signed distance from p to S. However, considering that the SAS is a special surface composed of spherical patches, this computation can be done analytically [10].

Further, the region enclosed by the VdW surface is called the VdW-cavity, that is, any point p in the VdW-cavity satisfies $f_{vdw}(p) \leq 0$. 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 $f_{sas}(p) \leq 0$, and any point p in the SES-cavity satisfies $f_{ses}(p) \leq 0$.

3. CONSTRUCTION OF MOLECULAR SURFACES

This section will focus on the construction of molecular surfaces by calculating different components of them. Since the basic ideas have been showed only briefly in [10], we will present here more details about how to construct such surfaces in practice. 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 by any other 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 neighbor SAS-spheres intersected with the i-th SAS-sphere. To retrieve all neighbor SAS-spheres, we can use a data structure called a binary spatial division tree proposed by Barnes and Hut with the average complexity $O(\log M)$ [1, 13] where M is the number of atoms. In practice, the maximum number of intersection 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 symmetry. 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 neighbor 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): indices 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

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

Spherical SAS Patch

- $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{n_2}$: indices of 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 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 (open) ball forming this loop, and the "exterior", the remaining part. Therefore, a loop itself belongs to its "interior". Denote by $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ the list of loops on the sphere. For any two loops $\mathcal{L}_i$ 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 $\mathcal{L}_i$ and $\mathcal{L}_j$. 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.

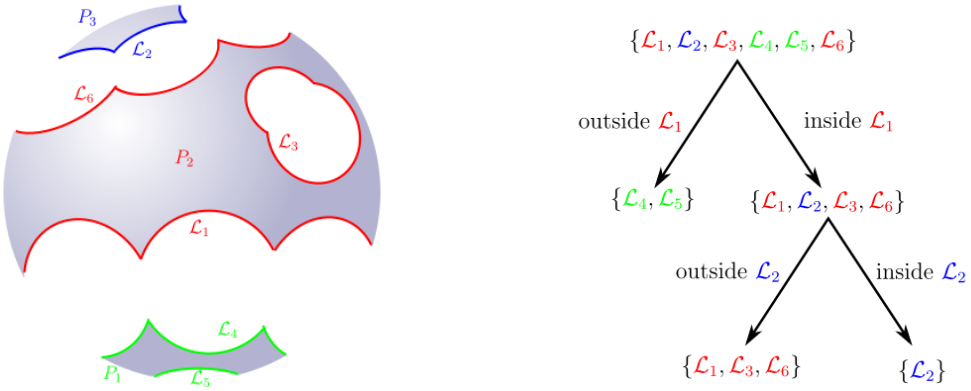

Figure 4. On the left is a brief schematic of all loops on an SAS-sphere. There are six loops on the SAS-sphere $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_5, \mathcal{L}_6, \mathcal{L}_6\}$, and three spherical patches with the boundaries formed by two loops in green $\{\mathcal{L}_4, \mathcal{L}_5\}$, 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.

The problem is to classify all loops into different classes such that each loop in one class is "inside" any other one in the same class. We first divide the set of loops on a sphere into two subsets by checking whether each element of the set is "inside" the first 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 4 for a schematics 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_m based on the formula presented in [10]:

$$P_{-} = P_0 \setminus \bigcap_{x \in K} B_{r_p}(x), \tag{5}$$

where P_0 is the spherical triangle containing P_{-}, K collects all SAS intersection points with distance less than $2r_p$ to x_m, and $B_{r_p}(x)$ denotes the open ball centered at an intersection point x with radius r_p. Notice that P_{-} might be composed of several (in most cases, unique) spherical sub-patches on $\partial B_{r_p}(x_m)$. The spherical triangle P_0 is as illustrated by the blue patch in Figure 3 and the surface that is cut out by the balls $B_{r_p}(x)$ is due to the so-called self-intersection of the surface.

