SIAM J. SCI. COMPUT.
Vol. 40, No. 2, pp. B507–B527
© 2018 Society for Industrial and Applied Mathematics

EFFICIENT AND QUALIFIED MESH GENERATION FOR GAUSSIAN MOLECULAR SURFACE USING ADAPTIVE PARTITION AND PIECEWISE POLYNOMIAL APPROXIMATION*

TIANTIAN LIU†, MINXIN CHEN‡, AND BENZHUO LU§

Abstract. Recent developments for mathematical modeling and numerical simulation of biomolecular systems raise new demands for qualified, stable, and efficient surface meshing, especially in implicit-solvent modeling [B. Z. Lu et al., *Commun. Comput. Phys.*, 3 (2008), pp. 973–1009]. In our former work, we have developed an algorithm for manifold triangular meshing for large Gaussian molecular surfaces, TMSmesh [M. Chen and B. Lu, *J. Chem. Theory Comput.*, 7 (2011), pp. 203–212; M. Chen, B. Tu, and B. Lu, *J. Molecular Graphics Model.*, 38 (2012), pp. 411–418]. In this paper, we present new algorithms to greatly improve the meshing efficiency and qualities, and implement them into a new program version, TMSmesh 2.0. In TMSmesh 2.0, in the first step, a new adaptive partition and estimation algorithm is proposed to locate the cubes in which the surface is approximated by a piecewise trilinear surface with controllable precision. Then, the piecewise trilinear surface is divided into single valued pieces by tracing along the fold curves, which ensures that the generated surface meshes are manifolds. Numerical test results show that TMSmesh 2.0 is capable of handling arbitrary sizes of molecules and achieves ten to hundreds of times speedup over the previous algorithm. In all of our extensively tested molecules, the resulting surface meshes are all manifolds and can be used in boundary element method (BEM) and finite element method (FEM) simulation. The binary version of TMSmesh 2.0 is downloadable from the web page http://lsec.cc.ac.cn/~lubz/Meshing.html.

Key words. surface mesh generation, Gaussian surface, triangulation, adaptive partition, trilinear polynomial

AMS subject classifications. 32B25, 65M50, 68N01

DOI. 10.1137/16M1099704

1. Introduction.

Molecular surface mesh generation is a prerequisite for using the boundary element method (BEM) and the finite element method (FEM) in implicit-solvent modeling (e.g., see a review in [25]). Recent developments in implicit-solvent modeling of biomolecular systems raise new demands for qualified, stable, and efficient surface meshing. The main concerns for improvement on existing methods for molecular surface mesh generation are efficiency, robustness, and mesh quality. Efficiency is necessary for simulations/computations requiring frequent mesh generation or requiring meshing for large systems. Robustness here means the meshing method is stable and can treat various, even arbitrary, sizes of molecular systems within computer power limitations. Mesh quality relates to mesh smoothness (avoiding sharp solid angles, etc.), uniformness (avoiding elements with very sharp angles

*Submitted to the journal's Computational Methods in Science and Engineering section October 21, 2016; accepted for publication (in revised form) January 22, 2018; published electronically April 5, 2018.

http://www.siam.org/journals/sisc/40-2/M109970.html

Funding: This work was funded by the Science Challenge Project (SCP TZ2016003-1), National Key Research and Development Program China (2016YFB0201304), China NSF (NSFC 91530102, NSFC 21573274, NSFC 11301368, NSFC 11404300), and NSF of Jiangsu Province (BK20130278).

†Department of Mathematics, Soochow University, Suzhou 215006, China, and State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (liutt@lsec.cc.ac.cn).

‡Corresponding author. Department of Mathematics, Soochow University, Suzhou 215006, China (chenminxin@suda.edu.cn).

§State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (bzlu@lsec.cc.ac.cn).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

or zero area), topological correctness (manifoldness, avoiding isolated vertices, element intersection, single-element-connected edges, etc.), and fidelity (faithful to the original defined molecular surface). The quality requirement is critical for some numerical techniques, such as the FEM, to achieve converged and reasonable results, which makes it a more demanding task in this aspect than the mesh generations only for the purposes of visualization or some structural geometry analysis.

There are various kinds of definitions for molecular surface, including the van der Waals (VDW) surface, the solvent accessible surface (SAS) [21], the solvent excluded surface (SES) [30], the minimal molecular surface [1], the molecular skin surface [16], and the Gaussian surface. The VDW surface is defined as the surface of the union of the spherical atomic surfaces with VDW radius of each atom within the molecule. The SAS and SES are represented by the trajectory of the center and the interboundary of a rolling probe on the VDW surface, respectively. The minimal molecular surface is defined as the result of the minimization of a type of surface energy. The molecular skin surface is the envelope of an infinite family of spheres derived from atoms by convex combination and shrinking. The Gaussian surface is defined as a level set of the summation of the Gaussian kernel functions, for which descriptions of the specific forms will be given in the next section.

For SAS and SES, numerous works have been committed to the computation of the molecular surface in the literature. In 1983, Connolly proposed algorithms to calculate the molecular surface and SAS analytically [9, 10]. In 1995, a popular program, GRASP, for visualizing molecular surfaces was presented [29]. An algorithm named SMART for triangulating SAS into curvilinear elements was proposed by Zauhar [40]. The software MSMS was proposed by Sanner, Olson, and Spehner in 1996 to mesh the SES and is a widely used program for molecular surface triangulation due to its high efficiency [32]. In 1997, Vorobjev and Hermans proposed SIMS, a method of calculating a smooth invariant molecular dot surface, in which an exact method for removing self-intersecting parts and smoothing the singular regions of the SES was presented [35]. Ryu, Park, and Kim proposed a method based on beta shapes [31], which is a generalization of alpha shapes [17]. Can, Chen, and Wang proposed LSMS to generate the SES on grid points using level-set methods [3]. In 2009, a program, EDTsurf, based on LSMS was proposed for generating the VDW surface, SES, and SAS [37]. A ray-casting-based algorithm, NanoShaper, was proposed to generate SES, skin surface, and Gaussian surface in 2013 [12].

For skin surface, Chavent, Levy, and Maigret presented MetaMtal to visualize the molecular skin surface using the ray-casting method [4], and Cheng and Shi used restricted union of balls to generate mesh for molecular skin surface [8]. For minimal surface, Bates, Wei, and Zhao [1] constructed a surface-based energy functional and used minimization and isosurface extraction processes to obtain a so-called minimal molecular surface.

For the Gaussian surface, existing techniques for triangulating an implicit surface can be used to mesh the Gaussian surface. These methods are divided into two main categories: spatial partition and continuation methods. The well-known marching cube method [24] and dual contouring method [19] are examples of the spatial partition methods. In 2006, Zhang, Xu, and Bajaj [42] used a modified dual contouring method to generate meshes for biomolecular structures. A later tool, GAMer [38], was developed for both the generation and improvement of the Gaussian surface meshes. An efficient mesh generation algorithm accelerated by multicore CPU and GPU was also proposed in 2013 [22].

Most of those softwares have some issues according to the above-mentioned crite-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

ria for mesh generation; e.g., MSMS and GAMer generate many nonmanifold defects in the mesh, fidelity is not well preserved for the EDTsurf and GAMer surfaces, and some traditional grid-based methods require a lot of memory for molecules with large size. More detailed comparison and discussion of the software can be found in [23]. As MSMS is a most commonly used software in this area, we will still use it as a main reference for our new algorithm in this article.

In 2011, we proposed an algorithm and implemented it in the program TMSmesh for triangular meshing of the Gaussian surface [6, 7, 23]. The trace technique, which is a generalization of the adaptive predictor-corrector technique, is used in TMSmesh to connect sampled surface points. TMSmesh contains two steps. The first step is to compute the intersection points between the molecular Gaussian surface and the lines parallel to the x-axis. In the second step, the sampled surface points are connected through three algorithms to form loops, and the whole closed manifold surface is decomposed into a collection of patches enclosed by loops on the surface. The patches are finally divided into single valued pieces, which means that on each piece of this type, each component of the normal direction (ϕ_x, ϕ_y, ϕ_z) does not change sign. So these pieces can be treated as two-dimensional polygons and can be easily triangulated through standard triangulation algorithms. In TMSmesh, there are no problems of overlapping, gap filling, or selecting seeds that need to be considered in traditional continuation methods. TMSmesh performs well in the following aspects. First, TMSmesh is robust. TMSmesh succeeds in generating surface meshes for biomolecules comprised of more than one million atoms. Second, the meshes produced by TMSmesh have good qualities (uniformness and manifoldness). Third, the generated surface mesh preserves the original molecular surface features and properties (topology, surface area and enclosed volume, and local curvature). However, as to the aspect of computational efficiency, although the computational complexity is linear with respect to the number of atoms as shown in [7], the overall low efficiency of TMSmesh still needs to be improved.