We first calculate all loops forming the boundary of P_{-} each having a data structure like 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 Molecular Surfaces

With the above data structures of different molecular components, we are ready to construct different molecular surfaces, i.e., assemble these data structures. The cSAS is composed of all spherical patches on each SAS-sphere, which have been calculated in Section 3.2. For the eSAS, 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.

Figure 5. 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.

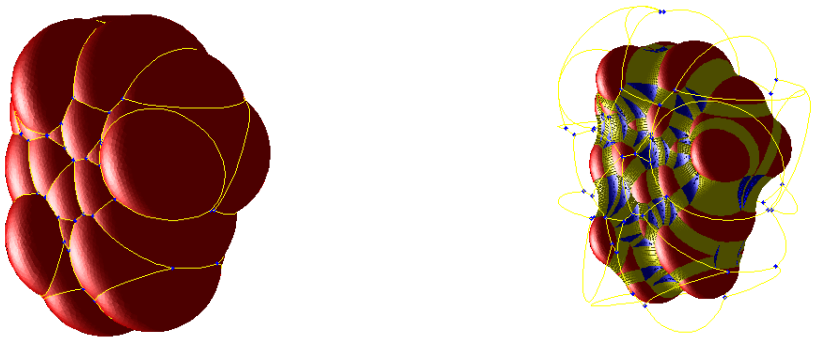

Figure 6. The SAS and the SES of caffeine with probe radius $r_p = 1.5$Å. On the left, the SAS is composed of spherical patches in red, circular arcs in yellow and intersection points in blue. On the right, the patches in red (resp. in yellow or in blue) are convex spherical patches (resp. toroidal patches or concave spherical patches) on the SES, each corresponding to a spherical patch (resp. a circular arc or an intersection point) on the SAS.

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 5) toroidal SES patch and an SAS intersection point corresponds to a concave spherical SES patch obtained by formula (5). A graphical illustration is presented in Figure 6 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 $(\mathcal{L}_1^t, \mathcal{L}_2^t)$: index of one loop composed of four circular arcs or indices of two circles forming the boundary

Double-triangle-shaped Toroidal Patch

- i_l: index of the corresponding circular SAS arc
- $(\mathcal{L}_1^t, \mathcal{L}_2^t)$: indices of 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
- $\mathcal{L}_1^+, \mathcal{L}_2^+, \ldots, \mathcal{L}_{m_1}^+$: indices of the loops forming the boundary of the patch

Concave Spherical SES Patch (or Sub-patch)

- i_I: index of the corresponding SAS intersection point
- $\mathcal{L}_1^-, \mathcal{L}_2^-, \ldots, \mathcal{L}_{m_2}^-$: indices of the 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 of an SAS loop containing both the corresponding circular arcs (or circles) and the index of the sphere on which the loop lies.

4. MESHING ALGORITHM

The SAS is composed of spherical patches like the VdW surface. In consequence, meshing the SAS or the VdW surface can be reduced to develop 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 fast meshing algorithm of molecular surface consisting of two sub-algorithms respectively for meshing a spherical patch and meshing a toroidal patch.

4.1. Boundary Division

We first give a strategy to divide the boundary of a (toroidal or spherical) patch on the SAS or the SES, which ensures that the meshes of two neighboring patches match, 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 (we use a value of $\alpha_0 = 60°$ in the numerical examples) 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:

$$N_{l_m} = \max\left\{ \left\lfloor \frac{r_{l_m}\theta_{l_m}}{d} \right\rfloor + 1,\ \left\lfloor \frac{\theta_{l_m}}{\alpha_0} \right\rfloor + 1 \right\}, \tag{6}$$

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.

4.2. Meshing a Spherical Patch

In this subsection, we present an advancing-front method, see [5, 8, 14, 4] for an overview of this technique, for fast meshing a (convex or concave, SAS or SES) spherical patch uniformly, with its center, its radius and its boundary information known. However, any other meshing algorithm can be applied to the spherical patch as long as they conserve the given boundary partition. Therefore, the following algorithm can be replaced with another algorithm of choice.