In this paper, we propose a new algorithm and updated program version, TMSmesh 2.0, to mesh the Gaussian surface efficiently. TMSmesh 2.0 is capable of handling arbitrary sizes of molecules and achieves ten to hundreds of times speedup over TMSmesh.

This paper is organized as follows. The new algorithm for triangulating the Gaussian surface is introduced in section 2. In section 3, some examples and applications are presented. Finally, section 4 gives some concluding remarks.

2. Meshing algorithm. The Gaussian surface is defined as a level set of the summation of Gaussian kernel functions,

$$\{\vec{x} \in R^3, \phi(\vec{x}) = c\}, \tag{1}$$

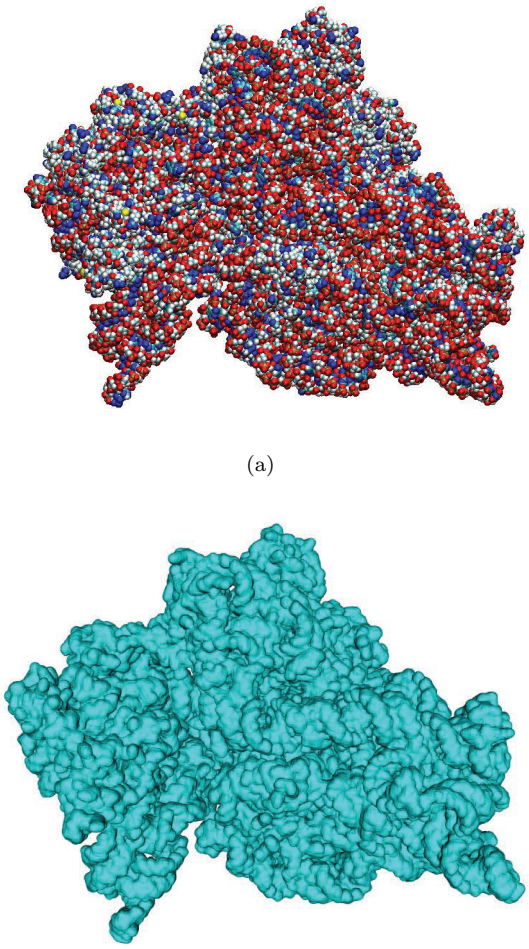
where

$$\phi(\vec{x}) = \sum_{i=1}^{N} e^{-D(\|\vec{x}-\vec{x}_i\|^2 - r_i^2)} \tag{2}$$

and $\vec{x}_i$ and r_i are the location and radius of the ith atom. The radius r_i is usually the VDW radius of the ith atom. D is the decay rate of the Gaussian kernel. c is the isovalue, and it controls the volume enclosed by the Gaussian surface. These two parameters, D and c, can be chosen properly to make the Gaussian surface approximate the SES, SAS, and VDW surfaces well [23]. Figure 1 shows an example of a Gaussian surface. Figure 1(a) shows the atoms included in the molecule, and

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



(a)

(b)

FIG. 1. *An example of Gaussian molecular surface. The molecule is* 30*S Ribosome.* (a) *shows the VDW surface, and* (b) *shows the Gaussian molecular surface generated by TMSmesh* [7] *with parameter D and c is* 0.5 *and* 1.0*, respectively. All coordinates and corresponding radii are drawn from the PQR file that is transformed from the PDB file,* 1*FJF, using the PDB*2*PQR tool* [13].

Figure 1(b) shows the Gaussian molecular surface. The definition of the Gaussian surface is quite different from many other types of molecular surface definitions. But so far, it is still arguable which one is the "correct one." The Gaussian surface and the other mentioned surface types are all widely used in the community [23]. Comparing with the other definitions, the Gaussian surface has the following advantages:

- The Gaussian surface is smooth.
- The Gaussian surface provides a realistic representation of the electron density of a molecule as compared to other molecular surface definitions [14].
- The Gaussian surface is well established [6, 22, 38, 42] and has a wide range of applications in computational biology, such as docking problems [26], molecular shape comparisons [18], calculating SAS areas [36], and the generalized Born models [39].

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

In this work, we focus on developing algorithms for the triangular mesh generation of the Gaussian molecular surface.

In this section, we describe the algorithms to construct the triangular surface meshes. The inputs of our method are PQR files which contain a list of centers and radii of atoms. The output of our method are OFF files which contain the triangular meshes. The algorithm outline is as follows:

- First, the space is adaptively divided into cubes, and an algorithm of dividing cubes and estimating the error between the Gaussian surface and approximated trilinear polynomial in each cube is developed. With this algorithm, the Gaussian surface is approximated by the piecewise trilinear surface.
- Second, in each cube, the trilinear surface is divided into a collection of single valued pieces in the x, y, and z directions by tracing along fold curves (in the trilinear surface case the fold curve can be directly calculated analytically).
- Third, each single valued piece is triangulated by standard polygon triangulation algorithms, such as the ear clipping algorithm [15, 27], using monotone polygons [11], Seidel's decomposition algorithm [33], or Chazelle's triangulation method [5].

In the following subsections, each stage is described in detail.

2.1. Approximating the Gaussian surface by piecewise trilinear surface. In this stage, the space is divided into cubes adaptively, and in each final cube, the Gaussian surface is close to a trilinear surface whose error is controllable. Initially, the molecule is placed in a three-dimensional orthogonal grid consisting of $n_x \times n_y \times n_z$ cubes. The initial grid is very coarse. Then the grid is refined adaptively by the following estimation and division steps:

- Step 1. In each cube, $\phi(x, y, z)$ is approximated by an nth-degree polynomial $\tilde{P}(x, y, z)$; i.e., the Gaussian surface $\phi(x, y, z) = c$ is replaced by the polynomial surface $\tilde{P}(x, y, z) = c$.
- Step 2. The lower and the upper bounds of $\tilde{P}(x, y, z)$, denoted by L and U, in each cube are estimated. If the isovalue c belongs to $[L, U]$, the cube has an intersection with the surface $\tilde{P}(x, y, z) = c$, and we go to Step 3; otherwise, the cube is abandoned.
- Step 3. Divide each remaining cube into 8 smaller child cubes, and compute the expression of $\tilde{P}(x, y, z)$ in each child cube. When the child cubes become smaller, the coefficients of higher order terms (higher than the linear order) of $\tilde{P}(x, y, z)$ go to zero. If they are under some user-specified bound, approximate $\tilde{P}(x, y, z)$ by trilinear polynomial; otherwise, go to Step 2.

With the above processes, the Gaussian surface finally is approximated by piecewise trilinear surfaces in cubes with different sizes. In the following subsections, we explain the details of the above estimation and division process.

2.1.1. Approximation with *n*th-degree polynomial. First, without loss of generality, we only consider the case of $D = 1$; then (2) is written in the following form:

$$\phi(\vec{x}) = \sum_{i=1}^{N} e^{-(\|\vec{x}-\vec{x}_i\|^2 - r_i^2)} = \sum_{i=1}^{N} e^{r_i^2} e^{-(x-x_i)^2} e^{-(y-y_i)^2} e^{-(z-z_i)^2}. \tag{3}$$

In an arbitrary cube $[a, b] \times [c, d] \times [e, f]$, equation (3) can be approximated by

$$P(x, y, z) = \sum_{i=1}^{N} e^{r_i^2} P_n(x, x_i, a, b) Q_n(y, y_i, c, d) R_n(z, z_i, e, f), \tag{4}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

where

$$P_n(x,x_i,a,b)=\sum_{j=0}^{n}\alpha_j(x_i,a,b)L_j\left(\frac{2x-(a+b)}{b-a}\right), \tag{5}$$

$$Q_n(y,y_i,c,d)=\sum_{j=0}^{n}\beta_j(y_i,c,d)L_j\left(\frac{2y-(c+d)}{d-c}\right), \tag{6}$$

$$R_n(z,z_i,e,f)=\sum_{j=0}^{n}\gamma_j(z_i,e,f)L_j\left(\frac{2z-(e+f)}{f-e}\right), \tag{7}$$

and

$$\alpha_j=\frac{2}{b-a}\int_a^b\phi\left(\frac{2x-(a+b)}{b-a}\right)L_j\left(\frac{2x-(a+b)}{b-a}\right)dx,$$

$$\beta_j=\frac{2}{d-c}\int_c^d\phi\left(\frac{2y-(c+d)}{d-c}\right)L_j\left(\frac{2y-(c+d)}{d-c}\right)dy,$$

$$\gamma_j=\frac{2}{f-e}\int_e^f\phi\left(\frac{2z-(e+f)}{f-e}\right)L_j\left(\frac{2z-(e+f)}{f-e}\right)dz,$$

$L_j(\cdot)$ is Legendre polynomial of order j, and n is set as 3 in our work. However, $P(x,y,z)$ is not continuous between neighbored cubes, so we do the following corrections of $P(x,y,z)$ to make $P(x,y,z)$ be C^0 continuous in the whole domain. For one component $P_n(x,x_i,a,b)$, we introduce two variables $\epsilon_0(x_i,a,b)$ and $\epsilon_1(x_i,a,b)$ as follows to do corrections:

(8)

$$\begin{aligned}\tilde{P}_n(x,x_i,a,b)&=P_n(x,x_i,a,b)+\epsilon_0(x_i,a,b)L_{n-1}\left(\frac{2x-(a+b)}{b-a}\right)\\&\quad+\epsilon_1(x_i,a,b)L_n\left(\frac{2x-(a+b)}{b-a}\right)\\&=\alpha_0(x_i,a,b)L_0\left(\frac{2x-(a+b)}{b-a}\right)+\cdots+\alpha_n(x_i,a,b)L_n\left(\frac{2x-(a+b)}{b-a}\right)\\&\quad+\epsilon_0(x_i,a,b)L_{n-1}\left(\frac{2x-(a+b)}{b-a}\right)+\epsilon_1(x_i,a,b)L_n\left(\frac{2x-(a+b)}{b-a}\right).\end{aligned}$$

The following two equations make $\tilde{P}_n(x,x_i,a,b)$ equal to the x component of $\phi(\vec{x})$ on the boundary of the box and C^0 continuous along the x directions:

$$\begin{cases}\tilde{P}_n(a,x_i,a,b)=e^{-(a-x_i)^2}, & \text{(9a)}\\ \tilde{P}_n(b,x_i,a,b)=e^{-(b-x_i)^2}. & \text{(9b)}\end{cases}$$

$\epsilon_0(x_i,a,b)$ and $\epsilon_1(x_i,a,b)$ can be easily solved from (9). Then $\tilde{P}_n(x,x_i,a,b)$ is written as follows:

$$\tilde{P}_n(x,x_i,a,b)=\sum_{j=0}^{n}\tilde{\alpha}_j(x_i,a,b)L_j\left(\frac{2x-(a+b)}{b-a}\right), \tag{10}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

where

$$\tilde{\alpha}_j(x_i,a,b)=\begin{cases}\alpha_j(x_i,a,b), & j<n-1,\\ \alpha_{n-1}(x_i,a,b)+\epsilon_0(x_i,a,b), & j=n-1,\\ \alpha_n(x_i,a,b)+\epsilon_1(x_i,a,b), & j=n.\end{cases} \tag{11}$$

After the above correction, $\tilde{P}_n(x,x_i,a,b)$ is the best least square approximation for the x component of $\phi(\vec{x})$ in the space spanned by $\{L_i,\ i=0,\dots,n-2\}$, and it is also C^0 continuous on the boundaries of the cubes along the x direction. The same method should be used to correct $Q_n(y,y_i,c,d)$ and $R_n(z,z_i,e,f)$ to make $\tilde{P}(x,y,z)$ be C^0 continuous along the y,z directions. We have

$$\tilde{Q}_n(y,y_i,c,d)=\sum_{j=0}^{n}\tilde{\beta}_j(y_i,c,d)L_j\left(\frac{2y-(c+d)}{d-c}\right), \tag{12}$$

$$\tilde{R}_n(z,z_i,e,f)=\sum_{j=0}^{n}\tilde{\gamma}_j(z_i,e,f)L_j\left(\frac{2z-(e+f)}{f-e}\right), \tag{13}$$

where the forms of $\tilde{\beta}_j(y_i,c,d)$ and $\tilde{\gamma}_j(z_i,e,f)$ are similar to $\tilde{\alpha}_j(x_i,a,b)$ in (11). Then the new nth polynomial is written as

$$\tilde{P}(x,y,z)=\sum_{i=1}^{N}e^{r_i^2}\tilde{P}_n(x,x_i,a,b)\tilde{Q}_n(y,y_i,c,d)\tilde{R}_n(z,z_i,e,f) \tag{14}$$

for $x\in[a,b], y\in[c,d], z\in[e,f]$. In practical computation of $\tilde{P}(x,y,z)$, we only need to compute the summation in (14) with respect to the neighborhood $\{x_i,y_i,z_i\}$ of the cube $[a,b]*[c,d]*[e,f]$, since the kernel $e^{-||\vec{x}-\vec{x}_i||^2}$ decays very quickly when $||\vec{x}-\vec{x}_i||$ becomes large.

2.1.2. Estimation of upper and lower bounds of $\tilde{P}(x,y,z)$. In order to rule out the cubes having no surface points, the lower and upper bounds of $\tilde{P}(x,y,z)$ in the cube are estimated. $\tilde{P}(x,y,z)$ in (14) can be written in the form of the product of tensors:

$$\tilde{P}(x,y,z)=A\bar{\times}_1\vec{L}\left(\frac{2x-(a+b)}{b-a}\right)\bar{\times}_2\vec{L}\left(\frac{2y-(c+d)}{d-c}\right)\bar{\times}_3\vec{L}\left(\frac{2z-(e+f)}{f-e}\right), \tag{15}$$

where $A=\sum_{i=1}^{N}e^{r_i^2}B_i$ and

$$\begin{cases}
B_i=b_i^{(1)}\otimes b_i^{(2)}\otimes b_i^{(3)}, & \text{(16a)}\\
b_i^{(1)}=(\tilde{\alpha}_0(x_i,a,b),\tilde{\alpha}_1(x_i,a,b),\dots,\tilde{\alpha}_n(x_i,a,b)), & \text{(16b)}\\
b_i^{(2)}=(\tilde{\beta}_0(y_i,c,d),\tilde{\beta}_1(y_i,c,d),\dots,\tilde{\beta}_n(y_i,c,d)), & \text{(16c)}\\
b_i^{(3)}=(\tilde{\gamma}_0(z_i,e,f),\tilde{\gamma}_1(z_i,e,f),\dots,\tilde{\gamma}_n(z_i,e,f)), & \text{(16d)}\\
\vec{L}\left(\frac{2x-(a+b)}{b-a}\right)=\left(L_0\left(\frac{2x-(a+b)}{b-a}\right),\dots,L_n\left(\frac{2x-(a+b)}{b-a}\right)\right), & \text{(16e)}\\
\vec{L}\left(\frac{2y-(c+d)}{d-c}\right)=\left(L_0\left(\frac{2y-(c+d)}{d-c}\right),\dots,L_n\left(\frac{2y-(c+d)}{d-c}\right)\right), & \text{(16f)}\\
\vec{L}\left(\frac{2z-(e+f)}{f-e}\right)=\left(L_0\left(\frac{2z-(e+f)}{f-e}\right),\dots,L_n\left(\frac{2z-(e+f)}{f-e}\right)\right). & \text{(16g)}
\end{cases}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

$\otimes$ is the product of tensor. $\bar{\times}_k$ is the k-mode (vector) product of a tensor $X \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with a vector $V \in \mathbb{R}^{I_k}$ denoted by $X\bar{\times}_k V$ and is of size $I_1 \times \cdots \times I_{k-1} \times I_{k+1} \times \cdots \times I_3$ [20]. Its $i_1 \cdots i_{k-1} i_{k+1} \cdots i_3$ entry is as follows:

$$(X\bar{\times}_k V)_{i_1 \cdots i_{k-1} i_{k+1} \cdots i_3} = \sum_{i_k=1}^{I_k} x_{i_1} x_{i_2} x_{i_3} v_{i_k}. \tag{17}$$

A is a three-dimensional tensor whose size is $(n+1) \times (n+1) \times (n+1)$, where n is the degree of the polynomial $\tilde{P}(x, y, z)$. To get the lower and upper bounds of $\tilde{P}(x, y, z)$, first, the main part of $\tilde{P}(x, y, z)$ is obtained by doing singular value decomposition (SVD) for A. Second, the upper and the lower bounds of the main part and the remainder are estimated, respectively.

Here we use SVD for A to approximate $\tilde{P}(x, y, z)$ by a multiplication of three polynomials in x, y, z, respectively. Taking $n = 3$ and $A = [a_{ijk}]_{4\times4\times4}$, for example, the algorithm of SVD is as follows.