4.2.1. General Strategy 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 4.1. 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.

4.2.2. Data Structure Denote by $P \in \mathbb{R}^{N_p \times 3}$ the 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 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 by the right endpoint (the starting endpoint) and the left endpoint (the ending endpoint).

Since the front might consist of several loops, we choose one of them as the active loop. This can be due since the initial boundary of the patch is already composed by several loops or, as we will see later, a loop can be split into two loops at any stage of the algorithm. 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 the previous subsubsection 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.

4.2.3. Testing 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 $\overrightarrow{n_e}$ 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. 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 and its projection on the sphere, which is a circular arc, lies on the plane generated by this edge and the center of the sphere. We say that two edges of the mesh "intersect" if their corresponding projected circular arcs intersect on the sphere.

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 $\overrightarrow{\tau_{12}}$.

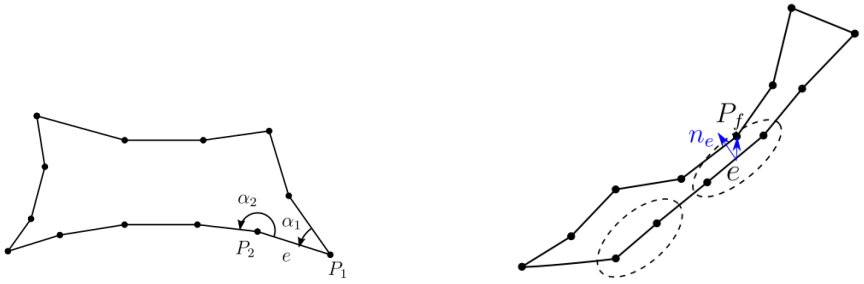

Figure 7. Planar schematics of two criterions for searching a possibly front point: angle criterion (left) and point-edge distance criterion (right).

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 7 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 7. Further, we check if the scalar product between $\overrightarrow{n_e}$ and the vector from $\frac{1}{2}(P_1 + P_2)$ (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, we add P_f to S_f. Then, we take a new front point, check again if all above conditions are satisfied and repeat until we have checked all front points.

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 chose the one denoted by P_0 that has the minimal distance to e, i.e.,

$$P_0 = \text{argmin}_{p \in S_f} \text{dist}(p, e), \tag{7}$$

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 $\triangle P_0P_1P_2$.

4.2.4. Creation of a new Point In the previous subsubsection, 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 the distance from $\frac{1}{2}(P_1 + P_2)$ to P_{test} is $\frac{\sqrt{3}}{2}d$. The existence of P_{test} is guaranteed if and only if the distance from $\frac{1}{2}(P_1 + P_2)$ to the sphere on which the patch lies is smaller than $\frac{\sqrt{3}}{2}d$. This distance is smaller than $\frac{|e|}{2}$ according to

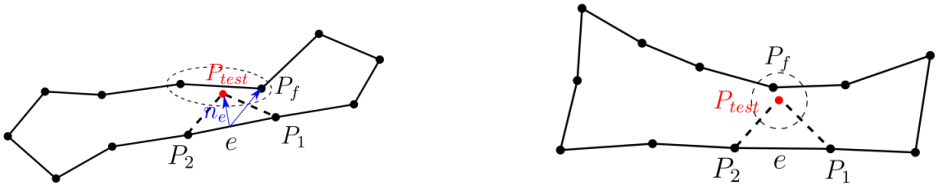

Figure 8. Planar schematics of two criterions for checking if P_{test} is suited as a new vertex of the mesh: point-edge criterion (left) and distance criterion (right).