Step 1. Transform A into a two-dimensional matrix:

$$A_1 = \begin{bmatrix} a_{111} & \dots & a_{141} & a_{211} & \dots & a_{241} & a_{311} & \dots & a_{341} & a_{411} & \dots & a_{441} \\ a_{112} & \dots & a_{142} & a_{212} & \dots & a_{242} & a_{312} & \dots & a_{342} & a_{412} & \dots & a_{442} \\ a_{113} & \dots & a_{143} & a_{213} & \dots & a_{243} & a_{313} & \dots & a_{343} & a_{413} & \dots & a_{443} \\ a_{114} & \dots & a_{144} & a_{214} & \dots & a_{244} & a_{314} & \dots & a_{344} & a_{414} & \dots & a_{444} \end{bmatrix}. \tag{18}$$

Step 2. Do SVD to A_1,

$$A_1 = UDV^*, \tag{19}$$

where $U = (\vec{u}_1, \vec{u}_2, \vec{u}_3, \vec{u}_4)$ is a 4×4 matrix, V^* is a 4×16 matrix, and

$$D = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}. \tag{20}$$

If j satisfies

$$\min\left\{ j : \sum_{i=1}^{j} \sigma_i \geq 0.99 \sum_{i=1}^{4} \sigma_i \right\}, \tag{21}$$

we reserve $\sigma_1, \dots, \sigma_j$ and abandon $\sigma_{j+1}, \dots, \sigma_4$.

Step 3. Transform each row of V^* into a square matrix. If

$$V^* = \begin{bmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,16} \\ v_{2,1} & v_{2,2} & \dots & v_{2,16} \\ v_{3,1} & v_{3,2} & \dots & v_{3,16} \\ v_{4,1} & v_{4,2} & \dots & v_{4,16} \end{bmatrix}, \tag{22}$$

we can transform the ith row of V^* into a 4×4 matrix denoted by V_i:

$$V_i = \begin{bmatrix} v_{i,1} & v_{i,2} & v_{i,3} & v_{i,4} \\ v_{i,5} & v_{i,6} & v_{i,7} & v_{i,8} \\ v_{i,9} & v_{i,10} & v_{i,11} & v_{i,12} \\ v_{i,13} & v_{i,14} & v_{i,15} & v_{i,16} \end{bmatrix}. \tag{23}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Step 4. Do SVD for V_i, respectively,

$$V_i = W_i D_i Z_i, \tag{24}$$

where $W_i = (\vec{w_1^i}, \vec{w_2^i}, \vec{w_3^i}, \vec{w_4^i}), D_i = diag(d_1^i, d_2^i, d_3^i, d_4^i), Z_i = (\vec{z_1^i}, \vec{z_2^i}, \vec{z_3^i}, \vec{z_4^i})^T$. If j_i satisfies

$$\min\left\{ j_i : \sum_{k=1}^{j_i} d_k^i \geq 0.99 \sum_{k=1}^{4} d_k^i \right\},$$

$d_1^i, \ldots, d_{j_i}^i$ are reserved and $d_{j_i+1}^i, \ldots, d_4^i$ are abandoned. Therefore, V_i can be approximated by the following formula:

$$V_i \approx d_1^i \vec{w_1^i} \otimes \vec{z_1^i} + \cdots + d_{j_i}^i \vec{w_{j_1}^i} \otimes \vec{z_{j_i}^i}. \tag{25}$$

Through the above calculation, A can be approximated by

$$A \approx \sum_{i=1}^{j} \sigma_i \vec{u}_i \otimes \left(\sum_{k=1}^{j_i} d_i^k \vec{w_i^k} \otimes \vec{z_i^k} \right). \tag{26}$$

The summation on the right-hand side of (26) is denoted by $\tilde{A}$. As a result, A can be split by $A = \tilde{A} + R$, where $\tilde{A}$ is the main part and R is the residue part.

With the above SVD process, $\tilde{P}(x, y, z)$ can be converted to the following form:

$$\tilde{P}(x, y, z) = S(x, y, z) + T(x, y, z), \tag{27}$$

where

$$S(x, y, z) = \tilde{A} \times_1 \vec{L}\left(\frac{2x - (a+b)}{b-a} \right) \times_2 \vec{L}\left(\frac{2y - (c+d)}{d-c} \right) \times_3 \vec{L}\left(\frac{2z - (e+f)}{f-e} \right), \tag{28}$$

$$T(x, y, z) = R \times_1 \vec{L}\left(\frac{2x - (a+b)}{b-a} \right) \times_2 \vec{L}\left(\frac{2y - (c+d)}{d-c} \right) \times_3 \vec{L}\left(\frac{2z - (e+f)}{f-e} \right). \tag{29}$$

For $S(x, y, z)$, the upper bound and lower bound are estimated through the following steps:

$$S(x, y, z) = \sum_{i=1}^{j} \sigma_i \tilde{U}^i(x) \left[\sum_{k=1}^{j_i} d_k^i \tilde{W}_k^i(y) \tilde{Z}_k^i(z) \right], \tag{30}$$

where

$$\begin{cases} \tilde{U}^i(x) = \vec{u}_i \cdot \vec{L}\left(\dfrac{2x - (a+b)}{b-a} \right), & \text{(31a)} \\ \tilde{W}_k^i(y) = \vec{w_k^i} \cdot \vec{L}\left(\dfrac{2y - (c+d)}{d-c} \right), & \text{(31b)} \\ \tilde{Z}_k^i(z) = \vec{z_k^i} \cdot \vec{L}\left(\dfrac{2z - (e+f)}{f-e} \right). & \text{(31c)} \end{cases}$$

First, we estimate the upper bound and lower bound of one-dimensional polynomials $\tilde{W}_k^i(y)$ and $\tilde{Z}_k^i(z)$, respectively. The upper and lower bounds of $\tilde{W}_k^i(y)$ are

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

denoted by M_k^y and m_k^y. And the upper and lower bounds of $\tilde{Z}_k^i(z)$ are denoted by M_k^z and m_k^z. Second, the upper bound and lower bound of $\tilde{W}_k^i(y)\tilde{Z}_k^i(z)$ are estimated by

$$M_k^{yz} = \max\{M_k^y M_k^z, M_k^y m_k^z, m_k^y M_k^z, m_k^y m_k^z\}, \tag{32}$$

$$m_k^{yz} = \min\{M_k^y M_k^z, M_k^y m_k^z, m_k^y M_k^z, m_k^y m_k^z\}. \tag{33}$$

Then the upper bound of $\sum_{k=1}^{j_i} d_k^i \tilde{W}_k^i(y)\tilde{Z}_k^i(z)$ is

$$M_i^{yz} = \sum_{k=1}^{j_i} d_k^i M_k^{yz}, \tag{34}$$

and the lower bound is

$$m_i^{yz} = \sum_{k=1}^{j_i} d_k^i m_k^{yz}. \tag{35}$$

Finally, we estimate the upper bound and the lower bound of $\tilde{U}^i(x)$, which are denoted by M_i^x and m_i^x. Then the bounds of $\tilde{U}^i(x)[\sum_{k=1}^{j_i} d_k^i \tilde{W}_k^i(y)\tilde{Z}_k^i(z)]$ can be estimated by

$$M_i = \max\{M_i^x M_i^{yz}, M_i^x m_i^{yz}, m_i^x M_i^{yz}, m_i^x m_i^{yz}\}, \tag{36}$$

$$m_i = \min\{M_i^x M_i^{yz}, M_i^x m_i^{yz}, m_i^x M_i^{yz}, m_i^x m_i^{yz}\}. \tag{37}$$

Therefore, the upper bound and lower bound of $S(x, y, z)$ are

$$M = \sum_{i=1}^{j} \sigma_i M_i \tag{38}$$

and

$$m = \sum_{i=1}^{j} \sigma_i m_i. \tag{39}$$

The range of each entry of $\vec{L}(\cdot)$ is $[-1, 1]$. Therefore, $T(x, y, z)$ can be estimated by

$$\begin{aligned} |T(x,y,z)| &= \left| R \times_1 \vec{L}\left(\frac{2x-(a+b)}{b-a}\right) \times_2 \vec{L}\left(\frac{2y-(c+d)}{d-c}\right) \times_3 \vec{L}\left(\frac{2z-(e+f)}{f-e}\right)\right| \\ &\le \left|\sum R_{ijk}\right| \\ &\le \sum |R_{ijk}|, \end{aligned} \tag{40}$$

where R_{ijk} is the (i, j, k) entry of R.

As a result, the upper and lower bounds of $\tilde{P}(x, y, z)$ are

$$U = M + \sum |R_{ijk}|, \tag{41}$$

$$L = m - \sum |R_{ijk}|. \tag{42}$$

If the bounds satisfy the condition that $L \le c \le U$, the surface may have an intersection with the cube. Otherwise, the cube should be ruled out.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 1
The number of cubes after each time of partition for an ADP molecule.

Level	Number of all cubes	Number of empty cubes	Number of nonempty cubes
1	882	641	241
2	1928	597	1331
3	4516	128	4388

2.1.3. Approximation by trilinear polynomial. In each remaining cube, $\phi(x, y, z)$ is approximated by polynomial $\tilde{P}(x, y, z)$ as shown in (15). Then we divide each remaining cube into 8 smaller child cubes and express $\tilde{P}(x, y, z)$ in each child cube by Legendre polynomials as follows:

(43)
$$\tilde{P}(x,y,z) = A' \times_1 \vec{L}\left(\frac{2x-(a_1+b_1)}{b_1-a_1}\right) \times_2 \vec{L}\left(\frac{2y-(c_1+d_1)}{d_1-c_1}\right) \times_3 \vec{L}\left(\frac{2z-(e_1+f_1)}{f_1-e_1}\right),$$

where $x \in [a_1, b_1]$, $y \in [c_1, d_1]$, and $z \in [e_1, f_1]$. Here, $[a_1, b_1] \times [c_1, d_1] \times [e_1, f_1]$ is the range of the child cube, and A' is the coefficient tensor of the Legendre polynomials in the child cube. When the child cubes become smaller, the coefficients of the higher order Legendre polynomials in the coefficient tensor go to zero. The division process is repeated until the coefficients of the higher order Legendre polynomials are close to zero enough to be neglected in all the remaining cubes. Table 1 shows the number of empty cubes and nonempty cubes after each round of division in TMSmesh 2.0 for an adenosine diphosphate (ADP) molecule. The nonempty cubes intersect the Gaussian molecular surface. The empty cubes are detected by checking whether the isovalue c is between the upper and lower bounds of $\tilde{P}(x, y, z)$ in each cube. The level in Table 1 means the number of rounds of division. From Table 1, we can see that after the first round of division, more than 70% of the cubes are deleted. In the second and third rounds of division, some empty cubes are deleted, but they are a lower proportion. At the finest level, most of the cubes are not empty because the cubes are closer to the Gaussian surface, and the cubes that do not intersect the Gaussian surface are abandoned. Figure 2 shows that for ADP, after the adaptive division and estimation process, most cubes that do not intersect the Gaussian molecular surface are ruled out.

After the above division and estimation process, in each remaining cube, we approximate the surface $\tilde{P}(x, y, z) = c$ by the following trilinear interpolation. Supposing the range of the remaining cube is $[-1, 1] \times [-1, 1] \times [-1, 1]$, the trilinear interpolation can be written in terms of the vertex values:

$$(44)\quad g(x,y,z) = \frac{1}{8} \sum_{\alpha\in\{1,-1\}} \sum_{\beta\in\{1,-1\}} \sum_{\gamma\in\{1,-1\}} \tilde{P}(\alpha,\beta,\gamma)(1+\alpha*x)(1+\beta*y)(1+\gamma*z).$$

To ensure the continuity of trilinear interpolation between two adjacent boxes with different sizes, the vertex value of the smaller box should be drawn from the trilinear interpolation of the adjacent larger box.

2.2. Triangulating the trilinear surface. In this subsection, we introduce our method of triangulating the piecewise trilinear surface in cubes with different sizes. Without loss of generality, suppose in cube $[-1, 1] \times [-1, 1] \times [-1, 1]$ the trilinear surface is $g(x, y, z) = c$.

This method contains three steps, which are shown in Figure 3. This figure shows the triangulation process in two neighbored cubes. First, the intersection points

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

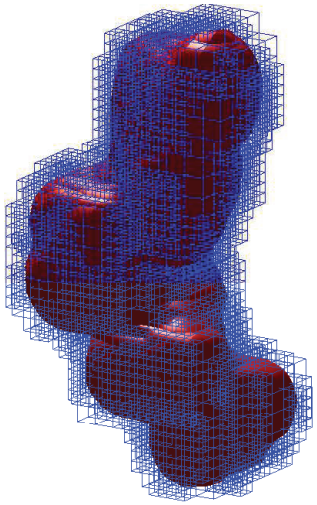


FIG. 2. *The cubes intersecting the Gaussian molecular surface for an ADP molecule. These cubes are represented in blue, and the molecular surface is represented in red.*

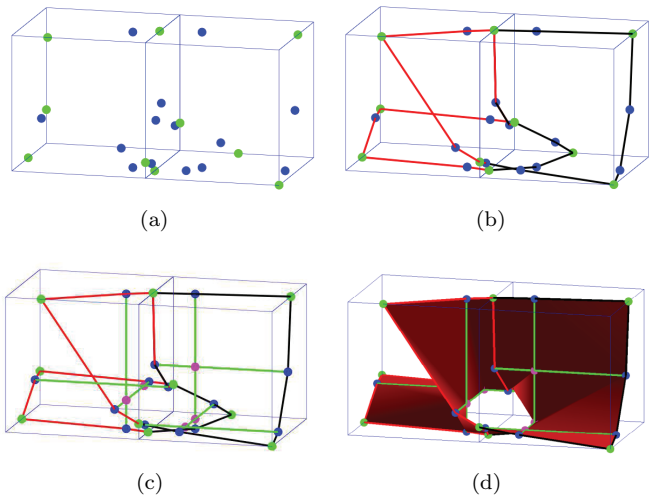


FIG. 3. *Method of triangulating the trilinear surface.* (a) *Step* 1: *Compute the intersection points and extreme points on the faces of the cubes. The green points are intersection points between the surface and the edge of the cube, and the blue points are extreme points.* (b) *Step* 2: *Connect the green intersection points and the blue extreme points by surface curves on the faces of the cubes to form the red loop in the remaining cube and the black loop in the right cube.* (c) *The green lines are fold curves, and the magenta points are critical points.* (d) *Step* 3: *The surface patches enclosed by the red and black loops are divided into six single valued pieces by the fold curves.*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

between $g(x,y,z)=c$ and the edges of a cube are computed. They are defined as

$$\begin{cases} g(x,y,z)=c, \\ \alpha=a, \quad \alpha,\beta\in\{x,y,z\}, \alpha\neq\beta, a,b\in\{1,-1\}, \\ \beta=b. \end{cases} \tag{45}$$

The extreme points on the faces of cubes are also computed. The extreme points in the x, y, and z directions are defined as

$$\begin{cases} g(x,y,z)=c, \\ \alpha=a, \quad \alpha\in\{x,y,z\}, a\in\{1,-1\}, \\ \dfrac{\partial g(x,y,z)}{\partial\alpha}=0. \end{cases} \tag{46}$$

Second, the intersection points and extreme points defined by (45) and (46) are connected by surface curves on the faces of cubes and form closed loops. Since the surface curves on the faces of cubes are simple hyperbola and the expression of the curves is explicit, it is easy to determine which two points are neighbored in the same branch of the hyperbola. The surface patches enclosed by these loops may contain holes and tunnels. In the third step, the surface patches are divided into single valued pieces along the x, y, and z directions by fold curves. "Single valued pieces" here means that on each piece of this type, each component of the normal direction (g_x, g_y, g_z) does not change sign. Here the fold curves for the x, y, and z directions are defined as

$$\left\{ g(x,y,z)=c, \frac{\partial g(x,y,z)}{\partial\alpha}=0 \right\}, \quad \alpha\in\{x,y,z\}. \tag{47}$$

Generally, the fold curves are not straight lines (see Figure 3 in [7]). But for the trilinear surface, the fold curves are straight line segments whose ends are extreme points. The fold curves along different directions may have intersections; they are critical points satisfying

$$\begin{cases} g(x,y,z)=c, \\ \dfrac{\partial g(x,y,z)}{\partial\alpha}=0, \quad \alpha,\beta\in\{x,y,z\},\ \alpha\neq\beta, \\ \dfrac{\partial g(x,y,z)}{\partial\beta}=0. \end{cases} \tag{48}$$

Figure 4 shows an example of subdividing a surface patch into single valued pieces along the x, y, and z directions by fold curves. The trilinear surface defined in (44) is folded at the fold curves. Cutting the trilinear surface along these fold curves ensures that on each of the resulting pieces, the signs of the x, y, and z components of the normal direction are invariant. Subdividing the loops along fold curves helps avoid incorrect connections during triangulation and helps find missed small surface structures, such as tunnels and holes, because these structures also fold at these curves. After the third step, each single valued piece is homomorphic to a two-dimensional polygon and can be triangulated by a standard method, such as ear clipping [15, 27], monotone polygons [11], Seidel's decomposition algorithm [33], or Chazelle's triangulation method [5].