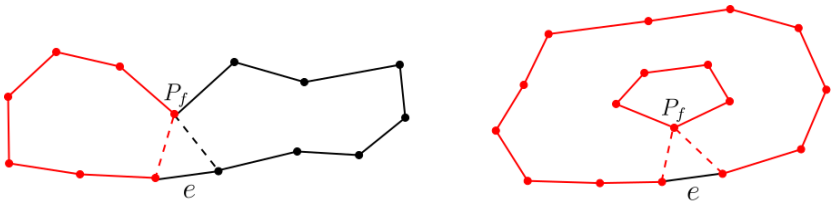

Figure 9. Planar schematics of updating the active loop in two cases: the case where P_0 is a front point on the active loop but not a neighboring active point of e (left) and the case where P_0 is a front point but not on the active loop (right). The new active loop is composed of the red edges and the dashed edges are newly created edges.

the triangle inequality, which guarantees the existence of P_{test} if $|e| < \sqrt{3}d$. Notice that in the case where $|e| = d$, the triangle $\triangle P_{test}P_1P_2$ is equilateral. We check if P_{test} is suited as a new vertex of the mesh based on 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 8. 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 $\frac{1}{2}(P_1 + P_2)$ 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. Then, we take a new front edge, check again if all above conditions are satisfied and repeat until we have checked all front edges.
- *Distance Criterion:*
  Set a distance tolerance between a testing point and a front point to ϵ_d. If there exits a front point P_f with distance to P_{test} smaller than ϵ_d (see the right of Figure 8) 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 (7) with this S_f. Otherwise, we define P_0 as P_{test}.

4.2.5. Updating an Active Loop and Controlling the Size of the Triangles 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, see the left of Figure 9. 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 a larger active loop, see the right of Figure 9. 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 4.1, 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 $\sqrt{3}d$ 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.

Algorithm 1 Advancing-Front Method for Meshing a Spherical Patch

procedure MESHING SPHERICAL PATCH
  Initialization of $A_e,\ N_{ae},\ P,\ N_p,\ T,\ N_t$
  while `the front is not merged` **do**
    for `each active edge` e **do**
      Define the set of possibly suited front points S_f
      if $\alpha_1 < \epsilon_\alpha$ **then** ▷ Angle Criterion
        Add P_1 to S_f
      else if $\alpha_2 < \epsilon_\alpha$ **then**
        Add P_2 to S_f
      end if
      for `each front point` P_f **do** ▷ Point-Edge Criterion
        if dist$(P_f, e) < |e| + \epsilon_e$ **then**
          Add P_f to S_f if $\triangle P_f P_1 P_2$ doesn't "intersect" the front
        end if
      end for
      if S_f is nonempty **then**
        $P_0 = \text{argmin}_{p \in S_f} \text{dist}(p, e)$
        break
      else
        Create a new point P_{test}
        for `each front edge` e_f **do** ▷ Point-Edge Criterion for P_{test}
          if dist$(P_{test}, e_f) < |e_f| + \epsilon_e$ **then**
            for `each endpoint` P_f `of` e **do**
              Add P_f to S_f if $\triangle P_f P_1 P_2$ doesn't "intersect" the front
            end for
          end if
        end for
        for `each front point` P_f **do** ▷ Distance Criterion for P_{test}
          if dist$(P_{test}, P_f) < \epsilon_d$ **then**
            Add P_f to S_f if $\triangle P_f P_1 P_2$ doesn't "intersect" the front
          end if
        end for
        if S_f is nonempty **then**
          $P_0 = \text{argmin}_{p \in S_f} \text{dist}(p, e)$
          break
        else
          $P_0 = P_{test}$
        end if
      end if
      Create a new triangle $\triangle P_0 P_1 P_2$ and update $A_e,\ N_{ae},\ P,\ N_p,\ T,\ N_t$
    end for
  end while
  return $P,\ N_p,\ T,\ N_t$
end procedure

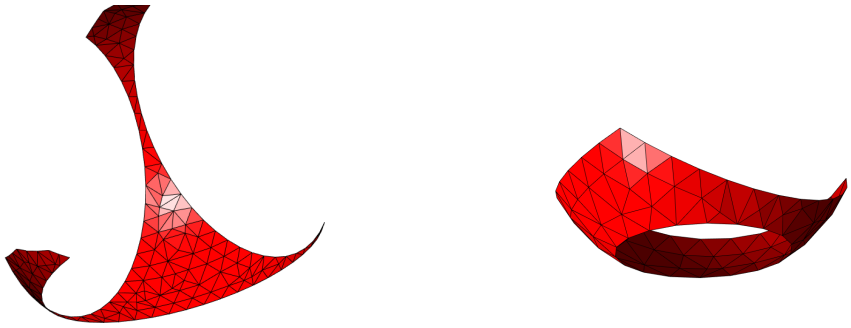

Figure 10. Meshing two spherical patches.