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

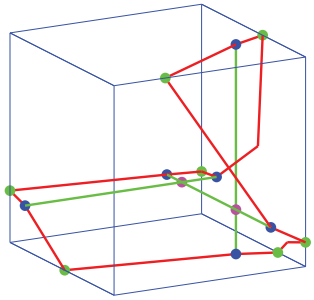


FIG. 4. *An example of connecting surface extreme points along the fold curves. The green points are intersection points, and the blue points are extreme points. The green lines are fold curves. The red curve forms a closed loop on the trilinear surface. The surface patch enclosed by the red loop is not single valued along the x, y, z directions, and it is divided into six single valued pieces along the x, y, z directions by the fold curves.*

TABLE 2
Test molecules in the PQR benchmark which can be downloaded from http://lsec.cc.ac.cn/~lubz/Download/PQR_benchmark.tar.

Molecule (name or PDB code)	Number of atoms	Description
GLY	7	a single glycine residue
ADP	39	ADP molecule
2LWC	75	Met-enkephalin in DPMC SUV
FAS2	906	fasciculin2, a peptidic inhibitor of AChE
AChE monomer	8280	mouse acetylcholinesterase monomer
AChE tetramer	36638	the structure of AChE tetramer, taken from [29]
30S ribosome	88431	30S ribosome, the PDB code is 1FJF
70S ribosome	165337	obtained from 70S_ribosome3.7A_model140.pdb.gz on http://rna.ucsc.edu/rnacenter/ribosome_downloads.html
3K1Q	203135	PDB code, a backbone model of an aquareovirus virion
2X9XX	510727	a complex structure of the 70S ribosome bound to release factor 2 and a substrate analog, which has 4 split PDB entries: 2X9R, 2X9S, 2X9T, and 2X9U
1K4R	1082160	PDB code, the envelope protein of the dengue virus

To summarize, the manifoldness of the generated mesh is guaranteed at algorithm level. First, the summation of Gaussian kernels is approximated by the piecewise nth-degree polynomial with C^0 continuity. And piecewise trilinear approximation is used in the surface extraction which also preserves the C^0 continuity between adjacent boxes. Therefore the piecewise trilinear surface is a manifold. Second, in each cube, the trilinear surface is divided into single valued pieces by fold curves. According to Morse theory [28], for nondegenerated surfaces, each single valued piece enclosed by the fold curves is homomorphic to a two-dimensional polygon, and they do not intersect each other. Third, each single valued surface piece is homomorphic to a two-dimensional polygon and can be triangulated into triangles without intersections by a standard polygon triangulation algorithm.

3. Experimental results.

3.1. Efficiency and robustness. Because MSMS is the most widely used efficient software for molecular surface triangulation, in this section, the performance

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 3
CPU time used for molecular surface generation by TMSmesh and MSMS.

Molecule	Natoms	Number of vertices			CPU time		
		TMSmesh	TMSmesh 2.0	MSMS	TMSmesh	TMSmesh 2.0	MSMS
FAS2	906	5170	6849	5258	6.4	0.36	0.13
		8309	8579	7888	8	0.43	0.18
AChE monomer	8280	24556	45711	34819	52	1.79	0.72
		39289	63836	51784	60	2.05	0.96
AChE tetramer	36638	95433	163736	132803	224	5.90	4.99
		152035	220089	192545	260	6.91	5.94
30S ribosome	88431	274297	489325	353272	721	14.89	13.21
		439020	631448	520986	1120	17.59	15.43
70S ribosome	165337	698055	869930	845550	1218	24.12	36.44
		1111399	1160622	Fail	1361	30.34	Fail
3K1Q	203135	509390	678915	666517	1440	26.92	36.85
		812774	975334	984234	1728	30.72	40.48
2X9XX	510727	1585434	2132433	Fail	4809	68.64	Fail
		2521233	2933346	Fail	5762	84.71	Fail
1K4R	1082160	3325975	4050952	Fail	7296	141.51	Fail
		5298234	5540049	Fail	12905	178.85	Fail

of TMSmesh 2.0 is compared with those of MSMS and the old version of TMSmesh. A set of biomolecules with different sizes is chosen as a test benchmark (see Table 2); this set was used in our previous work [6, 7] and can be downloaded from http://lsec.cc.ac.cn/~lubz/Download/PQR_benchmark.tar. The meshing softwares are run on molecular PQR files (PDB plus atomic charges and radii information). To make a reasonable comparison with MSMS, appropriate parameters, such as the error tolerance between Gaussian surface and approximated piecewise trilinear surface, are chosen for TMSmesh to achieve the surface vertex densities $1/\mathring{A}^2$ and $2/\mathring{A}^2$ used in MSMS mesh generation. The probe radius in MSMS is set to be $1.4\mathring{A}$. All computations were run on a computer with Intel Xeon CPU E5-4650 v2, 2.4GHz, and 126GB memory under a 64-bit Linux system.

Table 3 shows the CPU time cost of MSMS and TMSmesh with 1 and 2 vertex/$\mathring{A}^2$ mesh densities. In Table 3, TMSmesh denotes the old version in 2012 [7], and TMSmesh 2.0 is the new version used in this paper. The discrepancies between the numbers of vertices of the TMSmesh mesh and the MSMS mesh are due to different definitions of molecular surface and different meshing methods used in the two programs. The CPU time cost of TMSmesh 2.0 is much less than the cost of the old version of TMSmesh. TMSmesh 2.0 is at least 30 times faster than the old version. This is due to the following reasons. First, the new adaptive way of partitioning the process to locate the surface reduces the number of surface-intersecting cubes. We use different sizes of cubes according to the approximation accuracy of the piecewise trilinear surface in the new method instead of using the same sized cubes from the previous method. Fewer cubes are used to precisely locate the surface. Second, a more efficient and much sharper bound estimator of summation of Gaussian kernels in a cube is adopted, as shown in section 2.1.2. Third, the trilinear polynomials are used to approximate the surface to reduce computation costs greatly. For the trilinear surface, the surface points and fold curves can be computed explicitly, and the fold curves are explicitly straight lines, which makes the tracing process easier.

For the small molecules, the CPU time cost of MSMS is less than that of TMSmesh 2.0. But for the large molecules, MSMS requires more time than TMSmesh 2.0. This is because the computational complexity of MSMS is $O[N\log(N)]$, where N is the number of atoms. And the complexity of TMSmesh 2.0 is almost $O(N)$, which is shown in Figure 5. In TMSmesh 2.0, as the exponential kernels $e^{-||\vec{x}-\vec{x}_i||^2}$ in the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

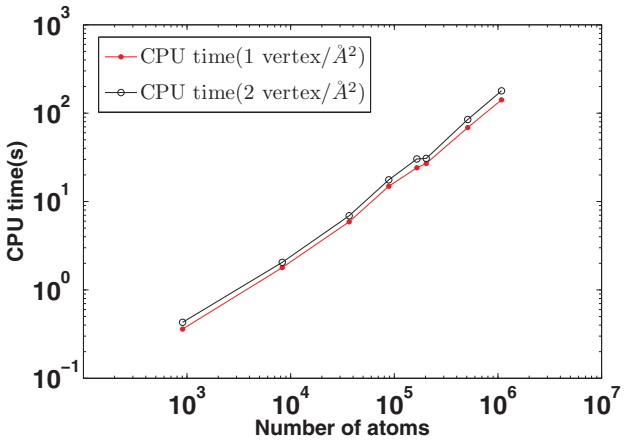


FIG. 5. *Computational performance of TMSmesh* 2.0.

Gaussian surface decay very quickly when the distances $||\vec{x} - \vec{x}_i||$ become large, all the calculations are done locally, and so no global information is needed in the whole process. TMSmesh 2.0 can successfully generate surface mesh for the biomolecules consisting of more than one million atoms, such as the dengue virus 1K4R. Because the virus's structure is among the largest in the Protein Data Bank, together with consideration of good algorithm stability, TMSmesh 2.0 is capable of handling biomolecules with arbitrary sizes.

3.2. Manifoldness and faithfulness. We study the manifoldness of the meshes generated by TMSmesh 2.0 and MSMS. The generated surface meshes should be manifold. A nonmanifold mesh can cause numerical problems in BEM and FEM simulations of biomolecules. Also, a nonmanifold surface cannot be directly used to generate the corresponding volume mesh due to its nonmanifold errors, such as intersections of triangles. The previous TMSmesh version has been shown to be able to guarantee manifold mesh generation [7]. Here, we check whether the meshes produced by TMSmesh 2.0 and MSMS are manifolds. A manifold mesh for a closed molecular surface should satisfy the following three necessary conditions [7]:

(a) Each edge should be shared by two and only two faces of the mesh.
(b) Each vertex should have one and only one neighborhood node loop.
(c) The mesh has no intersecting face pairs.

Table 4 shows the number of nonmanifold defects and the number of intersecting triangle pairs in the meshes produced by TMSmesh 2.0 and MSMS. Here, the number of nonmanifold defects is the number of vertices whose neighborhood does not satisfy aforementioned necessary conditions (a) and (b) for a manifold mesh. The meshes produced by TMSmesh 2.0 all satisfy the three necessary conditions for a manifold mesh. However, the meshes of large biomolecules generated by MSMS are not manifold.

It is also important to keep the features of the biomolecular surface. The meshes generated by our method are faithful to the original surface due to the following reasons. In the first step of our method, the Gaussian surface is approximated by a piecewise nth order polynomial surface on a uniform grid. In step two, we do the adaptive division and estimation process until the nth order polynomial in each

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 4
Number of nonmanifold errors in meshes produced by TMSmesh 2.0 *and MSMS.*

Molecule	Natoms	Number of nonmanifold defects		Number of intersecting triangle pairs	
		TMSmesh 2.0	MSMS	TMSmesh 2.0	MSMS
FAS2	906	0	0	0	0
		0	0	0	0
AChE monomer	8280	0	0	0	220
		0	0	0	265
AChE tetramer	36638	0	3	0	499
		0	50	0	662
30S ribosome	88431	0	2	0	1583
		0	4	0	2504
70S ribosome	165337	0	11	0	5235
		0	Fail	0	Fail
3K1Q	203135	0	15	0	893
F		0	15	0	1890
2X9XX	510727	0	Fail	0	Fail
		0	Fail	0	Fail
1K4R	1082160	0	Fail	0	Fail
		0	Fail	0	Fail

box can be approximated well enough by trilinear interpolation. In the third step, the piecewise trilinear surface is triangulated. So the differences between the final triangle meshes between the Gaussian surface are influenced by the approximation errors in these three steps. And the final voxel density is also affected by the approximated error in these three steps. When the error in the first step is large and the error in the second step goes to zero, the voxel density can also go to infinity. So we cannot say that only the voxel density is sufficient for capturing all features of the Gaussian surface. We only can state that our method finds all feature lines on the approximated piecewise trilinear surface. According to Morse theory [28], for a nondegenerate surface, the topological change appears around the critical points. In our algorithm, we find all the critical points and the fold curves across the critical points for the trilinear surface, which can guarantee to characterize all the features of the trilinear surface.

3.3. Boundary element method simulation. The surface mesh generated by TMSmesh 2.0 can be applied not only to molecular visualization and analysis of surface area, topology, and volume in computational structure biology and structural bioinformatics, but also to BEM simulations. We test the meshes in boundary element calculations of the Poission–Boltzmann electrostatics. The BEM software used is a publicly available PB solver, AFMPB [41]. As a representative molecular system, we choose the structure AChE monomer (see Table 2). The surface mesh is generated by TMSmesh 2.0 and contains 87044 nodes. Figure 6 shows the computed electrostatic potentials mapped on the molecular surface. The surface potential correctly captures the molecular charge property, which verifies the effectiveness and applicability of the mesh generated by our method.

3.4. Convergence. Figure 7 shows the surface areas and molecular volumes as well as the solvation energies computed from the meshes of three small molecules, GLY, ADP, and 2LWC (see Table 2), using different mesh densities. The surface area S is determined using the following formula:

$$S = \frac{1}{2}\sum_{i=1}^{m} \|\overrightarrow{A_2^i A_1^i} \times \overrightarrow{A_3^i A_1^i}\|, \tag{49}$$

where m is the number of triangle elements and A_1^i, A_2^i, A_3^i denote the coordinates of the three vertices for the ith triangle. The volume V enclosed by the surface mesh

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

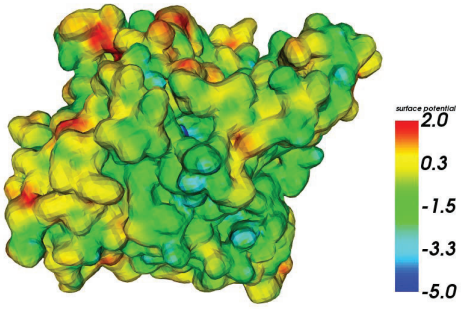


FIG. 6. *Electrostatic potential surface of AChE calculated by AFMPB.*

is determined using the following formula:

$$V = \frac{1}{6}\sum_{i=1}^{m} \overrightarrow{A_2^i A_1^i} \times \overrightarrow{A_3^i A_1^i} \bullet \overrightarrow{OA_1^i}, \tag{50}$$

where O is the origin point. The solvation energy is computed by a boundary element Poisson–Boltzmann solver called AFMPB [41].

The red lines in Figure 7 are generated by TMSmesh 2.0, and the black lines are generated by TMSmesh. The results show that the meshes produced by TMSmesh 2.0 lead to convergent and reasonable results for energy, area, and volume when the mesh density increases. The largest disparities between the results of TMSmesh 2.0 and TMSmesh are less than 3%. In the conditions of high vertex density, the disparities of the results are less than 1%. The results computed by TMSmesh converge a little more smoothly than those of TMSmesh 2.0 when the number of triangles is small. This is because we use the trilinear polynomial instead of a high order polynomial to approximate the Gaussian surface. Having fewer triangles leads to lower precision of the approximation, which causes more uncertainties.

3.5. Volume mesh generation conforming surface mesh. The surface mesh generated by TMSmesh 2.0 can be directly used to generate the corresponding surface conforming volume mesh. And the volume mesh generated by this method can be applied to FEM simulation directly. Figure 8 shows a cross section of the volume mesh for the ion channel, VDAC (PDB code: 2JK4). The VDAC serves an essential role in the transport of metabolites and electrolytes between the cell matrix and mitochondria [2]. For this example, the molecular surface mesh is generated by TMSmesh 2.0, and the corresponding volume mesh is generated by TetGen [34]. The channel pore is clearly represented in the mesh, and the detailed topology is correctly preserved, which is important for ion channel simulations. In addition, from the cross section we can see that the surface mesh is dense at the rugged parts and sparse at the smooth parts.

4. Conclusion. We have described a new algorithm in TMSmesh 2.0 for triangulating the Gaussian molecular surface. In TMSmesh 2.0, an adaptive surface partition is developed using a new method to estimate the upper and lower bounds of the surface function in a cell. In each located cube, a trilinear polynomial is used to approximate the Gaussian surface within controllable precision. The fold curves are

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

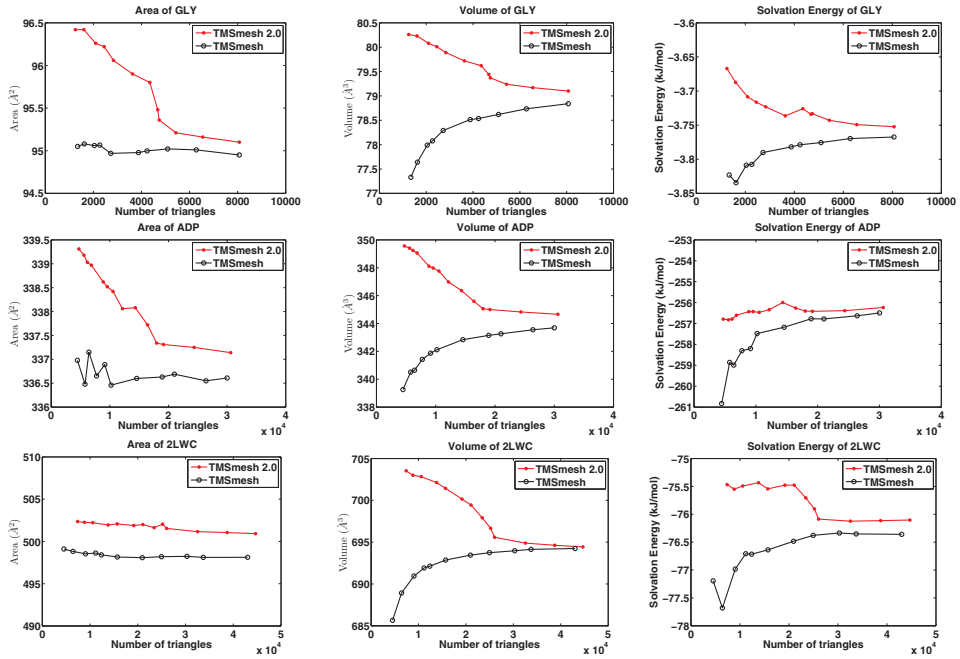


FIG. 7. *Area (left column), volume (middle column), and solvation energy (right column) for GLY (first row), ADP (second row), and 2LWC (third row).*