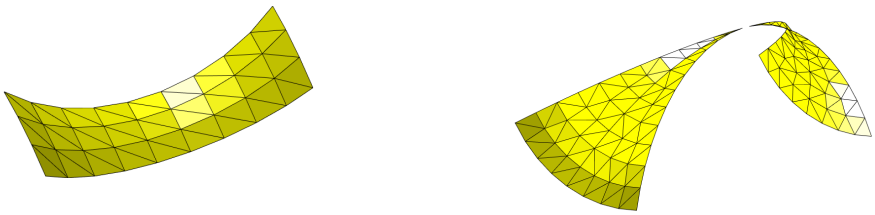

Figure 11. Meshing a rectangle-shaped patch (left) and a double-triangle-shaped patch (right).

4.3. Meshing a Toroidal Patch

To mesh a toroidal SES patch, we should distinguish the cases of a rectangle-shaped patch and a double-triangle-shaped patch. We parameterize a rectangle-shaped patch by defining a mapping from this patch to a rectangle. Given the boundary division of the rectangle, we uniformly mesh it and then map the vertices of the mesh back to the patch to obtain the mesh of the rectangle-shaped patch. Similarly, we parameterize a double-triangle-shaped patch by defining a mapping from this patch to two isosceles triangles. Then, we mesh these two isosceles triangles respecting the given boundary division and map the vertices of the mesh back to the patch to obtain the mesh of the double-triangle-shaped patch.

5. IMPLEMENTATION

5.1. Illustrations

We illustrate the mesh of a spherical patch that is obtained using the advancing-front algorithm in Figure 10. We also illustrate the mesh of a rectangle-shaped toroidal patch on the left of Figure 11 and the mesh of a double-triangle-shaped toroidal patch on the right. In addition, we present the SESs of some artificial and non-artificial molecules generated by the above meshing algorithm (Figure 12 and Figure 13).

5.2. 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

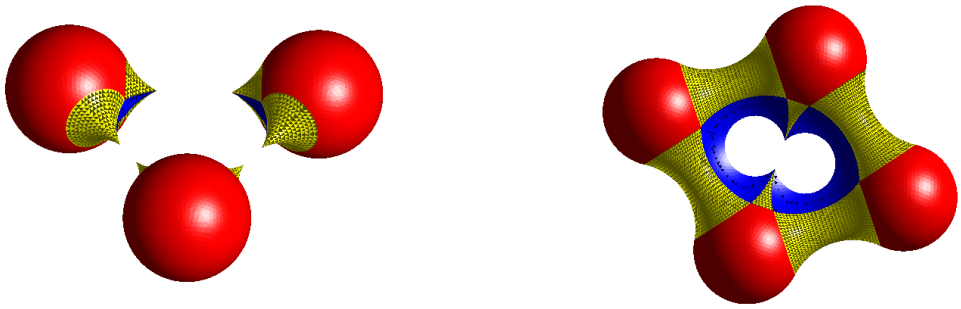

Figure 12. The SES of three artificial spheres (left) and the SES of four artificial spheres (right) where $r_p = 1.75$Å.

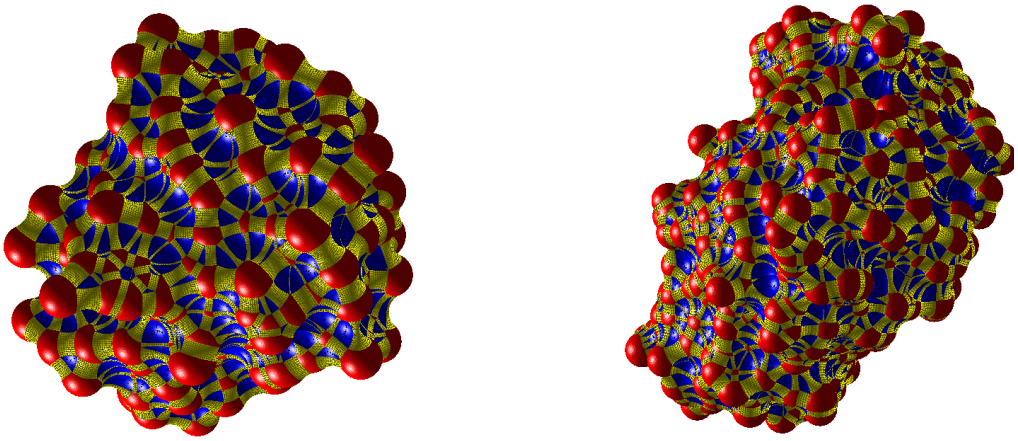

Figure 13. The SES of molecule 1B17 with 485 atoms (left) and the SES of molecule 101M with 1414 atoms (right) where $r_p = 1.75$Å.

of refinement is quite efficient with the complexity proportional to the number of triangles in the mesh.

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 allowed us to develop a fast 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. Furthermore, it is worth to mention that each vertex of the created mesh lies exactly on the molecular surface.

The obtained meshes allow then to discretize the partial differential equations used in implicit solvation models that are commonly reformulated as integral equations on the molecular surface, or to visualize the molecular surface.

ACKNOWLEDGEMENT

The authors thank Pascal Frey, Filippo Lipparini, Yvon Maday and Benedetta Mennucci for the encouraging and fruitful discussions we had in the last months.

REFERENCES

1. J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324:446–449, Dec. 1986.
2. F. A. Bulat, A. Toro-Labbé, T. Brinck, J. S. Murray, and P. Politzer. Quantitative analysis of molecular surfaces: areas, volumes, electrostatic potentials and average local ionization energies. *Journal of molecular modeling*, 16(11):1679–1691, 2010.
3. M. L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16(5):548–558, Oct 1983.
4. P. J. Frey, H. Borouchaki, and P.-L. George. 3d delaunay mesh generation coupled with an advancing-front approach. *Computer methods in applied mechanics and engineering*, 157(1):115–131, 1998.
5. P. L. George and É. Seveno. The advancing-front mesh generation method revisited. *International Journal for Numerical Methods in Engineering*, 37(21):3605–3619, 1994.
6. P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18(3):199–210, 2002.
7. B. Lee and F. M. Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology*, 55(3):379–IN4, 1971.
8. P. Möller and P. Hansbo. On advancing front mesh generation in three dimensions. *International Journal for Numerical Methods in Engineering*, 38(21):3551–3569, 1995.
9. J. Parulek and I. Viola. Implicit representation of molecular surfaces. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE*, pages 217–224. IEEE, 2012.
10. C. Quan and B. Stamm. Mathematical Analysis and Calculation of Molecular Surfaces. working paper or preprint, Sept. 2015.
11. A. K. Rappé, C. J. Casewit, K. Colwell, W. Goddard Iii, and W. 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.
12. F. M. Richards. Areas, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering*, 6:151–176, 1977.
13. M. F. Sanner, A. J. Olson, and J.-C. Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, 1996.
14. J. Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science*, 1(1):41–52, 1997.
15. J. Tomasi, B. Mennucci, and R. Cammi. Quantum mechanical continuum solvation models. *Chemical reviews*, 105(8):2999–3094, 2005.
16. N. R. Voss and M. Gerstein. 3v: cavity, channel and cleft volume calculator and extractor. *Nucleic acids research*, page gkq395, 2010.