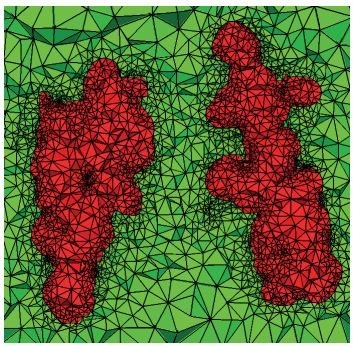


FIG. 8. *A cross section of the volume mesh for VDAC. The surface mesh is generated by TMSmesh* 2.0*, and the volume mesh is generated by TetGen.*

used to divide the trilinear surface in each cube into single valued pieces to guarantee a manifold mesh generation. Compared with the old version, TMSmesh 2.0 is more than 30 times faster. TMSmesh 2.0 is shown to be a robust and efficient software to mesh the Gaussian molecular surface. The meshes generated by TMSmesh 2.0 are manifold without intersections. And the mesh can be directly used in boundary element type simulations and volume mesh generations.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

REFERENCES

[1] P. W. Bates, G. W. Wei, and S. Zhao, *Minimal molecular surfaces and their applications*, J. Comput. Chem., 29 (2008), pp. 380–391.

[2] M. Bayrhuber, T. Meins, M. Habeck, S. Becker, K. Giller, S. Villinger, C. Vonrhein, and C. Griesinger, *Structure of the human voltage-dependent anion channel*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 15370–15375.

[3] T. Can, C. Chen, and Y. Wang, *Efficient molecular surface generation using level-set methods*, J. Molecular Graphics Model., 25 (2006), pp. 442–454.

[4] M. Chavent, B. Levy, and B. Maigret, *Metamol: High-quality visualization of molecular skin surface*, J. Molecular Graphics Model., 27 (2008), pp. 209–216.

[5] B. Chazelle, *Triangulating a simple polygon in linear time*, Discrete Comput. Geom., 6 (1991), pp. 485–524.

[6] M. Chen and B. Lu, *TMSmesh: A robust method for molecular surface mesh generation using a trace technique*, J. Chem. Theory Comput., 7 (2011), pp. 203–212.

[7] M. Chen, B. Tu, and B. Lu, *Triangulated manifold meshing method preserving molecular surface topology*, J. Molecular Graphics Model., 38 (2012), pp. 411–418.

[8] H.-L. Cheng and X. Shi, *Quality mesh generation for molecular skin surfaces using restricted union of balls*, Comput. Geom., 42 (2009), pp. 196–206.

[9] M. L. Connolly, *Analytical molecular surface calculation*, J. Appl. Crystallogr., 16 (1983), pp. 548–558.

[10] M. L. Connolly, *Solvent-accessible surfaces of proteins and nucleic acids*, Science, 221 (1983), pp. 709–713.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry*, Springer-Verlag, New York, 2000; Chapter 3: Polygon Triangulation, pp. 45–61.

[12] S. Decherchi and W. Rocchia, *A general and robust ray casting based algorithm for triangulating surfaces at the nanoscale*, PLoS One, 8 (2013), e59744.

[13] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, and N. A. Baker, *PDB2PQR: An automated pipeline for the setup, execution, and analysis of Poisson-Boltzmann electrostatics calculations*, Nucleic Acids Res., 32 (2004), pp. W665–W667.

[14] B. S. Duncan and A. J. Olson, *Shape analysis of molecular surfaces.*, Biopolymers, 33 (1993), pp. 231–238.

[15] D. Eberly, *Triangulation by Ear Clipping*, Geometric Tools, LLC, http://www.geometrictools.com/, 1998.

[16] H. Edelsbrunner, *Deformable smooth surface design*, Discrete Comput. Geom., 21 (1999), pp. 87–115.

[17] H. Edelsbrunner and E. P. Mucke, *Three-dimensional alpha shapes*, ACM Trans. Graphics, 13 (1994), pp. 43–72.

[18] J. A. Grant, M. A. Gallardo, and B. T. Pickup, *A fast method of molecular shape comparison: A simple application of a Gaussian description of molecular shape*, J. Comput. Chem., 17 (1996), pp. 1653–1666.

[19] T. Ju, F. Losasso, S. Schaefer, and J. D. Warren, *Dual contouring of hermite data*, ACM Trans. Graphics, 21 (2002), pp. 339–346.

[20] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500, https://doi.org/10.1137/07070111X.

[21] B. Lee and F. Richards, *The interpretation of protein structures: Estimation of static accessibility*, J. Mol. Biol., 55 (1971), pp. 379–400.

[22] T. Liao, Y. Zhang, P. M. Kekenes-Huskey, Y, Cheng, A. Michailova, A. McCulloch, M. Holst, and A. McCammon, *Multi-core CPU or GPU-accelerated multiscale modeling for biomolecular complexes*, Mol. Based Math. Biol., 1 (2013), pp. 164–179.

[23] T. Liu, M. Chen, and B. Lu, *Parameterization for molecular Gaussian surface and a comparison study of surface mesh generation*, J. Molecular Model., 21 (2015), 113.

[24] W. Lorensen and H. E. Cline, *Marching cubes: A high resolution* 3*d surface construction algorithm*, Computer Graphics, 21 (1987), pp. 163–169.

[25] B. Z. Lu, Y. C. Zhou, M. J. Holst, and J. A. MaCammon, *Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications*, Commun. Comput. Phys., 3 (2008), pp. 973–1009.

[26] M. R. Mcgann, H. R. Almond, A. Nicholls, A. J. Grant, and F. K. Brown, *Gaussian docking functions*, Biopolymers, 68 (2003), pp. 76–90.

[27] G. H. Meisters, *Polygons have ears*, Amer. Math. Monthly, 82 (1975), pp. 648–651.

[28] J. Milnor, *Morse Theory*, Princeton University Press, Princeton, NJ, 1963.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[29] A. Nicholls, R. Bharadwaj, and B. Honig, *Grasp: Graphical representation and analysis of surface properties*, Biophys. J., 64 (1995), pp. 166–167.
[30] F. M. Richards, *Areas, volumes, packing and protein structure*, Ann. Rev. Biophys. Bioengineering, 6 (1977), pp. 151–176.
[31] J. Ryu, R. Park, and D.-S. Kim, *Molecular surfaces on proteins via beta shapes*, Comput. Aided Des., 39 (2007), pp. 1042–1057.
[32] M. Sanner, A. Olson, and J. Spehner, *Reduced surface: An efficient way to compute molecular surface properties*, Biopolymers, 38 (1996), pp. 305–320.
[33] R. Seidel, *A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons*, Comput. Geom. Theory Appl., 1 (1991), pp. 51–64.
[34] H. Si, *TetGen, a Delaunay-based quality tetrahedral mesh generator*, ACM Trans. Math. Softw., 41 (2015), 11.
[35] Y. Vorobjev and J. Hermans, *SIMS: Computation of a smooth invariant molecular surface*, Biophys. J., 73 (1997), pp. 722–732.
[36] J. Weiser, P. Shenkin, and W. Still, *Optimization of Gaussian surface calculations and extension to solvent-accessible surface areas*, J. Comput. Chem., 20 (1999), pp. 688–703.
[37] D. Xu and Y. Zhang, *Generating triangulated macromolecular surfaces by Euclidean distance transform*, PLoS ONE, 4 (2009), e8140.
[38] Z. Yu, M. J. Holst, and J. McCammon, *High-fidelity geometric modeling for biomedical applications*, Finite Elem. Anal. Des., 44 (2008), pp. 715–723.
[39] Z. Yun, P. Matthew, and A. Richard, *What role do surfaces play in GB models? A new-generation of surface-generalized Born model based on a novel Gaussian surface for biomolecules*, J. Comput. Chem., 27 (2005), pp. 72–89.
[40] R. Zauhar, *SMART: A solvent-accessible triangulated surface generator for molecular graphics and boundary element applications*, J. Computer-Aided Mol. Des., 9 (1995), pp. 149–159.
[41] B. Zhang, B. Peng, J. Huang, N. P. Pitsianis, X. Sun, and B. Lu, *Parallel AFMPB solver with automatic surface meshing for calculations of molecular solvation free energy*, Comput. Phys. Comm., 190 (2015), pp. 173–181.
[42] Y. Zhang, G. Xu, and C. Bajaj, *Quality meshing of implicit solvation models of biomolecular structures*, Comput. Aided Geom. Des., 23 (2006), pp. 510–530.

Downloaded 04/17/18 to 124.16.148.8. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.